Revista Dinamarca Descubre
N°6, Año: 2026
www.dinamarca.edu.co

Institución Educativa Dinamarca
"Un lugar donde se aprende y se es feliz"

Pet Dispen Dispensador automático de comida para mascotas

Automatic Pet Food Dispenser

Simón López 1[1], Grado: 10°2
Jesús David Cardoza Ávila 2[2], Grado: 10°2
Esteban Agudelo Cuervo 3[3], Grado: 10°1

Afiliación institucional: Grupo de Investigación ODIN

Asesor/a: *Yadir Alexander Agudelo Durango*

1. Resumen

Nuestro proyecto busca resolver un problema común entre los propietarios de mascotas: qué no saben qué hacer cuando deben salir de viaje o no tienen suficiente tiempo disponible para alimentarlas. Para ello, proponemos el desarrollo de una aplicación que controle un dispensador automático de comida, el cual permitirá alimentar a la mascota de forma remota o programada, incluso cuando su dueño no esté en casa.

Palabras clave: Dispensador, Mascotas, Cuido, Programación, Alimentación, Automático, Dosificación, Controlador

2. Abstract

Our project aims to solve a common problem faced by pet owners: not knowing what to do when traveling or lacking the time to feed their pets. To address this, we propose developing an application that controls an automatic food dispenser, enabling pets to be fed remotely or on a schedule, even when the owner is away from home.

Keywords: Dispenser, Pets, Care, Programming, Feeding, Automatic, Dosing, Controller

[1] Correo electrónico institucional autor/a 1
[2] Correo electrónico institucional autor/a 2
[3] Correo electrónico institucional autor/a 3

3. Introducción

Hoy en día, muchas personas con mascotas enfrentan dificultades para alimentarlas a tiempo debido a sus extensas jornadas laborales, viajes o compromisos diarios. Esta falta de tiempo puede afectar la salud y la rutina de los animales de compañía.

Para responder a esta necesidad, el proyecto "Dispensador automático de comida para mascotas", desarrollado por estudiantes de la Media Técnica en Desarrollo de Software de la Institución Educativa Dinamarca, propone la creación de una aplicación web conectada a un dispensador automático. Esta herramienta permite a los dueños programar horarios y alimentar a sus mascotas a distancia de forma rápida y sencilla.

En este documento se presentan el planteamiento del problema, la justificación teórica, los objetivos, la metodología de trabajo y los productos esperados para demostrar la utilidad y correcto funcionamiento del sistema.

4. Pregunta de Investigación

¿Cómo podría el dispensador automático de alimento para mascotas facilitar a los dueños con poco tiempo libre el alimentar a sus mascotas mediante la implementación de una aplicación que permita accionar el dispensador de forma remota?

5. Objetivos

Objetivo General

Diseñar una aplicación que le permita a los usuarios sin disposición de tiempo libre que puedan alimentar a sus mascotas desde la distancia y lo más pronto posible.

Objetivos Específicos

1. Investigar las distintas formas de administrar el alimento a las mascotas para buscar mayor comodidad.
2. Adquirir las herramientas y habilidades relacionadas con la programación y diseño de aplicaciones web y de escritorio.
3. Implementar la aplicación web, que contenga la información pertinente.
4. Realizar las pruebas de la aplicación, para comprobar que todo funcione correctamente

6. Marco teórico y estado del arte.

6.1. Estado del Arte Desarrollo de Software

El desarrollo de software ha evolucionado de un proceso artesanal y secuencial a una disciplina estratégica, impulsada por la automatización, la inteligencia artificial (IA) y la necesidad de entregar valor empresarial a gran velocidad. En 2026, el **estado del arte** se caracteriza por la integración profunda de la IA en todo el ciclo de vida del

desarrollo (SDLC), la adopción masiva de arquitecturas cloud-native, el enfoque en la seguridad desde el diseño (DevSecOps) y la priorización de la sostenibilidad y la experiencia del desarrollador. Este resumen se basa en informes recientes de Gartner, DORA (DevOps Research and Assessment), Stack Overflow Developer Survey 2025 y análisis de industria como Innowise y Orienteed.

6.1.1 Evolución Histórica y Metodologías Dominantes

Tradicionalmente dominado por el modelo **Waterfall** (lineal y rígido), el desarrollo de software migró hacia enfoques **ágiles** (Scrum, Kanban) a partir de los años 2000, priorizando la iteración y la colaboración. En la última década, **DevOps** se consolidó como estándar, integrando desarrollo y operaciones mediante CI/CD (Continuous Integration/Continuous Deployment), Infrastructure as Code (IaC) y GitOps.

En 2025-2026, **Platform Engineering** emerge como evolución natural: equipos crean plataformas internas de autoservicio que mejoran la productividad de los desarrolladores (developer experience). El **DevSecOps** integra seguridad de forma proactiva ("shift-left"), mientras que el **MLOps** y **AIOps** extienden estos principios a sistemas de IA. Los informes DORA destacan que la IA actúa como **amplificador**: acelera el throughput en equipos maduros con arquitecturas loosely coupled y feedback rápido, pero magnifica problemas en equipos con procesos débiles.

6.1.2 Tecnologías y Lenguajes Actuales

- **Lenguajes**: Python ha experimentado un crecimiento acelerado (+7 puntos porcentuales en 2025 según Stack Overflow), impulsado por IA, data science y backend. Rust se consolida como estándar de rendimiento y seguridad (elimina bugs de memoria sin sacrificar velocidad), ideal para sistemas críticos en cloud y fintech. WebAssembly (WASM) permite ejecutar código universal en cualquier entorno (navegadores, edge, servidores). Otros destacados: Go, TypeScript y frameworks como FastAPI.

