UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

Sistema de monitoreo y control de alimentador y bebedero “Smart Feeder”

Autores : René Ayca
Claudio Carvajal
Yazuska Castillo
Israel Tebes
Profesor: Diego Aracena
Asignatura: Proyecto II

ARICA, 26 Noviembre 2025

Historial de Cambios

Fecha	Versión	Descripción	Autor(es)
15/10/2025	1.0	Versión preliminar del formato	Rene Ayca Claudio Carvajal Yazuska Castillo Israel Tebes
21/10/2025	1.1	Revisión y modificación del plan	Rene Ayca Claudio Carvajal Yazuska Castillo Israel Tebes
22/10/2025	1.2	Roles y Responsabilidades y Estimación de Costos	Rene Ayca Claudio Carvajal Yazuska Castillo Israel Tebes
26/10/2025	1.3	Planificación de la gestión de riesgos	Rene Ayca Claudio Carvajal Yazuska Castillo Israel Tebes
28/10/2025	1.4	Revisión final Informe 1	Rene Ayca Claudio Carvajal Yazuska Castillo Israel Tebes
18/11/2025	1.5	Corrección Informe 1	Rene Ayca Claudio Carvajal Yazuska Castillo Israel Tebes
19/11/2025	1.6	Planificación de los procesos técnicos	Rene Ayca Claudio Carvajal Yazuska Castillo Israel Tebes
20/11/2025	1.7	Revisión final informe II	Rene Ayca Claudio Carvajal Yazuska Castillo Israel Tebes
20/12/2025	1.8	Implementación, problemas y soluciones y trabajo futuro	Rene Ayca Claudio Carvajal Yazuska Castillo Israel Tebes
24/12/2025	1.9	Revisión final informe III	Rene Ayca Claudio Carvajal Yazuska Castillo Israel Tebes

Tabla de contenidos

Índice de Tablas

Índice de Figuras

1. Panorama General

1.1. Resumen del Proyecto

La creciente posesión de mascotas en las casas presenta un problema bastante común: garantizar la correcta distribución de comida y agua (que de ahora en adelante se abrevió en alimentación para mayor simplicidad) durante las largas ausencias de sus dueños o responsables, ya sea porque simplemente no están en casa o debido a viajes. La falta de una solución automatizada y confiable puede llevar a una mascota a una alimentación irregular, con raciones de comida inexactas o, en el peor de los casos, a que se quede sin comida o agua, afectando directamente su bienestar y salud.

La complejidad de este cuidado empieza cuando se necesita mantener no solo disponibilidad de alimento y agua, sino también un horario consistente y el control de las raciones de alimento, algo difícil de lograr mediante métodos manuales cuando el propietario no está presente. A las soluciones actuales les suele faltar inteligencia y el control remoto necesario para ofrecer tranquilidad real al usuario.

Para abordar esta problemática, se propone un Dispensador Automático IoT, un sistema inteligente que integra sensores y control remoto (app) para automatizar y supervisar la alimentación de las mascotas. El dispositivo utiliza un sensor de peso en el plato para medir con precisión la cantidad de comida consumida en gramos, permitiendo un seguimiento detallado de la dieta del animal. Todo el sistema es gestionado por una Raspberry Pi 4, que se conecta con los sensores y ejecuta órdenes programadas.

Se probó la idea del dispensador y su ambiente respectivo gracias a los meta quest 3. Donde se creó un escenario para poder identificar dónde estaría este dispensador. Además de mostrar algunas características que posteriormente serán mencionadas. La creación tanto de un escenario como el dispensador, sirvió para orientar y guiar la idea del proyecto.

La solución incluye una aplicación móvil remota que permite a los dueños interactuar con el dispensador desde cualquier lugar. A través de esta interfaz, es posible programar horarios de alimentación, dispensar comida o agua de forma manual, y recibir información en tiempo real sobre el consumo de la mascota. De este modo, el sistema no solo automatiza la provisión de recursos, sino que también proporciona datos para el cuidado responsable del animal.

En resumen, este dispensador IoT no solo soluciona el problema de la alimentación en ausencia del dueño, sino que también controla y supervisa el bienestar de la mascota. Al construir el prototipo con elementos reciclables, el proyecto refuerza además su compromiso con la sostenibilidad. Esta solución garantiza el bienestar de la mascota y de los dueños o familias a cargo de las mascotas, transformando el cuidado animal en una experiencia más conectada, eficiente y segura.

1.1.1. Propósito

El propósito de este proyecto es diseñar e implementar un sistema IoT que mejore el bienestar de las mascotas mediante la automatización de su alimentación e hidratación. Se busca ofrecer a los dueños una herramienta de control y monitoreo, que les pueda dar seguridad y tranquilidad al garantizar que sus mascotas reciban su alimentación necesaria, incluso durante sus ausencias prolongadas.

1.1.2. Alcance

Este proyecto abarca el diseño, desarrollo e implementación de un sistema automatizado de alimentación e hidratación para mascotas, compuesto por un dispensador de comida y agua. Incluirá sensores ultrasónicos y de peso, además de una cámara para el monitoreo en tiempo real. Mediante la conexión de un Raspberry Pi que recibirá los datos enviados por los sensores para analizar y verificar que los niveles están dentro de lo establecido.
En caso de detectar niveles bajos, el sistema activará los motores correspondientes para dispensar alimentos o agua automáticamente, garantizando el suministro necesario para el bienestar de la mascota. Además de una aplicación de software que permitirá la interacción del usuario.

1.1.3. Objetivo General

Desarrollar un dispensador automático IoT que se encargue de la alimentación e hidratación precisa para las mascotas. Permitiendo el control y monitoreo remoto por parte del dueño, con el fin de garantizar el bienestar del animal durante largas ausencias.

1.1.4. Objetivo Específicos

- Adquirir los conocimientos necesarios para el desarrollo del sistema.
- Diseñar maqueta y diagrama del dispensador, definiendo ubicación de los componentes y seleccionando e incorporando componentes hardware clave para el funcionamiento del dispensador.
- Planificar el desarrollo del proyecto para un avance eficiente.
- Desarrollar una aplicación móvil que permita la conexión con el sistema, la programación de horarios de alimentación y el monitoreo del estado de la comida y el agua.
- Documentar el desarrollo, resultados y conclusiones del proyecto realizado.
- Realizar pruebas para asegurar que el monitoreo y control automatizado funcione correctamente.

1.1.5. Suposiciones y restricciones

- **Suposiciones:**
    - Se asume que el usuario tendrá acceso a una red Wi-Fi estable en el lugar donde se instale el dispensador, para permitir la conectividad IoT.
    - Se asume que el usuario contará con un smartphone compatible y los conocimientos básicos para instalar y utilizar la aplicación móvil.
    - Se asume que el usuario proveerá alimento seco de tamaño y forma adecuados para que no cause atascos en el mecanismo de dispensación.
- **Restricciones:**
    - La construcción del prototipo del dispensador debe ser con materiales reciclables o de reuso, lo que limitará las opciones de diseño y durabilidad en comparación con materiales normales.
    - El sistema depende de una fuente de energía eléctrica continua. Un corte de energía interrumpirá la operación automatizada hasta que se restablezca.
    - La capacidad de los depósitos de alimento y agua está limitada por el diseño físico del prototipo, requiriendo recarga manual periódica por parte del usuario.

1.1.6. Entregables del Proyecto

Los entregables del proyecto son los siguientes:

1. Maqueta del sistema
2. Presentación de la maqueta
3. Informes del proyecto
4. Presentaciones del proyecto
5. Redmine UTA (wiki, bitácoras y carta Gantt)
6. Manual de usuario
7. Sistema “Smart Feed”

2. Organización del proyecto

2.1. Personal y entidades internas

Para desarrollar de forma óptima el proyecto, cada integrante del equipo fue asignado a un rol específico con responsabilidades claras, que son descritas a continuación.

- Jefe de proyecto: Representante del equipo. Supervisa y organiza el progreso del proyecto, tomando decisiones para el cumplimiento de los objetivos planteados en un plazo estipulado
- Analista: Encargado de analizar requerimientos y asegurar la integración entre hardware, software y nube del sistema IoT.
- Diseñador: Encargado del diseño de las presentaciones, del diseño del prototipo del dispensador y también del diseño de la interfaz de usuario.
- Programador: Encargado del área de la codificación y funcionamiento del Raspberry
- Documentador: Encargado de registrar el avance del proyecto, junto con la redacción de los informes.

2.2. Roles y responsabilidades

Rol	**Responsable**	**Involucrados**
Jefe de Proyecto	René Ayca	—
Analista	René Ayca	Yazuska Castillo Israel Tenes
Diseñador	Claudio Carvajal	—
Programador	Yazuska Castillo	Claudio Carvajal Israel Tebes René Ayca
Documentador	Israel Tebes	Yazuska Castillo

Tabla 1: Roles y Responsabilidades

2.3. Mecanismos de Comunicación

Durante el desarrollo del proyecto, se implementaron distintas herramientas para optimizar la comunicación, la coordinación y para tener un seguimiento de las actividades realizadas.

- Discord: Utilizado para realizar reuniones, ya que se puede realizar llamadas grupales, facilitando la discusión de los avances realizados, planificación y toma de decisiones grupales.
- WhatsApp: Utilizado para comunicarse de forma ágil para coordinar horarios, enviar recordatorios y enviar información breve como links, documentos, etc.
- Redmine: Utilizado para la gestión formal del proyecto, a través de esta herramienta se documentan avances, se guardan documentos importantes y se lleva un control de progreso mediante la carta Gantt.
- Drive: Utilizado para la organización de archivos, documentos, links, videos y referencias relacionadas con el proyecto.

