

Robot Basurero Autónomo con Navegación a Punto de Acopio

Universidad Nacional de Moquegua

Facultad de Ingeniería y Arquitectura

Escuela Profesional de Ingeniería de Sistemas e Informática



Presentado por:

Pongo Calderón, René
Quispe Llanque, Gabriel Omar
Argote Huancapaza, Juan Julio
Romucho Sullcahuaman, Paola Celeste

Ilo – Perú
Roobotica II

30 de Julio de 2025

Índice

1. Resumen	1
2. Introducción	1
2.1. Planteamiento del Problema	1
2.2. Justificación	2
2.3. Metodología y Diseño del Sistema	2
2.3.1. Arquitectura del Sistema	2
2.4. Objetivos	7
2.4.1. Objetivo General	7
2.4.2. Objetivos Específicos	8
2.5. Alcance y Limitaciones	8
3. Marco Teórico	9
3.1. Estado del Arte	9
3.2. Tecnologías de Implementación	11
3.2.1. Hardware	11
3.2.2. Software y Plataformas	18
3.3. Tecnologías y Arquitecturas de Software	18
3.3.1. Patrones de Arquitectura de Software	18
3.3.2. Tecnologías del Frontend	19
3.4. Modelo Cinemático y Control	19
3.5. Protocolos de Comunicación	20
3.5.1. Pila de Protocolos TCP/IP	20
3.5.2. HTTPS (Hypertext Transfer Protocol Secure)	21
3.5.3. NTP (Network Time Protocol)	21
3.5.4. WebSocket para Comunicación en Tiempo Real	22
4. Diseño de la Propuesta	23
4.1. Requerimientos funcionales	24
4.2. Requerimientos no funcionales	25
4.3. Arquitectura del sistema	25
4.4. Implementación	26
4.4.1. Apertura automática de la tapa	26
4.4.2. Implementación del chasis y navegación	27
4.4.3. Implementación de sensores IR para navegación	27
4.4.4. Modelo Cinemático del Robot Diferencial	28
4.4.5. Controlador PID para Seguimiento de Línea	29
4.4.6. Calibración de Motores: De Velocidad a PWM	29
4.4.7. SPA Dashboard - Monitoreo	31
5. Pruebas y Resultados	32
5.1. Pruebas Experimentales	32
5.1.1. Validación del Sistema de Tapa Automática (Objetivo 1)	32

5.1.2. Validación del Monitoreo de Llenado (Objetivo 2)	33
5.1.3. Validación de la Navegación Autónoma (Objetivo 3)	34
5.2. Resultados Obtenidos	34
6. Discusión	35
7. Conclusiones	36
8. Recomendaciones y Trabajo Futuro	37
9. Anexos	37

Índice de figuras

1.	Distribución de Componentes en el Diseño Mecánico del Robot	4
2.	FSM	5
3.	Microcontrolador ESP32	12
4.	Sensor ultrasónico HC-SR04	12
5.	Sensor de distancia láser VL53L0X	13
6.	Sensor infrarrojo TCRT5000	14
7.	Servomotor SG90	14
8.	Motores DC	15
9.	Driver L298N para motores DC	16
10.	Sensor MPU-6500	17
11.	Módulo regulador de voltaje XL6009	18
12.	Solicitud de una sola página (SPA)	19
13.	Control PID	20
14.	Protocolo TCP/IP	21
15.	Protocolo HTTPS	21
16.	Protocolo NTP	22
17.	WebSocket para Comunicación en Tiempo Real	22
18.	Esquema de Arquitectura de Sistema	26
19.	Esquema de Arquitectura del Sistema.	27
20.	Chasis con sensores y ESP32 integrados	27
21.	Sensores infrarrojos montados para la detección de línea.	28
22.	Relación entre velocidad y PWM	31
23.	Panel de monitoreo desde la interfaz web integrada.	32
24.	Calibración del servomotor	33
25.	Prueba de proximidad de basura	33
26.	Datos robot visto en el firebase	38
27.	Datos robot en formato json	38