- **Arquitecturas**: Microservicios orientados a eventos (EDA), serverless (máxima madurez), bases de datos híbridas/NewSQL y sincronización edge. IaC y GitOps garantizan predictibilidad y Policy as Code para seguridad.

- **Herramientas**: GitHub Copilot y ChatGPT dominan la asistencia IA; plataformas como Ollama y LangChain para agentes IA; Redis y vector databases para memoria de agentes.

6.1.3 Tendencias Principales en 2025-2026

La IA es el eje central. Según Gartner, las **AI-Native Development Platforms** lideran las tendencias estratégicas: plataformas que usan GenAI desde el diseño para generar aplicaciones completas con menos código tradicional, permitiendo que equipos pequeños (incluso "desarrolladores ciudadanos") creen software empresarial. Se espera

que para 2030, el 80% de las organizaciones reduzcan equipos grandes en favor de unidades ágiles aumentadas por IA.

Otras tendencias clave:

- **IA Agentica y Multi-Agent Systems**: La IA pasa de asistente a operador autónomo que planifica, decide y orquesta tareas (generación de código, pruebas, refactorización). En 2026, más del 40% del código en algunas empresas se genera con IA. Los sistemas multi-agente colaboran en tareas complejas.
- **Low-Code/No-Code + IA**: Plataformas accesibles permiten a no-técnicos crear aplicaciones. Integradas con IA, aceleran MVP y liberan ingenieros para arquitectura compleja.
- **DevSecOps y Confidential Computing**: Seguridad proactiva, zero-trust y plataformas de IA Security. El coste medio de una brecha de datos supera los 4 millones de dólares.
- **Edge Computing e IA On-Device**: Procesamiento local reduce latencia y mejora privacidad. Convergencia AIoT (mercado proyectado en crecimiento explosivo).
- **Sostenibilidad (Green Software)**: Optimización de código, algoritmos eficientes y arquitecturas que minimizan emisiones de CO_2.
- **AR/VR, WebAR, Blockchain y Tokenización**: Aplicaciones empresariales en formación, simulación y activos reales.
- **Computación Cuántica**: Aún emergente, pero con inversiones crecientes para optimización compleja.

Datos de adopción (Stack Overflow Developer Survey 2025):

- 84% de desarrolladores usan o planean usar herramientas IA (51% diariamente).
- Solo el 60% tiene una postura favorable (bajó del 70%+ en años anteriores).
- 46% desconfía de la precisión de la IA; 66% se frustra con código “casi correcto”.

6.1.4 Desafíos Actuales

- **Confianza y gobernanza de IA**: Precisión, sesgos, seguridad y privacidad son preocupaciones mayores (87% preocupados por agentes IA).

- **Talento y burnout**: Escasez de perfiles senior; IA aumenta productividad individual pero no siempre colaboración o estabilidad del sistema.
- **Técnicos**: Sistemas tightly coupled limitan beneficios de IA; deuda técnica y complejidad de arquitecturas híbridas.
- **Éticos y regulatorios**: Gobernanza de IA, sostenibilidad y soberanía tecnológica.

6.1.5 Perspectivas Futuras

Hacia 2030, el desarrollo de software será predominantemente **AI-augmented**, con plataformas nativas de IA, agentes autónomos y enfoques híbridos (cloud-edge). La ingeniería de plataformas y la cultura organizacional determinarán quién obtiene el máximo valor de estas tecnologías. En regiones como Latinoamérica (incluyendo Colombia), la adopción sigue tendencias globales, posicionándose como hub de nearshoring con fuerte énfasis en IA y cloud.

En resumen, el estado del arte en 2026 no se define solo por tecnologías, sino por la capacidad de las organizaciones para **integrar IA de forma responsable**, construir plataformas resilientes y priorizar tanto la velocidad como la sostenibilidad y la seguridad. Las empresas que adopten estas prácticas no solo entregarán software más rápido, sino que transformarán su modelo de negocio en plataformas vivas y adaptativas.

6.2 Estado del arte: Dispensadores automáticos de comida para mascotas en repositorios de universidades colombianas

El tema de los dispensadores automáticos de comida para mascotas (también llamados comederos o alimentadores inteligentes) ha sido abordado en varios proyectos de grado y prototipos desarrollados en universidades colombianas, principalmente en programas de Ingeniería Electrónica, Mecatrónica e Ingeniería de Sistemas. Estos trabajos responden a la necesidad de facilitar el cuidado de mascotas (perros y gatos principalmente) cuando los dueños están ausentes, abordando problemas como la dosificación precisa, el control remoto y el monitoreo.

La evolución tecnológica observada en los repositorios institucionales (como Bibliotecadigital Univalle, repositorios de UNAD, UNAB y UTP) muestra un progreso claro: desde sistemas locales temporizados con microcontroladores básicos (principios de los 2000) hacia soluciones IoT con conectividad Wi-Fi, protocolos como MQTT y control mediante apps móviles o bots (década de 2020). Los proyectos enfatizan bajo costo, precisión en la dosificación (generalmente ±10-22 g), sensores de nivel/peso y mecanismos mecánicos anti-obstrucción (vibradores o bandas). Limitaciones comunes incluyen dependencia de energía eléctrica estable, falta de IA para reconocimiento de mascota y escalabilidad para múltiples porciones o razas específicas.

A continuación, se detallan **cuatro proyectos representativos** identificados en repositorios universitarios colombianos, seleccionados por su relevancia, diversidad tecnológica y disponibilidad en acceso abierto. Se priorizaron trabajos de pregrado de universidades públicas y privadas.