3. Planificación de proceso de gestión

3.1. Planificación inicial del proyecto

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

- Raspberry Pi 4
- Sensores
- Motores
- Válvula solenoide
- NoteBook
- SmartPhone

Mientras que para la parte de software:

- Unity hub
- Canva
- Licencia Microsoft Office
- Visual Studio Code (Python)

3.1.1. Planificación de estimaciones

- Hardware

Producto	Cantidad	Costo por unidad	Costo total
Vivobook 16X	1 (30%)	$619.990	$185.997
HP Victus Gaming 15	1 (30%)	$859.990	$257.997
HP Victus Gaming 16	1 (30%)	$1.299.990	$389.997
HP 348 G7	1 (30%)	$799.990	$239.997
Raspberry Pi	1	$116.990	$116.900
Sensor de Peso	2	$5.500	$11.000
Sensor Ultrasónico	2	$2.590	$5.180
Cámara Raspberry	1	$4.753	$4.753
Válvula solenoide	1	$5.229	$5.229
Tornillo sin fin	1	$5.500	$5.500
		Total	**$1.222.550**

Tabla 2: Costos de Hardware.

- Software

Producto	Costo	Costo Total
Unity hub	Gratuito	Gratuito
Canva	Gratuito	Gratuito
Licencia Microsoft Office	$4.240/mes	$16.960 (4 meses)
Visual Studio Code	Gratuito	Gratuito
Drive	Gratuito	Gratuito
	Total	**$16.960**

Tabla 3: Costos de Software.

3.1.2. Planificación de Recursos humanos

Rol	Cantidad por rol	Costo/Hora	Horas Mensuales totales	Costo Total
Jefe de proyecto	1	$4.000 CLP/Hora	48	$192.000.
Programador	4	$5.231 CLP/Hora	48	$1.004.352
Diseñador	1	$3.692 CLP/Hora	48	$177.216.
Documentador	2	$4.082 CLP/Hora	48	$391.872
Analista	2	$4.923 CLP/Hora	48	$472.608
			Total por 1 mes	**$2.138.048**
			Total por 4 meses	**$8.552.192**

Tabla 4: Costos de RRHH.

3.1.3. Costos Totales

Elemento	Costo
Hardware	$1.222.550
Software	$16.960
Recursos Humanos	$8.552.192
Costo total del proyecto	**$9.791.702**

Tabla 5: Costos Totales.

3.2. Distribución de Tiempos

3.2.1. Carta Gantt

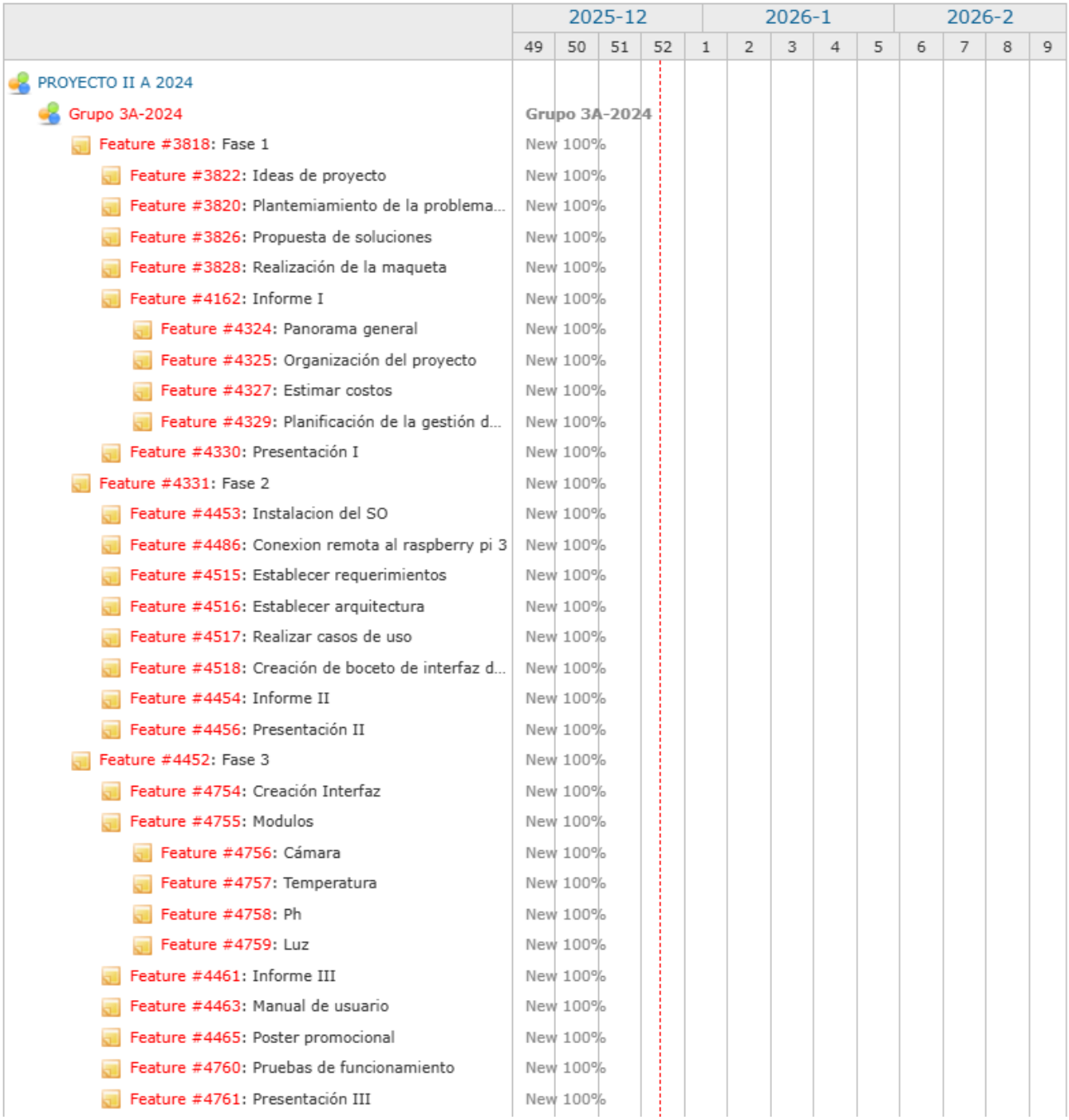


Ilustración 1: Carta Gantt

3.2.2. Asignación de Tiempo

FASE	Semanas comprendidas	Fecha estimada
1	Desde la 1º hasta la 6º semana	Del 9 de Septiembre al 15 de Octubre
2	Desde la 7º hasta la 12º semana	Del 21 de Octubre al 26 de Noviembre
3	Desde la 13º hasta la 16º semana	Del 2 de Diciembre al 23 de Diciembre

Tabla 6: Fases del Proyecto

3.3. Planificación de la gestión de riesgos

Se elabora la siguiente tabla para establecer las acciones a tomar ante los posibles riesgos, según su nivel de impacto. Los niveles de impacto son:

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

Riesgo	Nivel de impacto	Probabilidad de ocurrencia	Acción remedial
1.- Mala distribución de tareas en los integrantes del equipo.	3	30%	Realizar reuniones periódicas para revisar la asignación de tareas y equilibrar la carga de trabajo entre los integrantes.
2.- Falta de conocimiento necesario de los integrantes en las herramientas a utilizar.	2	80%	Dedicar horas externas al proyecto en estudiar y adquirir conocimientos de las herramientas a utilizar.
3.- Problema por disponibilidad de tiempo o enfermedad de algún integrante del equipo.	2	50%	Reasignación de la carga de tareas dentro de los integrantes con el fin de continuar trabajando sin mayores inconvenientes.
4.- Mala comunicación entre integrantes.	3	50%	Fomentar un ambiente colaborativo y mantener una comunicación constante mediante los canales del equipo.
5.- Desacuerdos del equipo durante la realización del proyecto.	2	30%	Conversaciones entre los integrantes para llegar a acuerdos sin afectar la velocidad en la que se desarrolla el proyecto.
6.- Daño del material de hardware utilizado.	1	20%	Reemplazo de una o más piezas según la gravedad del problema. Cualquier posible costo adicional es repartido entre los integrantes.
7.- Falta de materiales necesarios para el proyecto.	2	20%	Revisar los recursos requeridos antes de cada etapa del proyecto para asegurar su disponibilidad.

8.- Cancelación de sesiones de trabajo en clase por situaciones externas.	4	10%	Acuerdo entre los integrantes del equipo para hacer reuniones que recuperen el tiempo de trabajo contemplado.
9.- Mala administración del tiempo y/o retraso en los entregables del proyecto.	2	15 %	Planificar las actividades y dividir tareas en subtareas con fechas claras para asegurar el cumplimiento de los plazos.
10.- Pérdida de materiales o componentes de trabajo del proyecto.	3	60 %	Reposición del componente perdido realizada por los integrantes responsables. Pago total del costo en el caso de componentes prestados.
11.- Fallo en alguno de los componentes del proyecto.	1	20%	Reemplazo de una o más piezas según la gravedad del problema. Cualquier posible costo adicional es repartido entre los integrantes.
12.- Incompatibilidad entre componentes del hardware o con el software.	2	30 %	Comprobar las especificaciones de los componentes antes de su compra, conexión o implementación para evitar fallas.