Índice de cuadros

1.	Tabla de Conexiones de Pines del ESP32	3
2.	Diccionario de Datos del Objeto JSON del Sensor Inteligente	7
3.	Análisis comparativo de proyectos relacionados con gestión inteligente de residuos	10
4.	Características técnicas del ESP32	11
5.	Características del sensor HC-SR04	12
6.	Características del sensor VL53L0X	13
7.	Características del sensor TCRT5000	13
8.	Características del servomotor SG90	14
9.	Características del Micro Motorreductor	15
10.	Características del driver L298N	16
11.	Características del sensor inercial MPU-6500	17
12.	Características técnicas del módulo elevador XL6009 Step-Up	17
13.	Requisitos Funcionales (RF) y Métricas de Éxito	24
14.	Requisitos No Funcionales (RNF) y Métricas de Calidad	25
15.	Puntos de Datos para la Calibración Velocidad-PWM	30
16.	Proceso de Sintonización de Constantes PID	35
17.	Resultados Finales de Navegación del Controlador PID	35

1. Resumen

Este proyecto detalla el diseño, la implementación y la validación de un sistema robótico móvil para la gestión automatizada de residuos sólidos en el interior de los entornos. El prototipo consiste en un robot basurero que funciona como un punto de recolección estacionario y autónomo asistido. El sistema emplea un sensor ultrasónico para la apertura sin contacto de la tapa y un sensor de Tiempo de Vuelo (ToF) para la medición precisa del nivel de llenado. Un microcontrolador ESP32 que ejecuta los algoritmos de control y gestiona la conectividad WiFi, transmitiendo los datos en tiempo real a una base de datos NoSQL en la nube (Firebase Realtime Database) para su visualización en un dashboard. La innovación clave del sistema es su capacidad de cuando alcanza un umbral de llenado predefinido, transita a un modo de navegación autónoma. Empleando un algoritmo de control PID y sensores infrarrojos, el robot sigue una trayectoria designada hasta un punto de acopio centralizado, donde el personal de limpieza puede vaciarlo. Este modelo invierte la logística tradicional: en lugar de que el personal se desplace a cada contenedor, el contenedor se desplaza hacia el personal. Tras el vaciado, el sistema retorna a su estación de origen, reiniciando el ciclo y validando su viabilidad como una solución eficiente e higiénica. Siendo registrada la situación de llenado del contenedor en el dashboard en tiempo real.

2. Introducción

2.1. Planteamiento del Problema

El modelo actual de gestión de residuos sólidos urbanos (RSU) en entornos de alta concurrencia, como campus universitarios y edificios corporativos, se caracteriza por una notable ineficiencia logística. Este enfoque tradicional depende de rutas y horarios fijos para el personal de limpieza, quienes revisan y vacían los contenedores manualmente. Sin embargo, esta metodología no considera la variabilidad dinámica en ...la generación de residuos en diferentes espacios (aulas, oficinas, áreas comunes). Esto lleva a que algunos contenedores se desborden antes de la siguiente revisión programada, generando focos de insalubridad y desorden, mientras que otros son inspeccionados innecesariamente estando casi vacíos, lo que representa un desperdicio de recursos. La falta de un sistema adaptativo que responda a la demanda real perpetúa un ciclo de ineficiencia operativa, mala asignación de recursos y deficiencias en la higiene.

Diversos estudios científicos han puesto de manifiesto estas problemáticas y la necesidad de soluciones más inteligentes. La recolección de residuos con rutas y horarios fijos ha sido ampliamente criticada por su rigidez [33].

La recolección de residuos con rutas y horarios fijos ha sido ampliamente criticada por su rigidez. La variabilidad en la generación de residuos es un factor crítico que el modelo tradicional no aborda eficazmente. La ausencia de este monitoreo contribuye directamente a los problemas de higiene y al desperdicio de recursos mencionados [32].

La implementación de sistemas inteligentes de gestión de residuos se presenta como una alternativa viable para superar estas deficiencias. En resumen, la persistencia del modelo

logístico ineficiente en la gestión de RSU en instalaciones de alta concurrencia resulta en problemas operativos, ambientales y de salubridad. La literatura científica reciente respalda la urgencia de transitar hacia sistemas más inteligentes y adaptativos, basados en tecnologías como el IoT y el análisis de datos, para optimizar la recolección y garantizar un entorno más limpio y eficiente [24, 23].

2.2. Justificación

Este proyecto aborda dicha problemática invirtiendo el paradigma de la recolección de residuos. En lugar de que el personal se mueva hacia los contenedores, se propone un sistema donde el contenedor, una vez lleno, se mueve autónomamente hacia el personal. La justificación de este nuevo modelo es triple: : [36].