6.2.1 Universidad del Valle (2013) – Enfoque en control remoto vía SMS y app Android

Título: Dispensador automático de comida para mascotas, programable y controlado remotamente (Feed Your Pet – FYP).

Autores: John David León Quenguan y Daniel Rueda Almario.

Programa: Ingeniería Electrónica, Facultad de Ingeniería.

Este proyecto desarrolla un prototipo con tolva de acrílico (capacidad 8 kg), dosificación por banda transportadora accionada por motor DC y vibrador. Incluye sensores infrarrojos para nivel de tolva (9 niveles) y celda de carga para peso del plato (±10 g de precisión). El control es local (teclado matricial + LCD) y remoto vía módulo GPRS (SMS) con app Android desarrollada en Java (Eclipse). Permite programar hasta 3 horarios diarios, alarmas por bajo nivel y llamado a la mascota (bocina). El firmware corre en microcontrolador ATmega644 con FreeRTOS.

Resultados clave: Error de dosificación ±22 g; consumo 18 W; costo aproximado $1.167.000 COP (2013). Se destaca la innovación en conectividad remota vía redes móviles, ausente en comederos comerciales de la época (como Petmate o PetSafe).

Cita APA: León Quenguan, J. D., & Rueda Almario, D. (2013). Dispensador automático de comida para mascotas, programable y controlado remotamente [Proyecto de grado]. Universidad del Valle. https://bibliotecadigital.univalle.edu.co/bitstream/10893/9148/1/CB-0527751.pdf

6.2.2 Universidad Tecnológica de Pereira (UTP, 2017 y actualización 2024) – Raspberry Pi y control IoT vía app/Telegram

Título principal (2017): Diseño e implementación de un prototipo de dispensador automático de comida para animales basado en Raspberry Pi controlado mediante una aplicación móvil.

Autor (2017): Jorge Iván Zapata Valencia (referenciado en trabajos posteriores).

Título relacionado (2024): Propuesta para desarrollo de un comedero inteligente de mascotas con control a través de Telegram programado por medio de una ESP32.

Autor (2024): Freddy Santiago Sánchez Villaneda.

El proyecto de 2017 utiliza Raspberry Pi como núcleo central con app móvil para control remoto (servidor web/HTTP). El de 2024 evoluciona hacia ESP32 con integración de bot de Telegram para comandos remotos (IoT). Incluye tolva de doble compartimento (diseño en SolidWorks), mecanismos de dosificación experimental y énfasis en asequibilidad y gestión remota.

Resultados clave: Enfoque en integración de plataformas móviles/web para control desde cualquier lugar, superando limitaciones de sistemas GSM anteriores. Sirve de base para prototipos posteriores en IoT.

Cita APA (2017): Zapata Valencia, J. I. (2017). Diseño e implementación de un prototipo de dispensador automático de comida para animales basado en Raspberry Pi controlado mediante una aplicación móvil [Proyecto de grado]. Universidad Tecnológica de Pereira. Repositorio UTP.

Cita APA (2024): Sánchez Villaneda, F. S. (2024). Propuesta para desarrollo de un comedero inteligente de mascotas con control a través de Telegram programado por medio de una ESP32 [Proyecto]. Universidad Tecnológica de Pereira. https://repositorio.utp.edu.co/bitstreams/17836289-3ed7-4e63-bade-d0ab1e6b7064/download

6.2.3 Universidad Nacional Abierta y a Distancia (UNAD, 2025) – IoT con ESP32 y MQTT

Título: Diseño e implementación de prototipo de dispensador remoto de comida para perros y gatos a través de IoT.

Autor: John Jaiver Ardila Vargas (asesorado por Ing. Jairo Luis Gutiérrez).

Programa: Ingeniería Electrónica, Escuela de Ciencias Básicas, Tecnología e Ingeniería.

Prototipo con tolva impresa en 3D (PLA), dosificación por gravedad mediante servomotor MG90S y semicodo de PVC (porciones ~60 g). Incluye motor vibratorio anti-compactación, control local (botón) y remoto vía ESP32-WROOM-32 (Wi-Fi) con protocolo MQTT (bróker gratuito emqx.io) y app móvil PetFoodApp desarrollada en MIT App Inventor. Estructura en MDF.

Resultados clave: Precisión ±4-8 % en pruebas remotas (Bogotá, Medellín y España); funcionamiento estable durante 20 días. Demuestra viabilidad de IoT de bajo costo para domótica pet-friendly, con latencia baja y retroalimentación en app.

Cita APA: Ardila Vargas, J. J. (2025). Diseño e implementación de prototipo de dispensador remoto de comida para perros y gatos a través de IoT [Tesis de pregrado]. Universidad Nacional Abierta y a Distancia (UNAD). https://repository.unad.edu.co/jspui/bitstream/10596/70833/3/jardilav.pdf

6.2.4 Universidad Autónoma de Bucaramanga (UNAB, 2025) – Sistema integral dosificador + entretenimiento con ESP32

Título: Desarrollo de un sistema dosificador y de entretenimiento automático para perros de tamaño mediano.

Autores: Julián Andrés Carvajal Galofre y Paula Andrea Portilla Corredor.

Programa: Ingeniería Mecatrónica, Facultad de Ingeniería.

Sistema multifunción (comida, agua y entretenimiento) con ESP32, celda de carga HX711 (precisión ±0.3 g), sensores de nivel ópticos, electroválvula, solenoide para lanzador de pelota y cámara ESP32-CAM. App web "Dosifican" (Node.js + Socket.IO) permite programación de horarios, monitoreo en tiempo real y modo juego. Estructura modular en acrílico/PLA/ acero inoxidable.