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:

1. El sistema debe permitir al usuario programar horarios de alimentación a través de una aplicación móvil.
2. El sistema debe dispensar una cantidad de alimento predefinida por el usuario cuando se cumpla el horario programado o se active manualmente.
3. El sistema debe medir y registrar la cantidad de alimento dispensado y consumido (en gramos) utilizando el sensor de peso.
4. La aplicación móvil debe mostrar en tiempo real el estado de los depósitos (nivel de comida y agua) y el historial de consumo.
5. El sistema debe notificar al usuario cuando los niveles de comida o agua en los depósitos estén bajos.

Requerimientos No Funcionales:

1. La aplicación móvil debe ser intuitiva, permitiendo a un usuario configurar de manera rápida un horario de alimentación.
2. Minimizar los tiempos de respuesta en la conexión remota para un rápido envío de los parámetros del dispensador y envío de alertas.
3. El sistema debe mantener su funcionalidad automatizada básica (ejecutar horarios) aunque se pierda la conexión a la red.
4. Permitir la integración de sensores adicionales o accionadores para adaptarse a futuros requerimientos o mejoras del sistema.

4.1.2. Descripción de la Arquitectura

La arquitectura del proyecto se basa en la estructura Cliente-Servidor y se representa de la siguiente manera:

Cliente: El usuario utilizará la aplicación que permitirá el monitoreo y control del sistema. A través de esta aplicación el usuario podrá recibir alertas en tiempo real sobre el estado de los parámetros críticos del entorno.

Servidor: El servidor está implementado en una Raspberry Pi, sobre ella se ejecuta un software que actúa como intermediario entre el hardware (sensores y actuadores) y el usuario.

Hardware: El hardware está compuesto por la Raspberry Pi 4, que actúa como el controlador central y se interconecta con los siguientes componentes: sensor de ultrasonido, sensor de cámara, sensor de peso, tornillo sin fin, válvula solenoide.

Además, tanto el cliente como el servidor deben estar conectados a la misma red Wi-Fi, lo que facilita la comunicación y transferencia de datos en tiempo real entre ambos.

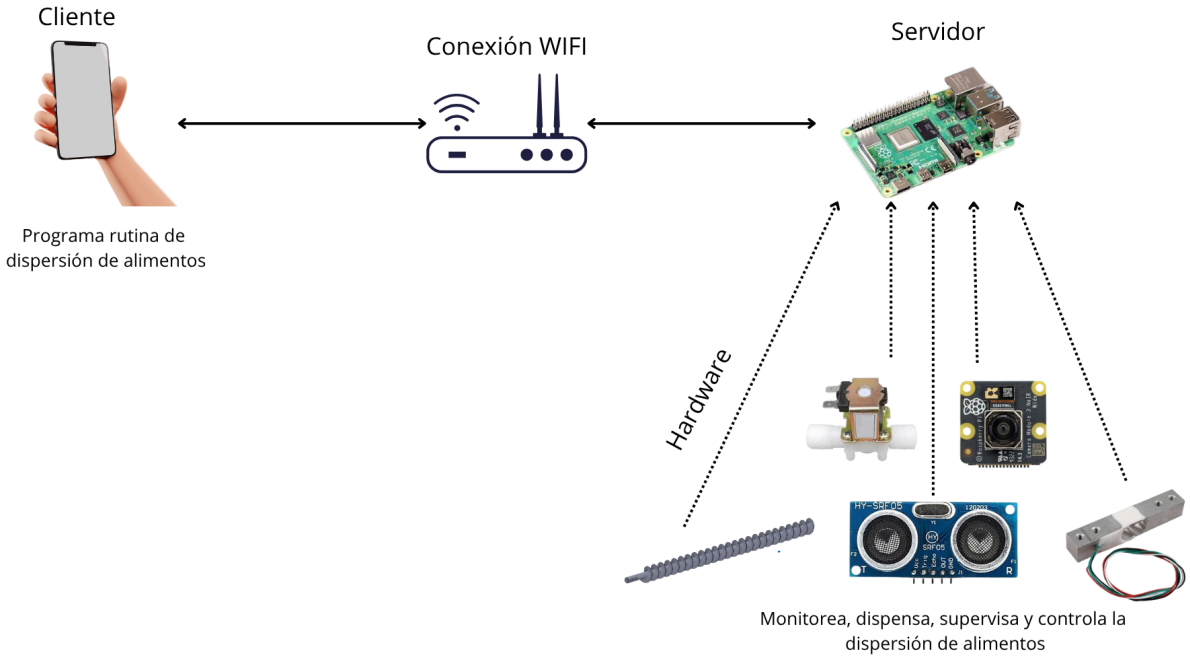


Ilustración 2: Descripción de Arquitectura

4.1.3. Diagrama de Clase

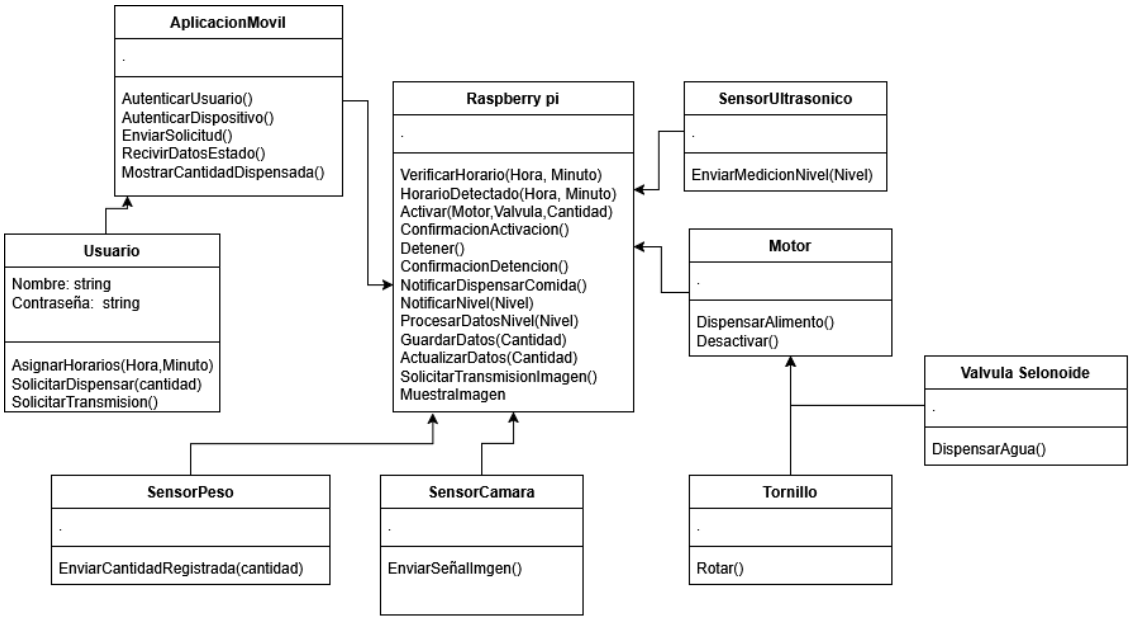


Ilustración 3: Diagrama de Clase

4.1.4. Diseño de Interfaz de Usuario

Pantalla inicial de la app del dispensador *Smart Feed*.
Permite al usuario configurar la conexión del dispositivo por primera vez. Incluye dos secciones: una para ingresar los datos de la red WiFi y otra para registrar la información del dispensador

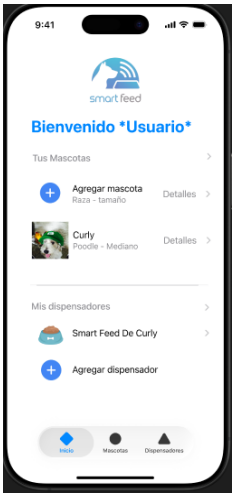
Ilustración 4: Pantalla Inicial

Pantalla inicio de sesión de la aplicación *Smart Feed*.
Muestra campos de usuario y contraseña que previamente están registrados de lo contrario se incluye un enlace para que los usuarios se registren.

Ilustración 5: Pantalla de Inicio

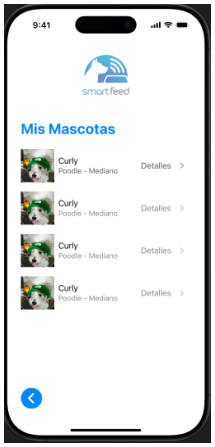
Pantalla registro de la app *Smart Feed*.
Presenta un formulario donde se ingresa nombre completo, nombre de usuario, contraseña y correo electrónico mediante campos de texto.

Ilustración 6: Pantalla Registro



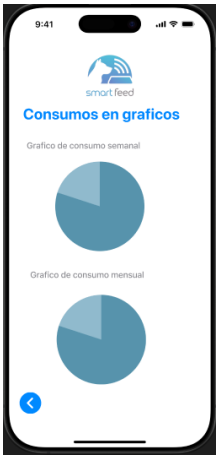
Pantalla principal de *Smart Feed*.
Se organiza en dos secciones: **Tus Mascotas**, donde se pueden ver o agregar mascotas, y **Mis dispensadores**, que muestra los dispositivos vinculados y permite añadir uno nuevo. En la parte inferior incluye un **menú de navegación** para acceder rápidamente a Inicio, Mascotas y Dispensadores.

Ilustración 7: Pantalla Principal logueado



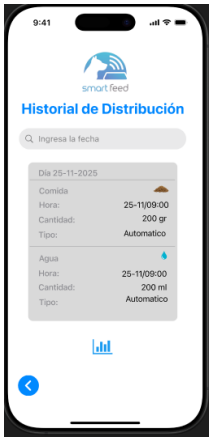
Pantalla “Mis Mascotas” dentro de la app *Smart Feed*.
Despliega una lista de mascotas registradas por el usuario, donde cada elemento incluye la foto del animal, su nombre, raza y tamaño, junto con un botón de Detalles para acceder a información más específica.