Optimización de Procesos: Este enfoque permite una transición de un modelo de "limpieza programada." uno de "limpieza por demanda." "limpieza dinámica". Se optimizan las horas-hombre del personal de mantenimiento, quienes, liberados de la tarea monótona de revisar cada contenedor, pueden enfocarse en labores de limpieza de mayor prioridad. Se asegura que la recolección ocurra precisamente cuando es necesario, eliminando tanto el problema de los desbordamientos prematuros como las revisiones innecesarias.

Mitigación de Riesgos Sanitarios: La operación sin contacto para el depósito de residuos y la eliminación de desbordamientos contribuyen directamente a mejorar los estándares de higiene.

Innovación y Escalabilidad: El proyecto sirve como una prueba de concepto (PoC) para un nuevo modelo logístico en la gestión de residuos, sentando las bases para futuras soluciones escalables en el ámbito de las ciudades y edificios inteligentes (Smart Cities/Buildings).

2.3. Metodología y Diseño del Sistema

2.3.1. Arquitectura del Sistema

Se adoptó una metodología de desarrollo de prototipos incrementales.

Arquitectura de Hardware El ESP32 funciona como unidad central, recibiendo datos de sensores (ultrasónico, ToF, IR) y controlando actuadores (servo y motores) mediante salidas PWM. El sistema se alimenta con baterías 18650.

Pin ESP32	Componente	Función / Propósito	Tipo
19	Sensor Ultrasonido	Trigger: Envía el pulso ultrasónico	Sensor
18	Sensor Ultrasonido	Echo: Recibe el rebote del pulso	Sensor
21	Sensor Láser (VL53L0X)	SDA: Línea de datos del bus I2C principal (Wire)	Sensor
22	Sensor Láser (VL53L0X)	SCL: Línea de reloj del bus I2C principal (Wire)	Sensor
4	Giroscopio (MPU-6500)	SDA: Línea de datos del bus I2C secundario (I2C_MPUS)	Sensor
16	Giroscopio (MPU-6500)	SCL: Línea de reloj del bus I2C secundario (I2C_MPUS)	Sensor
32	Seguidor de Línea (IZQ)	Lectura analógica del sensor izquierdo	Sensor
35	Seguidor de Línea (CEN)	Lectura analógica del sensor central	Sensor
34	Seguidor de Línea (DER)	Lectura analógica del sensor derecho	Sensor
17	Servomotor (Tapa)	Señal PWM: Controla el ángulo del servomotor	Actuador
25	Driver Motores (L298N)	ENA: Habilita y controla la velocidad del Motor A (PWM)	Actuador
26	Driver Motores (L298N)	IN1: Controla la dirección de giro del Motor A	Actuador
27	Driver Motores (L298N)	IN2: Controla la dirección de giro del Motor A	Actuador
13	Driver Motores (L298N)	ENB: Habilita y controla la velocidad del Motor B (PWM)	Actuador
14	Driver Motores (L298N)	IN3: Controla la dirección de giro del Motor B	Actuador
12	Driver Motores (L298N)	IN4: Controla la dirección de giro del Motor B	Actuador

Cuadro 1: Tabla de Conexiones de Pines del ESP32

Diseño Mecánico

El sistema se monta sobre un chasis tipo diferencial compuesto por dos ruedas motrices y una rueda loca posterior que proporciona estabilidad durante la navegación. La carcasa del contenedor fue diseñada para alojar los componentes electrónicos, permitiendo una distribución eficiente del espacio. Los sensores fueron ubicados estratégicamente para garantizar una detección precisa y evitar interferencias físicas entre ellos.