Resultados clave: Precisión >95-99 % en pruebas con perros reales; conectividad Wi-Fi estable; operación autónoma >48 h. Integra entretenimiento (lanzador de pelota) y monitoreo visual, superando proyectos solo dosificadores. Limitaciones: sensibilidad a vibraciones y dependencia de Wi-Fi.

Cita APA: Carvajal Galofre, J. A., & Portilla Corredor, P. A. (2025). Desarrollo de un sistema dosificador y de entretenimiento automático para perros de tamaño mediano [Trabajo de grado]. Universidad Autónoma de Bucaramanga. https://repository.unab.edu.co/bitstream/handle/20.500.12749/30858/Libro_proyecto_de_grado_Portilla_Carvajal.pdf?sequence=1&isAllowed=y

6.3 Análisis comparativo y tendencias

- **Evolución tecnológica:** Los proyectos de 2013 (GSM/SMS) dan paso a Raspberry Pi (2017) y luego a ESP32 + IoT/MQTT/Telegram (2024-2025), reduciendo costos y mejorando latencia/alcance. La precisión de dosificación y el monitoreo (nivel/peso) son constantes; se añade entretenimiento y video en los más recientes.
- **Fortalezas comunes:** Bajo costo, diseño mecánico simple (tolvas + actuadores) y enfoque en usabilidad remota.
- **Brechas identificadas:** Pocos incorporan IA (reconocimiento de mascota o ajuste automático por peso/raza), sensores ambientales o integración con voz (Alexa/Google). La mayoría asume conexión estable y no aborda fallos de energía o ciberseguridad profunda.
- **Impacto en Colombia:** Responden al crecimiento del mercado de mascotas (datos DANE citados en proyectos recientes) y promueven soluciones asequibles para hogares colombianos.

6.4 Dispensador automático de alimento para mascotas

El marco teórico de un dispensador automático de comida para mascotas (también denominado comedero inteligente o alimentador automático) se fundamenta en la integración de principios de **mecatrónica**, **automatización industrial**, **Internet de las Cosas (IoT)** y **cuidado animal**. Este marco proporciona las bases conceptuales, tecnológicas y normativas que sustentan el diseño, implementación y operación de estos sistemas, permitiendo la dosificación precisa y remota de alimento para perros y gatos, especialmente en ausencia de los propietarios.

6.4.1 Conceptos básicos y definiciones

Un **dispensador automático de comida para mascotas** es un dispositivo programable que regula el suministro de alimento sólido (croquetas secas) en porciones controladas, mediante mecanismos mecánicos y electrónicos, con capacidad de operación local o remota. Se distingue de los comederos manuales por su autonomía y precisión en la dosificación volumétrica o gravimétrica.

La **dosificación** se define como el proceso de administración precisa de alimento considerando edad, tamaño, raza, actividad física y estado de salud del animal, evitando subalimentación o sobrealimentación. Según recomendaciones nutricionales, los perros adultos requieren 2-3 comidas diarias, ajustadas por peso (ej. Purina Dog Chow, 2015, citado en proyectos universitarios).

Los **sistemas domóticos** (domótica) son conjuntos de dispositivos interconectados en el hogar que automatizan tareas para mejorar la comodidad, eficiencia energética y monitoreo remoto. En este contexto, el dispensador forma parte de la **domótica pet-friendly**.

6.4.2 Fundamentos nutricionales y de bienestar animal

La alimentación automática responde a necesidades biológicas: los perros y gatos requieren acceso constante a porciones frescas para prevenir obesidad, desnutrición o estrés. La Ley 1774 de 2016 (Colombia) y el Estatuto Nacional de Protección Animal (Ley 84 de 1989) obligan a garantizar alimentación suficiente y oportuna, evitando sufrimiento por hambre o sed. Los dispensadores

automatizados contribuyen al **bienestar animal** al mantener horarios consistentes y porciones controladas (±4-22 g de precisión en prototipos colombianos).

6.4.3 Mecanismos de dosificación mecánica

Los mecanismos de dosificación para sólidos secos se clasifican según su principio de operación:

- **Tornillo sin fin (auger)**: El alimento avanza proporcionalmente a la velocidad del motor. Simple y adaptable, pero menos preciso en flujos irregulares (Torres, 2012).
- **Compuerta rotativa**: Construcción robusta; precisión depende de poleas. Reduce contacto directo con el alimento (Consuegra & González, 2011).
- **Banda transportadora o rodante**: Controlada por motor DC + reductora; ideal para evitar estancamiento de croquetas. Se combina con vibradores para anti-compactación (León Quenguan & Rueda Almario, 2013).

Otros diseños emplean **gravedad** con semicodo de PVC y paleta/servomotor (porciones ~60 g) o compuerta rotativa con cálculo de torque:

$$T=Fg(1+\mu)\ R \quad T = F_g\ (1 + \mu)\ R \quad T=Fg(1+\mu)\ R$$

donde $Fg=m\cdot g$ $F_g = m \cdot g$ $Fg=m\cdot g$ (fuerza gravitatoria) y μ μ μ es el coeficiente de fricción (Carvajal Galofre & Portilla Corredor, 2025).

El **recipiente/tolva** (capacidad 2,8-8 kg) se fabrica en acrílico, PLA (impresión 3D), LDPE o acero inoxidable para preservar la integridad del alimento, cumplir regulaciones de migración de sustancias (Castillo, 2014; Pinto & Durán, 2006) y facilitar limpieza.