Ilustración 8: Pantalla Mis Mascotas



Pantalla “Consumos en gráficos” de la app *Smart Feed*.
La interfaz presenta dos gráficos circulares: uno que representa el consumo semanal y otro que muestra el consumo mensual de alimentos. Cada gráfico está acompañado por su respectiva etiqueta descriptiva.

Ilustración 9: Pantalla Consumos en Gráficos

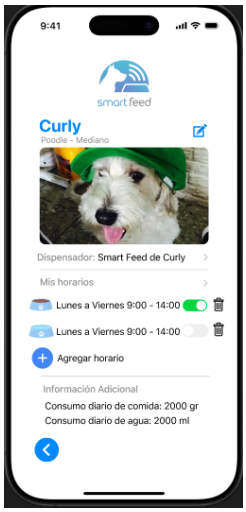


Pantalla Historial de Distribución de la app *Smart Feed*.
La interfaz presenta una tarjeta con las distribuciones realizadas en el día seleccionado, mostrando tanto las de comida como las de agua, incluyendo información como hora, cantidad dispensada y tipo de activación.

Ilustración 10: Pantalla Historial

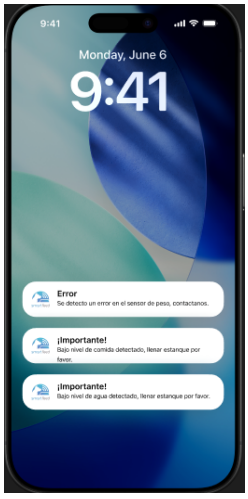
Pantalla vista principal del dispensador de alimentos.
Presenta el nombre del dispositivo, la mascota asociada y los niveles actuales de comida y agua. También muestra la próxima y la última dispensación registrada. En la parte inferior incluye controles para dispensar manualmente comida o agua y accesos rápidos a la cámara, el historial y la configuración del dispositivo.

Ilustración 11: Pantalla Vista Principal



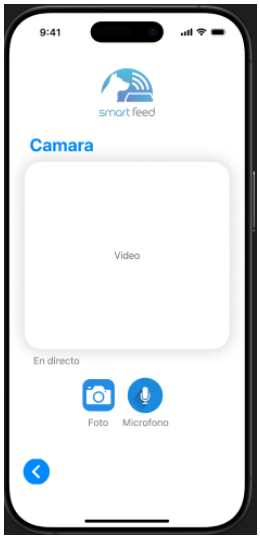
Esta pantalla muestra el perfil de la mascota, incluyendo su foto, nombre, raza y tamaño, junto con el dispensador asociado. Permite gestionar los horarios de alimentación (activar, desactivar, eliminar o agregar nuevos). Al final, se muestra la información de consumo diario recomendado, y se incluye un botón para volver a la pantalla anterior.

Ilustración 12: Pantalla Perfil Mascota



Pantalla ejemplo notificaciones del sistema enviadas por la app *Smart Feed* en la pantalla de bloqueo del teléfono.
Cada notificación incluye el logotipo de la aplicación, un título (como “Error” o “¡Importante!”) y un mensaje descriptivo que informa al usuario sobre situaciones relevantes del dispensador, como fallos, niveles bajos de comida o agua, o alertas que requieren atención inmediata.

Ilustración 13: Pantalla Notificaciones



Pantalla vista de cámara del dispensador.
Interfaz muestra con la transmisión de vídeo en directo al centro. Incluye botones para tomar una foto y activar o desactivar el micrófono.

Ilustración 14: Pantalla Vista de Cámara

Pantalla permite agregar un nuevo horario de alimentación.
Incluye un formulario con campos para nombre, tipo de alimento, cantidad, fecha y hora, además de un interruptor para activarlo o desactivarlo.

Ilustración 15: Pantalla Agregar Horario

4.1.5. Diagrama de contexto

El presente diagrama de contexto describe las interfaces principales del sistema "Smart Feeder" con su entorno. Se especifican las interacciones entre los componentes internos del dispensador (sensores y actuadores) y cómo estos se relacionan con las entidades externas.

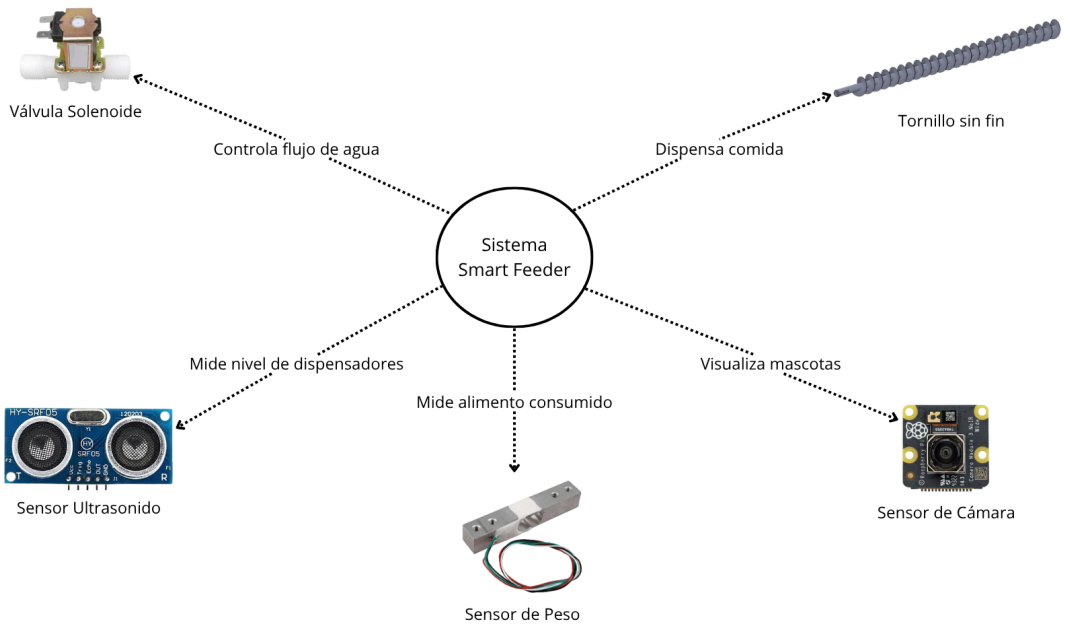


Ilustración 16: Diagrama de contexto

4.1.6. Casos de Uso General

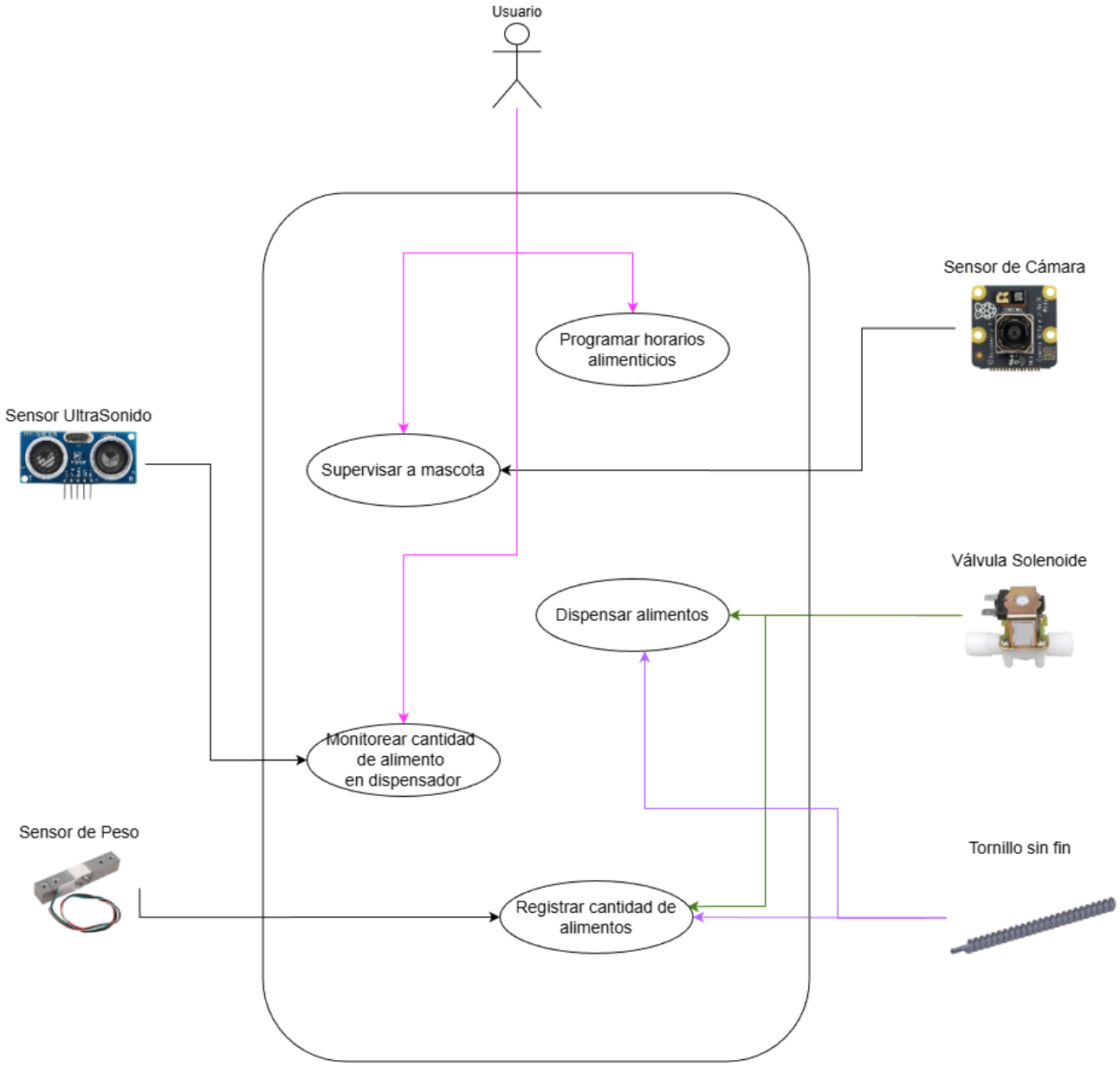


Ilustración 17: Casos de uso general.