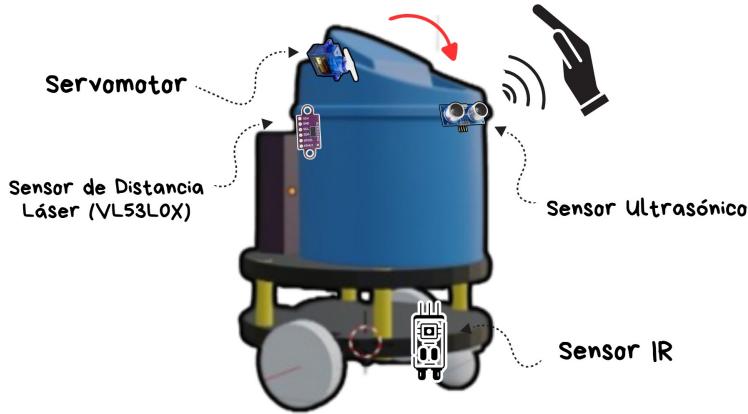


Figura 1: Distribución de Componentes en el Diseño Mecánico del Robot

Diseño del Software

Arquitectura del Firmware (Máquina de Estados) El software del microcontrolador ESP32 se estructura como una **máquina de estados finitos (FSM)**. Este patrón de diseño organiza el comportamiento del robot en un conjunto de estados discretos y define las transiciones entre ellos basándose en las entradas de los sensores y temporizadores. Este enfoque resulta en un código más organizado, mantenable y robusto para gestionar la lógica de control del robot.

El estado actual del sistema se almacena en la variable `estadoActual`, la cual puede adoptar uno de los siguientes valores definidos en `EstadoRobot`:

- **CALIBRANDO:** El estado inicial del robot al encenderse. En este modo, los motores están detenidos y el sistema espera a que se realicen las calibraciones de los sensores de línea y del giroscopio a través de la interfaz web. Una vez que ambas calibraciones son completadas, el sistema transita al siguiente estado.
- **ESPERANDO_EN_BASE:** El robot se encuentra en su punto de partida, detenido y funcional como un tacho de basura inteligente. Gestiona la apertura y cierre automático de la tapa al detectar una persona (`gestionarTapa`) y mide periódicamente el nivel de residuos (`medirNivelLlenado`). La transición al modo de transporte se activa cuando el `porcentajeLlenado` alcanza o supera el umbral definido (`UMBRAL_LLENO`).
- **VIAJE_A_DESTINO:** Al llenarse el contenedor, el robot cierra la tapa e inicia su desplazamiento siguiendo la línea en el suelo. El movimiento es controlado por la función `seguirLineaPID`, que ajusta la velocidad de los motores para mantener el robot centrado en la ruta.
- **LLEGADA_A_DESTINO:** El robot se detiene al detectar una marca de parada en la línea (`detectaParada`). En este estado, se asegura de que la tapa esté cerrada y ejecuta un giro de 180 grados utilizando el giroscopio (`girarConIMU`) para posicionarse correctamente para el vaciado.

- **ESPERANDO_VACIADO**: El robot espera durante un tiempo predefinido (DURACION_VACIADO_MS) para que el contenedor sea vaciado por un operario. Transcurrido este tiempo, vuelve a medir el nivel de llenado. Si ya no está lleno, inicia su regreso; de lo contrario, espera nuevamente.
- **VIAJE_DE_REGRESO**: Una vez vaciado, el robot emprende el camino de vuelta a su base, siguiendo la misma línea pero en sentido contrario.
- **LLEGADA_A_BASE**: Al detectar la marca de parada en la base (detectaParada), el robot se detiene y realiza otro giro de 180 grados para volver a su orientación original. Al finalizar, transita al estado **ESPERANDO_EN_BASE**, completando el ciclo y quedando listo para un nuevo uso.

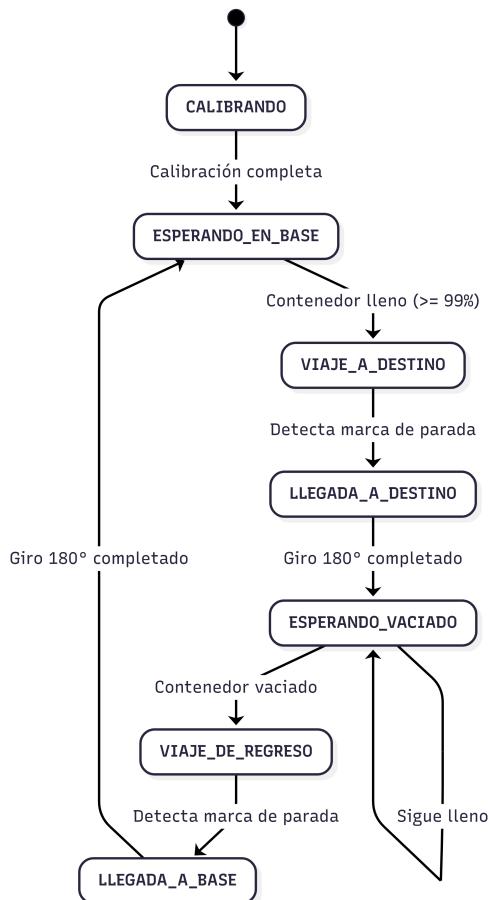


Figura 2: FSM

Arquitectura del Dashboard (SPA Basada en Componentes) La aplicación web (dashboard) se desarrolla con React como una Single Page Application (SPA) basada en componentes. En este proyecto, la lógica principal se concentra en el componente App.tsx, que maneja el estado global, la obtención de datos en tiempo real desde Firebase y la renderización directa de elementos como un medidor de nivel (usando un gauge de Chart.js), gráficos de línea para el nivel de llenado diario y actividad reciente, un gráfico de barras para