6.4.4 Aspectos electrónicos y microcontroladores

El núcleo de control suele ser un **microcontrolador embebido**:

- **ATmega644** (proyectos 2013): 16 MHz, 32 KB flash, FreeRTOS para multitarea (monitoreo de nivel, celda de carga, GPRS).
- **ESP32** (proyectos 2024-2025): Dual-core 240 MHz, Wi-Fi/Bluetooth integrado, bajo consumo, pines GPIO para sensores y actuadores. Ideal para IoT por su costo y versatilidad (Ardila Vargas, 2025; Sánchez Villaneda, 2024).

Sensores:

- Infrarrojos o ópticos (nivel de tolva, hasta 9 niveles).
- Celdas de carga + HX711 (±0,3 g precisión, calibración lineal).
- Pulsadores para control local.

Actuadores:

- Servomotores MG90S (ángulo preciso 30°-45°).
- Motores DC/vibradores (anti-obstrucción).
- Electroválvulas o solenoides (en sistemas multifunción con agua o entretenimiento).

La **celda de carga** se calibra mediante regresión lineal (ej. $y=909{,}09x-636{,}36$ $y = 909{,}09x - 636{,}36$ $y=909{,}09x-636{,}36$, $R2=1$ $R^2 = 1$ $R2=1$) para medir peso en plato.

6.4.5 Tecnologías IoT y protocolos de comunicación

El **Internet de las Cosas (IoT)**, acuñado por Kevin Ashton (2009), es la interconexión de dispositivos físicos que intercambian datos vía internet sin intervención humana (Fernández, 2021). En dispensadores, permite control remoto y monitoreo en tiempo real.

Protocolo MQTT (Message Queuing Telemetry Transport): Ligero, basado en publish/subscribe sobre TCP/IP. Incluye broker (ej. emqx.io), topics y clientes. Ideal para redes de bajo ancho de banda y dispositivos embebidos (Llamas, 2019; Cerón, 2024). Ejemplo: tópicos “casa/food” para comandos ON/OFF.

Comunicación alternativa:

- GSM/GPRS + SMS (proyectos antiguos, módulo Quectel M95).
- Bots de Telegram (ESP32 + Bot API).
- Apps móviles (MIT App Inventor o web con Node.js + Socket.IO).

La arquitectura típica es: usuario → app/broker → ESP32 → actuadores + sensores (retroalimentación).

6.4.6 Normatividad colombiana y consideraciones de seguridad

- **Código Sanitario Nacional** (Ley 9 de 1979) y Decreto 1500 de 2007: regulan manejo y almacenamiento de alimentos para animales.
- **Resolución 294 de 2012**: estándares de calidad e inocuidad para alimentos de mascotas.
- Materiales deben evitar contaminación (acrílico/PLA apto para alimentos).

Se recomienda cumplimiento de prácticas de higiene, fuentes de alimentación estable (5V/12V con reguladores LM2596/LM338) y protección contra sobrecorrientes (diodos flyback, transistores Darlington).

6.4.7 Principios de funcionamiento y evolución tecnológica

El funcionamiento integra:

1. Almacenamiento en tolva.
2. Activación (horario/programado/remoto) → actuador + vibrador.
3. Medición (celda de carga/sensor nivel) → detención precisa.
4. Retroalimentación (app, SMS, LED).

La evolución en proyectos colombianos pasa de GPRS/SMS (2013) a ESP32 + MQTT/Telegram (2024-2025), incorporando entretenimiento (lanzador de pelotas) y monitoreo visual (ESP32-CAM). La precisión típica oscila entre ±4-22 g, con operación autónoma >20-48 h.

Este marco teórico, sustentado en tesis de Universidad del Valle (2013), UNAD (2025), UNAB (2025) y UTP (2024), demuestra que los dispensadores automáticos son una aplicación madura de mecatrónica e IoT, alineada con el bienestar animal y las necesidades de hogares colombianos.

6.5 Marco Teórico del Desarrollo de Software

El marco teórico del desarrollo de software constituye la base conceptual y metodológica para cualquier proyecto de grado en ingeniería de sistemas, informática o áreas afines. Este capítulo presenta los conceptos fundamentales de la ingeniería de software, su ciclo de vida, metodologías, herramientas y tecnologías clave (incluyendo lenguajes de programación, bases de datos, arquitecturas y más). Se basa en principios

establecidos por autores como Sommerville y estándares como los del IEEE, con el objetivo de proporcionar un fundamento sólido que oriente el diseño, implementación y evaluación del proyecto.

6.5.1 Ingeniería de Software y Conceptos Básicos

La **ingeniería de software** es la disciplina de la ingeniería que aplica principios científicos, métodos y herramientas para el desarrollo, operación y mantenimiento de software de manera sistemática, disciplinada y cuantificable. Según el IEEE (1993), se define como "la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software".

El **software** es el conjunto de programas, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema informático. Se diferencia del hardware (componente físico) por ser intangible y lógico. Su desarrollo busca resolver problemas de manera eficiente, confiable y adaptable a las necesidades del usuario.

La importancia de la ingeniería de software radica en gestionar la complejidad de los proyectos, controlar costos, tiempos y calidad, y minimizar riesgos (como errores que generan altos costos de corrección). Su evolución ha pasado de enfoques ad-hoc (como "codificar y corregir") a procesos estructurados y ágiles.

6.5.2 Ciclo de Vida del Desarrollo de Software (SDLC)

El **Ciclo de Vida del Desarrollo de Software (SDLC)** es el proceso estructurado que describe las etapas para crear, mantener y retirar un sistema de software de manera eficiente. Proporciona un marco sistemático con entregables específicos en cada fase.