4.1.7. Casos de Uso

<table>
<tr><td colspan="2">Nombre Caso de Uso: Monitorear cantidad alimento en dispensador</td></tr>
<tr><td colspan="2">Autor/Fecha: Yazuska Castillo / 24-11-2025</td></tr>
<tr><td colspan="2">Descripción: El Sensor Ultrasónico envía datos a la Raspberry Pi, la cual determina la cantidad de alimento disponible en el plato y en el depósito.</td></tr>
<tr><td colspan="2">Actores: Sensor Ultrasónico</td></tr>
<tr><td colspan="2">Precondición:
1- El dispensador debe estar encendido y conectado.

2- Los sensores deben estar funcionando correctamente.

3- El sistema debe estar sincronizado con la aplicación móvil.</td></tr>
<tr><td>Flujo Principal: Sensor

1. El Sensor Ultrasónico envía a la Raspberry Pi la medición del nivel del depósito de alimentos.</td><td>Flujo principal: Sistema

2. La Raspberry Pi recibe los datos de ambos sensores.

3. Procesa los datos para determinar la cantidad de alimento disponible.

4. Se le notifica al usuario cuál es el nivel correspondiente del dispensador.</td></tr>
<tr><td colspan="2">Postcondiciones: Se notifica en la aplicación móvil.</td></tr>
</table>

Tabla 8: Monitorear cantidad alimento

<table>
<tr><td colspan="2">Nombre Caso de Uso: Dispensar alimento</td></tr>
<tr><td colspan="2">Autor/Fecha: Claudio Carvajal / 24-11-2025</td></tr>
<tr><td colspan="2">Descripción: El dispensador entrega alimento o agua a la mascota. Esto puede ocurrir de dos formas:
1. Manual, cuando el usuario lo solicita desde la aplicación móvil.
2. Automática, cuando la Raspberry Pi ejecuta un horario previamente programado en el sistema.
En ambos casos, la Raspberry controla el motor de tornillo sin fin y la válvula solenoide para liberar la cantidad definida.</td></tr>
<tr><td colspan="2">Actores: Válvula solenoide, Tornillo sin fin.</td></tr>
<tr><td colspan="2">Precondición:
1- El plato debe estar vacío.

2- El dispensador debe tener alimento disponible.

3- El motor debe estar operativo.

4- Para el flujo manual: la app debe estar conectada al sistema.

5- Para el flujo automático: debe existir un horario programado.</td></tr>
<tr><td>Flujo Principal: Motores

3. El tornillo sin fin del dispensador de comida en relación a sus giros dispensa comida. La válvula solenoide con respecto a segundos dispensa agua.

5. Una vez dispensados los alimentos, los sensores son detenidos.</td><td>Flujo principal: Sistema (Manual)

1. El sistema recibe orden de dispensar alimentos.

2. Activa el motor de tornillo sin fin y válvula solenoide para liberar la cantidad indicada.

4. Detiene el motor cuando se alcanza la cantidad establecida.

6. Envía una notificación de que dispensó alimento a la mascota.</td></tr>
<tr><td>Flujo alternativo: Motores</td><td>Flujo alternativo: Sistema (Automático)

1. Raspberry Pi detecta que se alcanzó un horario programado.

2. Activa el motor de tornillo sin fin y válvula solenoide para liberar la cantidad indicada.</td></tr>
</table>

3. El tornillo sin fin del dispensador de comida en relación a sus giros dispensa comida. La válvula solenoide con respecto a segundos dispensa agua. 5. Una vez dispensados los alimentos, los sensores son detenidos.	4. Detiene el motor cuando se alcanza la cantidad establecida. 6. Envía una notificación de que dispensó alimento a la mascota.
Postcondiciones: El alimento es dispensado correctamente y gracias a otro sensor la cantidad de alimento queda registrada en el sistema.	

Tabla 9: Dispensar alimento

<table>
<tr><td colspan="2">Nombre Caso de Uso: Registrar cantidad de alimento</td></tr>
<tr><td colspan="2">Autor/Fecha: René Ayca / 24-11-2025</td></tr>
<tr><td colspan="2">Descripción: El sistema registra la cantidad de alimento presente en el plato mediante los datos enviados por el sensor de peso. Esta información queda almacenada para estadísticas, control y supervisión del cliente.</td></tr>
<tr><td colspan="2">Actores: Sensor de peso, Válvula solenoide, Tornillo sin fin.</td></tr>
<tr><td colspan="2">Precondición:
1- El sensor de peso debe estar funcionando y conectado.

2- La Raspberry Pi debe estar encendida y comunicándose con el sistema.</td></tr>
<tr><td>Flujo Principal: Sensor

1. El sensor de peso registra cantidades dispensadas en el plato de la mascota.</td><td>Flujo principal: Sistema

2. El sistema recibe las cantidades de alimentos dispensadas y las guarda en la base de datos del sistema.

3. El sistema en la aplicación móvil muestra mediante tablas la cantidad registrada de alimentos.</td></tr>
<tr><td colspan="2">Postcondiciones: La cantidad de alimento queda almacenada correctamente en el sistema para su consulta futura, los datos pueden utilizarse para historial de consumo o análisis nutricional.</td></tr>
</table>

Tabla 10: Registrar cantidad de alimento

<table>
<tr><td colspan="2">Nombre Caso de Uso: Supervisar a la mascota</td></tr>
<tr><td colspan="2">Autor/Fecha: Israel Tebes / 24-11-2025</td></tr>
<tr><td colspan="2">Descripción: El cliente utiliza la aplicación móvil para ver en tiempo real lo que capta la cámara del dispensador, con el fin de supervisar la actividad de su mascota.</td></tr>
<tr><td colspan="2">Actores: Cliente.</td></tr>
<tr><td colspan="2">Precondición:
1- La cámara debe estar en funcionamiento y conectada al sistema.

2- La Raspberry Pi debe estar encendida y comunicándose con la aplicación.

3- El cliente debe tener acceso a la aplicación móvil.</td></tr>
<tr><td>Flujo Principal: Cliente

1. El cliente con la aplicación solicita al sistema la transmisión de la cámara.

8. El cliente visualiza la imagen que aparezca en pantalla.</td><td>Flujo principal: Sistema

2. El sistema recibe la solicitud del cliente.

3. La Raspberry Pi solicita al sensor de cámara la captura o transmisión de la imagen.

4. La cámara envía la señal de video o imagen al sistema.

5. El sistema procesa la información recibida.

6. Envía la visualización en tiempo real a la aplicación móvil.

7. La aplicación muestra la vista al cliente.</td></tr>
<tr><td colspan="2">Postcondiciones: La imagen o video de la mascota es visualizado por el cliente.</td></tr>
</table>

Tabla 11: Supervisar a la mascota

<table>
<tr><td colspan="2">Nombre Caso de Uso: Programar horarios alimenticios</td></tr>
<tr><td colspan="2">Autor/Fecha: René Ayca / 24-11-2025</td></tr>
<tr><td colspan="2">Descripción: El cliente configura un horario específico para que el dispensador libere alimento de manera automática.</td></tr>
<tr><td colspan="2">Actores: Cliente.</td></tr>
<tr><td colspan="2">Precondición:
1- El sistema debe estar en funcionamiento y conectado a la Raspberry Pi.