Las fases principales del SDLC son:

- **Planificación**: Identificación de objetivos, alcance, recursos y viabilidad (costo-beneficio).
- **Análisis de requisitos**: Elicitación, documentación y validación de requisitos funcionales (qué hace el sistema) y no funcionales (rendimiento, seguridad, usabilidad).
- **Diseño**: Definición de la arquitectura, componentes y modelos (lógico y físico).
- **Implementación (Codificación)**: Construcción del software mediante programación.
- **Pruebas**: Verificación y validación (unitarias, de integración, sistema y aceptación).
- **Despliegue**: Instalación y puesta en producción.
- **Mantenimiento**: Correcciones, adaptaciones, mejoras perfectivas y preventivas (correctivo, adaptativo, perfectivo y preventivo).

Los modelos de SDLC incluyen el **Modelo en Cascada** (secuencial y lineal, ideal para requisitos estables), **Modelo en Espiral** (iterativo con énfasis en riesgos), **Modelo Evolutivo/Incremental** y **Modelo de Transformación** (automatizado).

6.5.3 Metodologías de Desarrollo de Software

Las **metodologías** son conjuntos integrados de técnicas, métodos y procedimientos que guían el ciclo de vida del proyecto. Se dividen en tradicionales y ágiles.

- **Metodologías Tradicionales**: Enfocadas en planificación exhaustiva, documentación y control riguroso. Ejemplos: Cascada (Waterfall), Prototipado, Espiral, Incremental y RAD (Rapid Application Development). Son predictivas pero poco flexibles ante cambios.

- **Metodologías Ágiles**: Basadas en el **Manifiesto Ágil** (2001), priorizan individuos e interacciones, software funcional, colaboración con el cliente y respuesta al cambio. Son iterativas, incrementales y adaptativas. Ejemplos principales:
    - **Scrum**: Iteraciones cortas (sprints de 2-4 semanas), roles (Product Owner, Scrum Master, Equipo), artefactos (Product Backlog, Sprint Backlog) y eventos (Daily Scrum, Sprint Review, Retrospective).
    - **Kanban**: Flujo visual continuo con tableros, límites de trabajo en progreso (WIP) y enfoque en optimización del flujo.

    - **Extreme Programming (XP)**: Prácticas como programación en pares, pruebas unitarias continuas (TDD), integración continua y simplicidad.
    - Otras: Crystal, FDD, Lean Software Development y Proceso Unificado (RUP).

La elección depende del tamaño del equipo, incertidumbre de requisitos y contexto del proyecto.

6.5.4 Ingeniería de Requisitos y Diseño de Software

La **ingeniería de requisitos** involucra la elicitación (entrevistas, observación), análisis, especificación y validación de necesidades del usuario y del sistema.

El **diseño de software** define cómo se construirá el sistema. Incluye:

- **Arquitecturas**: Monolítica (todo en un solo bloque), cliente-servidor, de tres capas (presentación, lógica de negocio, datos), **Microservicios** (servicios independientes y escalables) y orientada a eventos.
- **Patrones de Diseño**: Soluciones reutilizables a problemas comunes (Gang of Four). Categorías: creacionales (Singleton, Factory), estructurales (Adapter,

Facade) y de comportamiento (Observer, Strategy). Ejemplo clave: **MVC (Model-View-Controller)** para separar datos, interfaz y lógica.

- **UML (Unified Modeling Language)**: Lenguaje estándar para modelar. Diagramas principales: Casos de Uso, Clases, Secuencia, Actividad, Estados y Componentes.

6.5.5 Lenguajes de Programación y Paradigmas

Los **lenguajes de programación** son herramientas formales para expresar algoritmos e instrucciones a la computadora. Su selección depende del dominio (web, móvil, datos, sistemas embebidos), rendimiento y ecosistema.

Paradigmas principales (estilos de programación):

- **Imperativo/Procedural**: Enfocado en secuencias de comandos (C, Pascal).
- **Orientado a Objetos (OOP)**: Basado en clases, objetos, herencia, polimorfismo y encapsulamiento (Java, C++, Python, C#).
- **Funcional**: Enfocado en funciones puras, inmutabilidad y evitación de estado (Haskell, Scala, partes de JavaScript/Python).
- **Declarativo**: Describe el qué, no el cómo (SQL, HTML/CSS).

Lenguajes más importantes y populares (2025):

- **Python**: Versátil, simple y potente en IA, ciencia de datos y scripting.
- **JavaScript/TypeScript**: Esencial para desarrollo web (frontend/backend con Node.js).
- **Java**: Robusto para aplicaciones empresariales y Android.
- **C#**: Ideal para .NET, juegos (Unity) y Windows.
- Otros: C++ (rendimiento alto), Go (concurrencia y microservicios), SQL (bases de datos).

6.5.6 Bases de Datos

Las **bases de datos** almacenan y gestionan datos de forma persistente y eficiente. Son fundamentales en casi todo proyecto de software.

- **Bases de Datos Relacionales (SQL)**: Basadas en el modelo relacional (tablas, filas, columnas y claves). Cumplen propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad). Ejemplos: MySQL, PostgreSQL, Oracle. Incluyen diseño con diagramas ER, normalización (1NF a 5NF) para evitar redundancias y consultas con SQL.
- **Bases de Datos NoSQL**: Esquema flexible, escalables horizontalmente y orientadas a datos no estructurados o semiestructurados. Cumplen BASE (Basically Available, Soft state, Eventual consistency) y priorizan el teorema CAP (Consistency, Availability, Partition tolerance). Tipos principales:
    - Documentos (MongoDB – JSON-like).
    - Clave-Valor (Redis).