2- El cliente debe estar autenticado en la aplicación.</td></tr>
<tr><td>Flujo Principal: Usuario

1. El usuario ingresa la hora deseada y la cantidad de alimento a dispensar.

2. Confirma la programación y la aplicación envía los parámetros al sistema.</td><td>Flujo principal: Sistema

3. El sistema recibe la programación enviada por el cliente.

4. Valida formato, horario y cantidad ingresada.

5. Registra el horario en la base de datos.

6. Envía la nueva programación al Raspberry Pi.

7. La Raspberry Pi almacena y activa el nuevo horario automático.

8. El sistema envía una confirmación a la aplicación móvil.</td></tr>
<tr><td colspan="2">Postcondiciones: El horario alimenticio queda registrado en el sistema y en la Raspberry Pi, la dispensación automática se ejecutará en la hora programada, el cliente recibe confirmación exitosa del registro.</td></tr>
</table>

Tabla 12: Programar horarios alimenticios

4.1.8. Diagrama de Actividad o Secuencia

Monitorear cantidad alimento en dispensador

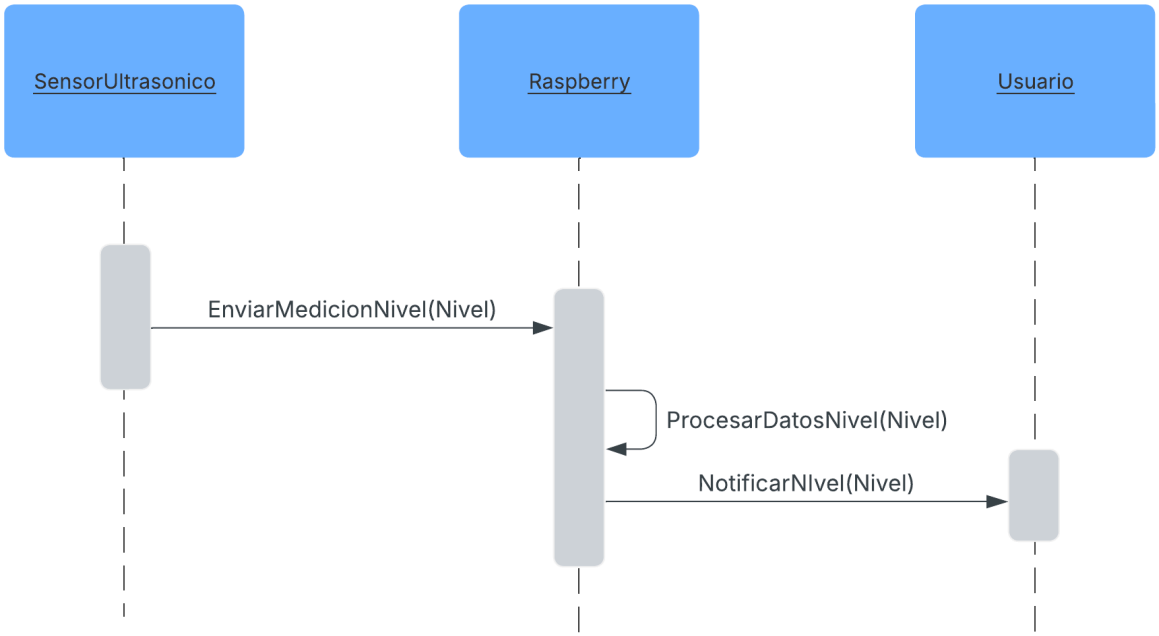


Ilustración 18: Diagrama de Secuencia Monitorear Alimento en Dispensador

Registrar cantidad de alimento

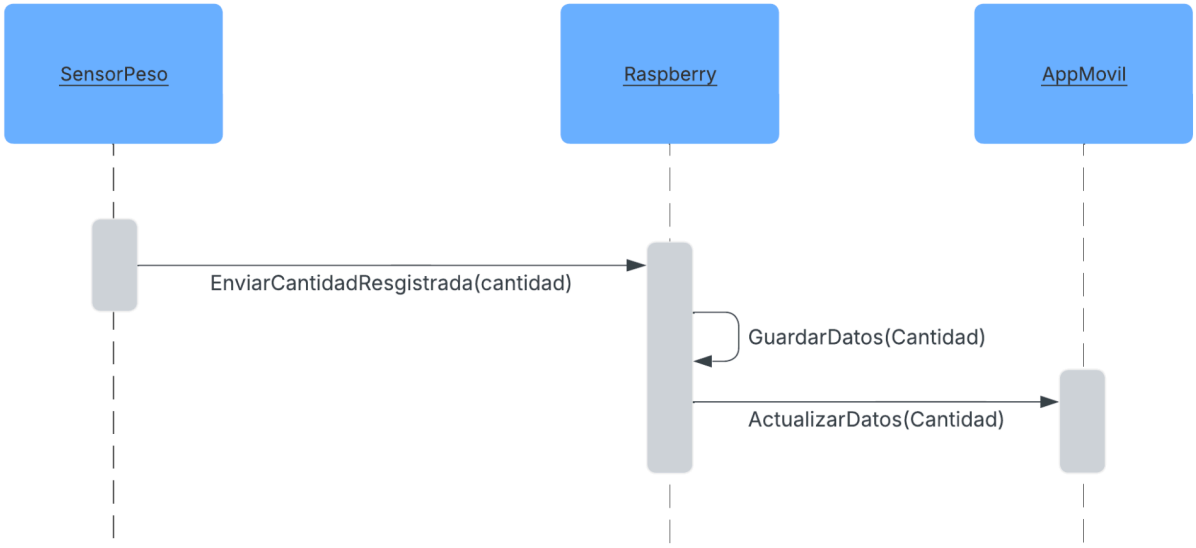


Ilustración 19: Diagrama de Secuencia Registrar Cantidad de Alimento

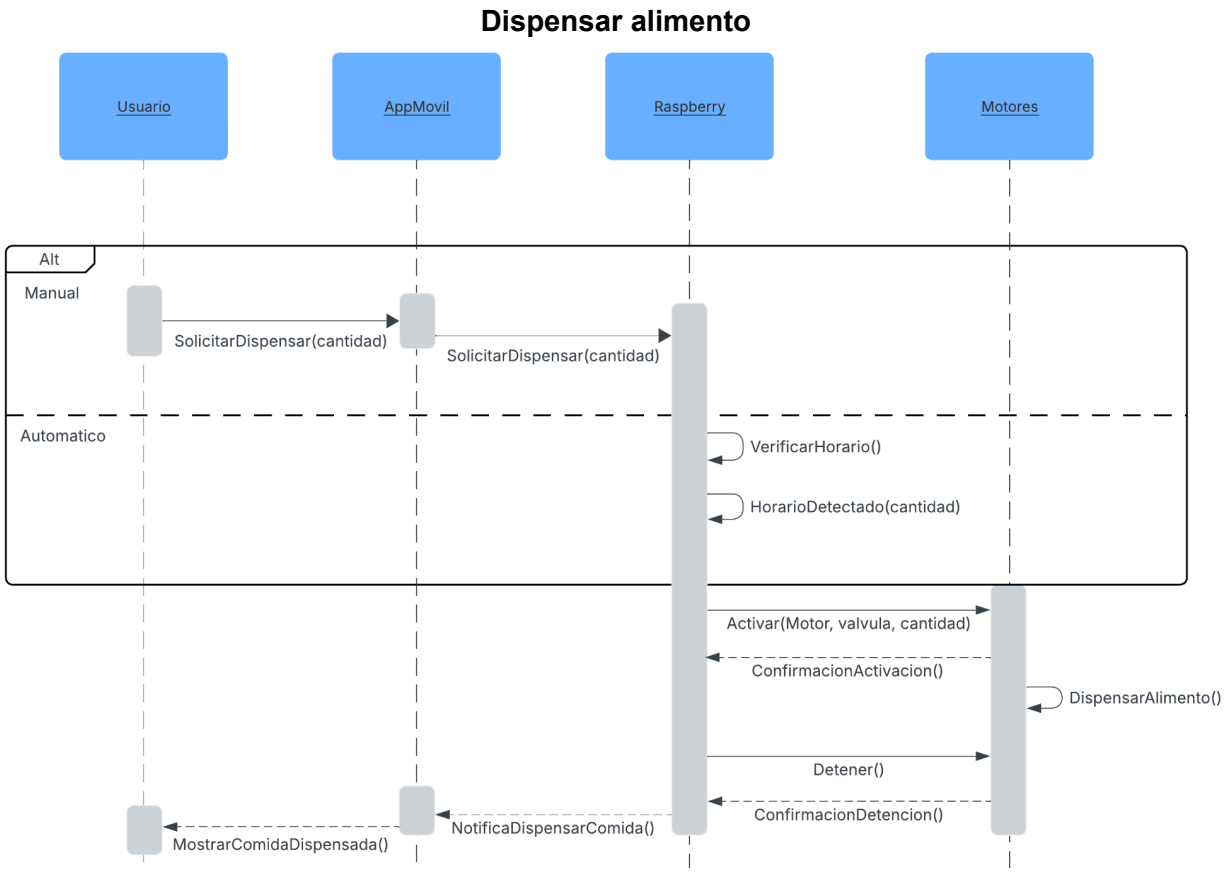


Ilustración 20: Diagrama de Secuencia Dispensar Alimento

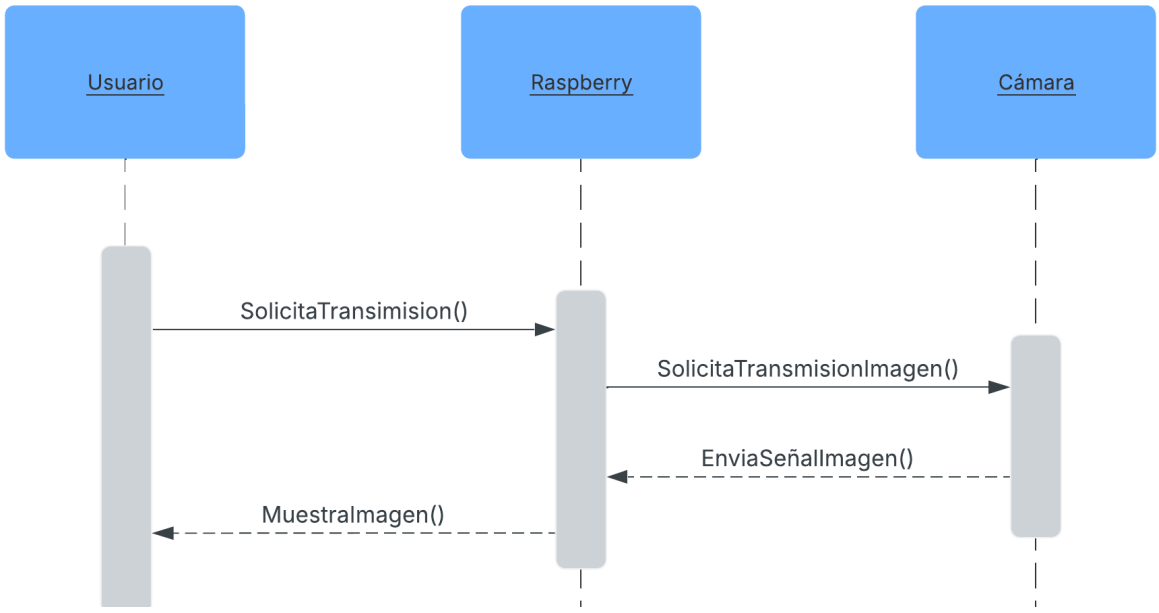


Ilustración 21: Diagrama de Secuencia Supervisar a la Mascota

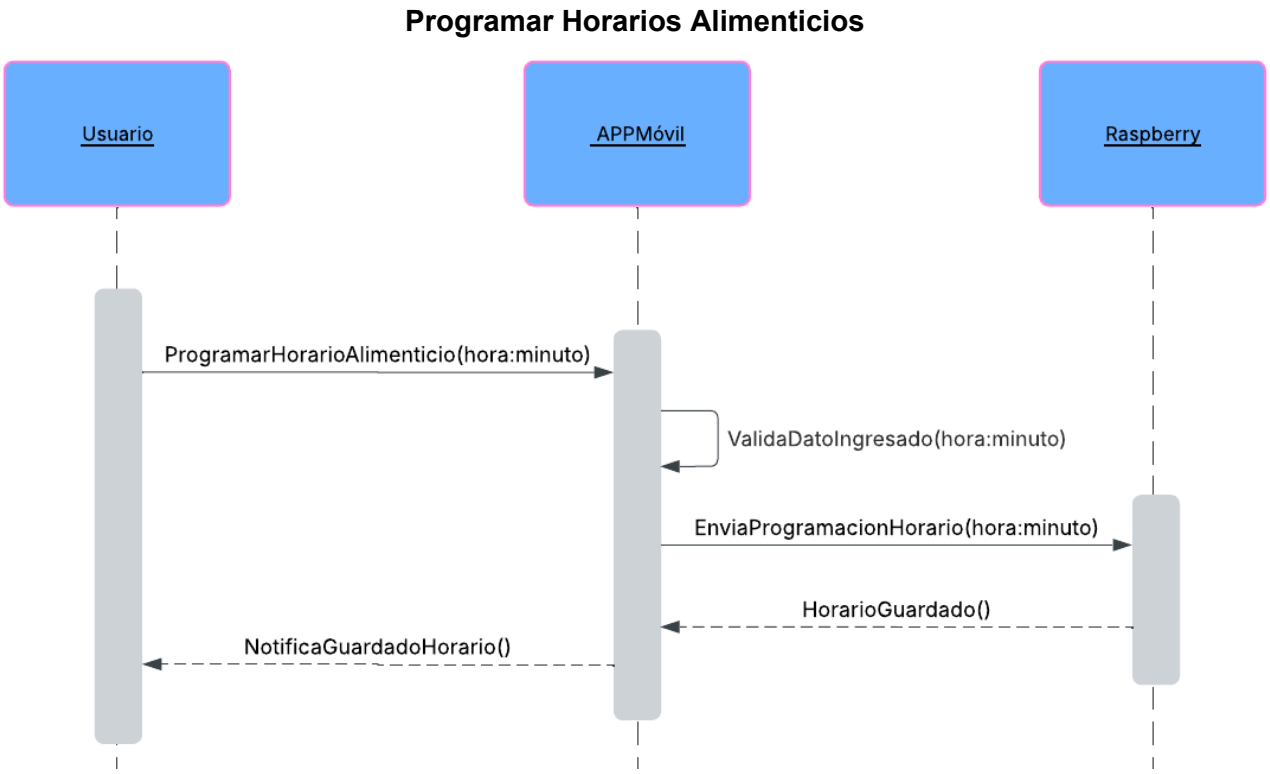


Ilustración 22: Diagrama de Secuencia Programar Horarios Alimenticios

4.2. Herramientas y Técnicas

Para llevar a cabo el desarrollo y ejecución del proyecto, será necesario contar con distintas herramientas. Las herramientas necesarias para la implementación son las siguientes:

- **Visual Studio Code**: Un entorno de desarrollo integrado (IDE) que facilita la escritura, depuración y administración de código. Utilizaremos Visual Studio Code para desarrollar y editar el código Python y para gestionar el proyecto en general.

- **Python**: Un lenguaje de programación versátil y de alto nivel. Se utilizará Python como el lenguaje principal para desarrollar la lógica de la aplicación.

Para obtener los datos de los sensores se utilizará la librería grovepi, la cual es una librería de Python que permite obtener entradas de los dispositivos de entrada y salida de propósito general.

5. Implementación

5.1. Plan de integración

5.1.1. Integración de Hardware y Software

En esta etapa se realizó la integración de los componentes físicos con el sistema lógico del proyecto. Se implementaron sensores de ultrasonido para medir el nivel de comida y agua en sus respectivos dispensadores, mientras que los actuadores se encargan de la dispensación controlada de ambos recursos.

La Raspberry Pi, junto con el módulo GrovePi, gestiona el funcionamiento general del sistema, actuando como el cerebro de la solución. Esta se comunica con el software desarrollado en Python, el cual es responsable de procesar los datos provenientes de los sensores y ejecutar las acciones correspondientes sobre los actuadores.

5.1.2. Integración de la Interfaz de Usuario

Posteriormente, se integró la aplicación móvil con el software del componente físico del sistema. Para ello, se desarrolló una aplicación que permite a los usuarios ver el estado de los dispensadores, configurar los tiempos de dispensación y activar manualmente los actuadores.