- Columnas anchas (Cassandra).
- Grafos (Neo4j).

La elección depende de la estructura de datos, escalabilidad y necesidades de consultas.

6.5.7 Implementación, Pruebas, Calidad y Mantenimiento

- **Control de Versiones**: Git es el estándar (comandos básicos: commit, branch, merge, pull/push). Plataformas: GitHub, GitLab.
- **Pruebas**: Unitarias, de integración, sistema, aceptación. Enfoques: TDD (Test-Driven Development), BDD. Automatización con frameworks como JUnit, Selenium.
- **Calidad**: Estándares ISO/IEC 25010 (funcionalidad, eficiencia, mantenibilidad, seguridad, usabilidad). Principios SOLID, DRY, KISS, YAGNI.
- **Mantenimiento**: Corregir errores y adaptar el software post-despliegue (representa hasta el 60-80% del costo total).
- **DevOps y CI/CD**: Cultura de colaboración entre desarrollo y operaciones. Herramientas: Docker (contenedores), Kubernetes (orquestación), Jenkins/GitHub Actions (integración/despliegue continuo) y cloud (AWS, Azure, Google Cloud).

6.6 Tendencias Actuales y Conclusión del Marco Teórico

Tendencias incluyen computación en la nube, inteligencia artificial en desarrollo (low-code/no-code, GitHub Copilot), seguridad (DevSecOps), arquitectura serverless y sostenibilidad (green software).

Este marco teórico proporciona los conceptos esenciales para cualquier proyecto de grado. Su aplicación práctica asegura un software de calidad, mantenible y alineado con las necesidades reales. Se recomienda complementar con referencias clásicas como Ingeniería del Software de Sommerville o Ingeniería del Software: Un enfoque práctico de Pressman, junto con documentación oficial de metodologías y tecnologías.

7. Metodología

El proyecto se desarrollará mediante una **metodología híbrida** basada en el Ciclo de Vida del Desarrollo de Software (SDLC) con elementos ágiles. Se dividirá en cuatro etapas, cada una alineada con un objetivo específico.

7.1 Etapa 1: Investigación (Objetivo Específico 1)

Actividades: Revisión del estado del arte de dispensadores automáticos, aplicación de encuestas a dueños de mascotas de la institución educativa Dinamarca y definición de requisitos funcionales y no funcionales de la aplicación.

Recursos: Computadores, acceso a internet y bibliografía existente. Tiempo: 2 semanas.

Entregable: Documento de análisis de requisitos con matriz de requisitos.

7.2 Etapa 2: Adquisición de herramientas y diseño (Objetivo Específico 2)

Actividades: Capacitación en programación web, diseño de la arquitectura de la aplicación y creación de wireframes de baja fidelidad.

Recursos: Herramientas de desarrollo (VS Code, Figma, Git), computadores. Tiempo: 3 semanas.

Entregable: Wireframes y diagrama de arquitectura de la aplicación.

7.3 Etapa 3: Implementación (Objetivo Específico 3)

Actividades: Desarrollo del frontend y backend de la aplicación web, e integración con el prototipo físico del dispensador automático de comida para mascotas.

Recursos: Prototipo del dispensador, módulo de comunicación (ESP32), servidor de pruebas. Tiempo: 4 semanas.

Entregable: Aplicación web funcional integrada con el dispensador.

7.4 Etapa 4: Pruebas y validación (Objetivo Específico 4)

Actividades: Ejecución de pruebas unitarias, de integración y de usabilidad con usuarios reales. Corrección de errores.

Recursos: Dispositivos de prueba y usuarios externos. Tiempo: 2 semanas.

Entregable: Reporte de pruebas y versión final de la aplicación lista para demostración.

8. Resultados

Como resultado del desarrollo del proyecto Dispensador automático de comida para mascotas, se espera obtener los siguientes productos:

- **8.1 Aplicación web funcional**: Una aplicación que permita a los dueños de mascotas programar horarios de alimentación, activar el dispensador de forma remota y recibir notificaciones en tiempo real (nivel de comida y dispensaciones realizadas). La aplicación estará integrada con el prototipo físico del dispensador.

- **8.2 Prototipo integrado**: Sistema completo formado por el dispensador automático de comida (ya validado en prototipos previos) y la aplicación web, permitiendo el control remoto y programado de la alimentación.

- **8.3 Documentación técnica**: Manual de usuario y manual técnico de la aplicación, que incluya instrucciones de instalación, uso y mantenimiento del sistema.

- **8.4 Reporte de pruebas y validación**: Documento que evidencie las pruebas realizadas (funcionales, de integración y de usabilidad) con dueños de mascotas, demostrando que la aplicación reduce significativamente el tiempo y esfuerzo dedicado a la alimentación.

- **8.5 Video demostrativo**: Registro audiovisual del funcionamiento completo del sistema (activación remota, programación automática y monitoreo).

Estos productos permitirán entregar una solución práctica y útil para dueños de mascotas con poco tiempo libre, cumpliendo con el objetivo general del proyecto. Adicionalmente, el prototipo integrado podrá servir como base para futuros desarrollos o como material de apoyo para la Institución Educativa Dinamarca en temas de robótica, IoT y programación.

9. Discusión

El desarrollo e integración del Dispensador automático de comida para mascotas demuestra cómo la convergencia entre aplicaciones web y tecnologías del Internet de las Cosas (IoT) ofrece una respuesta efectiva a las dinámicas actuales de los propietarios con tiempo restringido.