Durante esta etapa, también se realizaron pruebas de conectividad con el objetivo de asegurar una comunicación estable y confiable entre la aplicación móvil y la Raspberry Pi.

5.1.3. Pruebas del sistema

Finalmente, se realizaron pruebas con el sistema completamente operativo, simulando la dispensación de comida y agua. Durante estas pruebas se verificó que las órdenes enviadas a través de la aplicación móvil se ejecutarán sin problemas, así como también que la información entregada al usuario refleja de forma precisa el estado real de los dispensadores.

5.2. Modelo de implementación

El modelo de implementación del sistema se estructuró en tres fases principales, las cuales permitieron una integración progresiva y ordenada del hardware, el software y la interfaz de usuario:

5.2.1. Fase 1: Instalación Física

En esta fase se realizó el diseño de la maqueta y la definición de la ubicación óptima de los sensores y actuadores, considerando el correcto funcionamiento del sistema. Posteriormente, se llevó a cabo la conexión de los actuadores utilizando un protoboard, asegurando una correcta distribución eléctrica. Finalmente, se realizó el montaje completo del hardware junto a la maqueta, integrando la Raspberry Pi, los sensores y los actuadores**.**

5.2.2. Fase 2: Configuración Inicial

Durante esta etapa se desarrolló y configuró el software encargado del funcionamiento de los sensores y actuadores. Se programaron las funciones para la lectura de datos y la ejecución de acciones automáticas. Además, se realizó el ajuste de parámetros relacionados con la dispensación automática de comida y agua, tales como tiempos de activación y niveles mínimos definidos por el usuario a través de la aplicación móvil.

5.2.3. Fase 3: Pruebas Funcionales

En la última fase se simularon escenarios de uso mediante la aplicación móvil, relacionados con la alimentación y suministro de agua, con el objetivo de evaluar la respuesta del sistema. así como la coherencia entre la información mostrada en la aplicación y el estado real de los dispensadores.
Adicionalmente, se realizó la capacitación del usuario final, proporcionando un manual de usuario detallado.

5.3. Módulos implementados

5.3.1. Interfaz de Usuario

Este bloque de código se encarga de la configuración del servidor para permitir la comunicación con la aplicación móvil. En él se instancia la aplicación Flask y se definen los encabezados HTTP necesarios para habilitar el acceso desde clientes externos, permitiendo solicitudes provenientes de distintos orígenes.

```python
app = Flask(__name__)

@app.after_request
def after_request(response):
    response.headers.add('Access-Control-Allow-Origin', '*')
    response.headers.add('Access-Control-Allow-Headers', 'Content-Type')
    response.headers.add('Access-Control-Allow-Methods', 'GET,POST,OPTIONS')
    return response
```

Ilustración 23: Código Interfaz de usuario 1

Este bloque de código define los parámetros de conexión utilizados por la aplicación móvil para comunicarse con la Raspberry Pi. Se establecen constantes que contienen la dirección IP y el puerto del servidor.

```javascript
const API = "http://192.168.1.194:5000";
const IP_RASPBERRY = "192.168.1.194";
```

Ilustración 24: Código Interfaz de usuario 2

5.3.2. Controlador Central

Este módulo corresponde al servidor principal del sistema Smart Feed, encargado de iniciar y coordinar todos los componentes del dispensador inteligente. En él se inicializa el servidor Flask, se habilita la ejecución concurrente mediante hilos para la lectura continua de sensores y la planificación automática de horarios.

```python
if __name__ == "__main__":
    # Iniciar sensores en hilos
    threading.Thread(target=leer_sensores, daemon=True).start()
    threading.Thread(target=scheduler, daemon=True).start()

    print("=" * 60)
    print("           DISPENSADOR IoT - FOTO EN MEMORIA")
    print("=" * 60)
    print("\n SENSORES INICIADOS:")
    print(f"  Comida: D{ULTRA_COMIDA}")
    print(f"  Agua: D{ULTRA_AGUA}")
    print(f"  Relay agua: D{RELAY_AGUA}")

    print("\n ENDPOINTS CÁMARA:")
    print("  GET   /video           - Stream MJPEG")
    print("  GET   /foto            - Foto en tiempo real (memoria)")
    print("  POST  /foto-guardar    - Tomar y guardar foto en disco")

    print("=" * 60)
    print(f"\n Servidor iniciado en: http://0.0.0.0:5000")
    print(" Video en vivo: http://<TU_IP>:5000/video")
    print(" Foto instantánea: http://<TU_IP>:5000/foto")
    print("=" * 60)

    try:
        app.run(host="0.0.0.0", port=5000, threaded=True, debug=False)
    except KeyboardInterrupt:
        print("\n Cerrando servidor...")
        release_camera()
    finally:
        release_camera()
```

Ilustración 25: Controlador central 1

5.3.3. Sensores y Actuadores

Este código en Python se encarga de leer periódicamente los valores generados por los sensores de nivel de comida y agua conectados a la Raspberry Pi a través de la placa GrovePi. La función realiza lecturas continuas de los sensores ultrasónicos, calcula el nivel y el porcentaje disponible de cada contenedor.

```python
def leer_sensores():
    global estado
    while True:
        try:
            distancia = grovepi.ultrasonicRead(ULTRA_COMIDA)
            nivel, porcentaje = calcular_nivel(distancia, "comida")
            estado["comida"] = {
                "distancia_cm": distancia,
                "nivel": nivel,
                "porcentaje": round(porcentaje, 1)
            }
        except Exception as e:
            print(f"Error sensor comida: {e}")
            estado["comida"]["nivel"] = "error"

        try:
            distancia = grovepi.ultrasonicRead(ULTRA_AGUA)
            nivel, porcentaje = calcular_nivel(distancia, "agua")
            estado["agua"] = {
                "distancia_cm": distancia,
                "nivel": nivel,
                "porcentaje": round(porcentaje, 1)
            }
        except Exception as e:
            print(f"Error sensor agua: {e}")
            estado["agua"]["nivel"] = "error"

        time.sleep(2)
```

Ilustración 26: Sensores y Actuadores 1

Este código se encarga de activar el relé correspondiente al sistema de agua del dispensador durante un tiempo determinado. La función recibe como parámetros el tipo de recurso a dispensar y la duración de la activación, permitiendo controlar el encendido y apagado del relé de forma temporizada.

```
def activar_relay(tipo, segundos):
    if tipo == "agua":
        grovepi.digitalWrite(RELAY_AGUA, 1)
        time.sleep(segundos)
        grovepi.digitalWrite(RELAY_AGUA, 0)
```

Ilustración 27: Sensores y Actuadores 2

Este código implementa un sistema de planificación automática para el dispensador automático. La función verifica de forma periódica la fecha y hora actual, compara estos valores con la configuración de horarios definida por el usuario.

```
def scheduler():
    while True:
        ahora = datetime.datetime.now()
        hora = ahora.strftime("%H:%M")
        dia = ahora.strftime("%A")

        config = cargar_config()

        for h in config["horarios"]:
            if not h["activo"]:
                continue
            if h["hora"] != hora:
                continue
            if dia not in h["dias"]:
                continue

            activar_relay(h["tipo"], h["tiempo"])

        time.sleep(60)
```

Ilustración 28: Sensores y Actuadores 3

6. Problemas encontrados y soluciones propuestas

6.1. Regulador de voltaje

Se tenía conectado una fuente de alimentación de 12 volts, para alimentar un motor de solo 3 volts y una válvula solenoide de 12 volts, el problema es que al regular el voltaje, no era capaz de disipar el calor, lo que hacía que del regulador de voltaje salga humo.

Solución:
Se optó como decisión del equipo y sugerencia del profesor, reemplazar la fuente de alimentación de 12 volts al motor y dedicarle una fuente de voltaje especializado para el motor de 3 volts, es decir, solo para el motor de 3 volts se alimentó con pilas y se dejó el motor de 12 volts para la válvula solenoide.

6.2. Tornillo sin fin

Una vez puesto el tornillo sin fin en la maqueta, el motor de 3 volts no tenia la suficiente fuerza para mover el tornillo sin fin con comida.

Solución:
Al ser un problema casi directamente conectado con el problema anterior, se optó por mejor reemplazar el motor de 3 volts, por uno de 12 volts y con esto se aprovechaba solo una fuente para alimentar un motor y una válvula, las 3 conectadas al protoboard.

6.3. Sensor de peso

Con el sensor de peso hubieron bastantes problemas, casi desde el inicio de la asignatura. En el departamento de Informática no se tenían sensores de peso, se fue a buscar sensores al edificio de electricidad y al igual que en el de Informática, no se tenían sensores de peso.

Solución:
Como gasto de equipo se compraron 2 sensores de peso, con sus respectivas placas hx711.

Y el problema más grave que tuvimos fue la conexión de los cables a la placa hx711, la calibración de peso para el sensor y finalmente el posible deterioro de la placa hx711, lo cual se hizo imposible encontrar una solución acorde al tiempo que se estipulaba para el sensor de peso.

7. Trabajo futuro

El proyecto Smart Feeder sin dudar ha logrado un progreso importante en su desarrollo, sin embargo, aún hay diversas áreas que pueden ser mejoradas y ampliadas para optimizar su rendimiento y ampliar su alcance y funcionalidad.

7.1. Sensores de peso

Con la experiencia recolectada en el proceso de hacer funcionar el sensor de peso, se considera como trabajo futuro la correcta implementación de ambos sensores de peso, para que se puede monitorear los alimentos dispensados y tener un registro exacto de la alimentación de la mascota.

7.2. Tamaño personalizado

El prototipo de Smart Feeder fue confeccionado para el tamaño de un perro de raza pequeña, por lo tanto, se propone que una segunda idea para la escalabilidad sea confeccionar dispensadores de tamaños personalizados, lo cual pueda cumplir con los requerimientos y expectativas de los clientes.

7.3. Multi-Dispensadores

En fases anteriores a esta última se logró tomar como sugerencia tomar a un solo usuario que sea capaz de controlar una múltiple cantidad de dispensadores a la vez. Usando tan solo un Raspberry como controlador central para controlar al resto de dispensadores.

8. Conclusión

La implementación de un sistema IoT para la automatización del cuidado de mascotas representa una solución innovadora para un problema cotidiano creciente. A lo largo de esta última fase se logró demostrar como fue el análisis, desarrollo e implementación del proyecto.

A lo largo de este informe, se detalló cómo se concretó la planificación, el desarrollo y análisis de fases anteriores. Se muestra el diseño de la maqueta, el funcionamiento y la conexión con los códigos de la Raspberry, la Aplicación, los actuadores y sensores.

En resumen, esta última fase ha permitido entender el funcionamiento del Smart Feeder, además plasmar todo lo que se propuso en un inicio de la asignatura. Los cimientos de un proyecto claro, conciso y bien orientado ya fueron establecidos, por lo tanto se logró una implementación correcta y clara de las ideas iniciales. El empleo de un modelo de proceso estructurado y la selección de herramientas y técnicas apropiadas han asegurado que el diseño no solo sea conceptualmente sólido, sino también práctico para un posible trabajo futuro.

9. Referencias

[1] Talent.com (n.d). Salario de Jefe de proyecto en Chile. Talent.com. https://cl.talent.com/salary?job=jefe+de+proyecto

[2] Talent.com (n.d). Salario de Programador en Chile. Talent.com. https://cl.talent.com/salary?job=Programador

[3] Talent.com (n.d). Salario de Diseñador en Chile. Talent.com. https://cl.talent.com/salary?job=diseñador

[4] Talent.com (n.d). Salario de Documentador en Chile. Talent.com. https://cl.talent.com/salary?job=Documentador

[5] Talent.com (n.d). Salario de Analista en Chile. Talent.com. https://cl.talent.com/salary?job=Analista

[6] Raspberry Pi 4 Modelo B / 8GB RAM https://raspberrypi.cl/producto/raspberry-pi-4-modelo-b-8gb-ram

[7] GrovePi+ Starter Kit - Dexter Industries https://www.dexterindustries.com/store/grovepi-starter-kit/

[8] Conexión GPIO de Raspberry Pi 3 | Electrónica y Ciencia https://www.electronicayciencia.com/2016/11/conexion-gpio-de-raspberry-pi-3.html

[9] Raspberry Pi y el IoT: guía para entender su papel en el Internet de las Cosas https://monraspberry.com/es/guia-de-raspberry-pi-iot/

[10] Seeed Studio (n.d). *Grove – ADC for Load Cell (HX711).* Seeed Studio. https://www.seeedstudio.com/Grove-ADC-for-Load-Cell-HX711-p-4361.html

[11] Dexter Industries (n.d). *GrovePi.* https://www.dexterindustries.com/grovepi/

[12] Made-in-China (n.d). *Stainless Steel Screw Flight Spiral Blade Helical Blade Shaftless for Drilling Machine.* https://es.made-in-china.com/co_seitotech/product_Stainless-Steel-Screw-Flight-Spiral-Blade-Helical-Blade-Shaftless-for-Drilling-Machine_yssisygiuy.html

[13] Dispensador inteligente para mascotas con IoT – monitoreo y control remoto https://www.researchgate.net/publication/348825241_Monitoreo_y_control_remoto_de_un_dispensador_de_alimento_para_mascotas_basado_en_IoT