Al comparar nuestra propuesta con los antecedentes registrados en repositorios universitarios colombianos, se evidencia una clara evolución tecnológica. Mientras que proyectos iniciales basaban su control remoto en mensajería SMS y redes GSM (como el sistema FYP de la Universidad del Valle en 2013), la implementación con arquitecturas web e IoT mediante microcontroladores como el ESP32 (similares a los trabajos de la UTP, UNAD y UNAB) permite un control más ágil, en tiempo real y con notificaciones instantáneas sobre el nivel de comida y las dispensaciones ejecutadas.

Desde los fundamentos del desarrollo de software, la aplicación de una metodología híbrida basada en el Ciclo de Vida del Desarrollo de Software (SDLC) con enfoque ágil garantizó una estructuración ordenada de requisitos, prototipado e integración. Este enfoque minimiza los errores de acoplamiento entre el frontend, backend y el prototipo físico.

Alcances y limitaciones:

Alcances: Se logró una solución funcional que permite programar y activar la dispensación a distancia, contribuyendo al cumplimiento estricto de las dietas de las mascotas y reduciendo la incertidumbre del cuidador durante ausencias o viajes. Además, promueve el bienestar animal al asegurar porciones adecuadas de alimento.

Limitaciones: El sistema mantiene una dependencia crítica de una conexión estable a internet (Wi-Fi) y de suministro eléctrico continuo. Asimismo, el prototipo actual está diseñado para alimento seco tipo croqueta, lo que limita su uso con alimento húmedo o dietas específicas.

Investigaciones futuras:

Se sugiere incorporar algoritmos de Inteligencia Artificial (IA) para el reconocimiento visual de la mascota mediante cámara, ajustando automáticamente la ración según su peso o raza. También se propone integrar mecanismos de alimentación de respaldo con baterías recargables y la inclusión de sensores de presencia ambiental o interacción sonora/video en tiempo real.

10. Conclusiones

- **Cumplimiento de objetivos:** Se diseñó una solución tecnológica accesible y remota que responde directamente a la problemática de los dueños de mascotas sin disponibilidad de tiempo libre, permitiéndoles alimentar a sus animales de compañía a distancia de forma oportuna.

- **Adquisición de competencias:** El proyecto permitió a los estudiantes de la Media Técnica en Desarrollo de Software adquirir y poner en práctica herramientas clave en desarrollo web, modelado de bases de datos, control de versiones e integración de hardware con software mediante el uso del microcontrolador ESP32.

- Aporte **al entorno escolar e investigación:** El prototipo integrado sirve como base pedagógica para la Institución Educativa Dinamarca y el Grupo de Investigación ODIN,

demostrando la viabilidad de implementar proyectos de domótica e IoT de bajo costo en el ámbito escolar.

• Recomendaciones**:** Para futuras iteraciones, se recomienda fortalecer la ciberseguridad en la comunicación entre la aplicación y el bróker de mensajería (IoT), además de incorporar módulos de almacenamiento energético secundario para garantizar el funcionamiento ininterrumpido ante cortes de energía.

11. Referencias bibliográficas

Ardila Vargas, J. J. (2025). *Diseño e implementación de prototipo de dispensador remoto de comida para perros y gatos a través de IoT* [Tesis de pregrado, Universidad Nacional Abierta y a Distancia (UNAD)]. Repositorio Institucional UNAD. https://repository.unad.edu.co/jspui/bitstream/10596/70833/3/jardilav.pdf

Carvajal Galofre, J. A., & Portilla Corredor, P. A. (2025). *Desarrollo de un sistema dosificador y de entretenimiento automático para perros de tamaño mediano* [Trabajo de grado, Universidad Autónoma de Bucaramanga (UNAB)]. Repositorio UNAB. https://repository.unab.edu.co/bitstream/handle/20.500.12749/30858/Libro_proyecto_de_grado_Portilla_Carvajal.pdf

DORA. (2025). *2025 DORA State of AI-Assisted Software Development Report*. Google Cloud. https://dora.dev/dora-report-2025/

Gartner. (2025, 20 de octubre). *Gartner identifies the top strategic technology trends for 2026*. Gartner Press Release. https://www.gartner.com/en/newsroom/press-releases/2025-10-20-gartner-identifies-the-top-strategic-technology-trends-for-2026

IEEE. (1993). *IEEE Standard glossary of software engineering terminology* (IEEE Std 610.12-1990). Institute of Electrical and Electronics Engineers.

León Quenguan, J. D., & Rueda Almario, D. (2013). *Dispensador automático de comida para mascotas, programable y controlado remotamente* [Proyecto de grado, Universidad del Valle]. Biblioteca Digital Univalle. https://bibliotecadigital.univalle.edu.co/bitstream/10893/9148/1/CB-0527751.pdf

Pressman, R. S., & Maxim, B. R. (2021). *Ingeniería del software: Un enfoque práctico* (9.ª ed.). McGraw-Hill.

Sánchez Villaneda, F. S. (2024). *Propuesta para desarrollo de un comedero inteligente de mascotas con control a través de Telegram programado por medio de una ESP32* [Proyecto de grado, Universidad Tecnológica de Pereira]. Repositorio UTP. https://repositorio.utp.edu.co/bitstreams/17836289-3ed7-4e63-bade-d0ab1e6b7064/download

Sommerville, I. (2017). *Software engineering* (10.ª ed.). Pearson.

Stack Overflow. (2025). *2025 Stack Overflow developer survey*. Stack Overflow. https://survey.stackoverflow.co/2025/

Zapata Valencia, J. I. (2017). *Diseño e implementación de un prototipo de dispensador automático de comida para animales basado en Raspberry Pi controlado mediante una aplicación móvil* [Proyecto de grado, Universidad Tecnológica de Pereira]. Repositorio UTP.