la actividad semanal, y paneles de estado para indicadores como el estado de la tapa, el robot y otros sensores. Esta estructura centralizada facilita la integración de datos en tiempo real y la mantenibilidad, aunque podría beneficiarse de refactorizaciones futuras para componentes más modulares, lo cual mejora la escalabilidad del código.

Estructura de Datos en Firebase: Los datos se organizan en el árbol JSON de Firebase Realtime Database de la siguiente manera:

```
1 {
2     "sensor": {
3         "currentStatus": {
4             "calibrado": true,
5             "enMovimiento": false,
6             "estaLleno": false,
7             "estado": "ESPERANDO_EN_BASE",
8             "giroCalibrado": true,
9             "personaDetectada": false,
10            "porcentajeLlenado": 3.84615,
11            "tapaAbierta": false,
12            "timestamp": 1753633565
13        },
14        "historialLlenado": {
15            "-OWBh4q9X32-1okz0ifV": {
16                "porcentajeLlenado": 0,
17                "timestamp": 1753633545
18            }
19        }
20    }
21 }
```

Clave	Descripción	Tipo	Ejemplo
sensor	Nodo principal que contiene toda la información del dispositivo sensor.	Objeto	{...}
currentStatus	Estado actual del sensor capturado en tiempo real.	Objeto	{...}
calibrado	Indica si el sensor ha sido correctamente calibrado.	Booleano	true
enMovimiento	Verdadero si el contenedor o sensor está en movimiento físico.	Booleano	false
estaLleno	Señala si el sensor detecta que el contenedor está lleno.	Booleano	false
estado	Estado lógico o de operación del sensor o sistema.	Texto	ESPERANDO EN BASE
giroCalibrado	Muestra si el mecanismo de giro está calibrado.	Booleano	true
personaDetectada	Indica si hay una persona cerca (sensor de proximidad).	Booleano	false
porcentajeLlenado	Porcentaje actual de llenado detectado por el sensor.	Decimal	3.84615
tapaAbierta	Determina si la tapa del contenedor está abierta.	Booleano	false
timestamp	Marca temporal UNIX del estado actual registrado.	Entero	1753633565
historialLlenado	Objeto que contiene registros históricos de llenado.	Objeto	{...}
-ID-llenado	ID único de un registro individual de llenado.	Texto	-OWBh4q9X32-1okz0ifV
porcentajeLlenado	Nivel de llenado en el momento del registro histórico.	Decimal	0
timestamp	Marca de tiempo UNIX asociada a ese registro histórico.	Entero	1753633545

Cuadro 2: Diccionario de Datos del Objeto JSON del Sensor Inteligente

2.4. Objetivos

2.4.1. Objetivo General

Diseñar y construir un robot móvil automatizado que funcione como contenedor de residuos, capaz de enviar información de llenado a un dashboard web en tiempo real y desplazarse de forma guiada hacia un punto de acopio cuando esté lleno.

2.4.2. Objetivos Específicos

1. Automatizar la apertura y cierre de la tapa del contenedor mediante un sensor de proximidad y un servomotor.
2. Monitorear el nivel de llenado del contenedor utilizando un sensor laser y mediante un dashboard web en tiempo real a través de Firebase.
3. Implementar un controlador Proporcional-Integral-Derivativo para el robot basurero.
4. Elaborar la documentación técnica del proyecto, incluyendo diagramas, código fuente comentado y el presente informe.

2.5. Alcance y Limitaciones

Para definir claramente los límites del prototipo, se especifica el siguiente alcance:

▪ **Funcionalidades Incluidas:**

- Apertura y cierre automático de la tapa.
- Monitoreo del nivel de llenado y visualización en un dashboard online.
- Navegación autónoma desde su estación hasta un punto de acopio predefinido cuando está lleno.
- Espera de 20s en el punto de acopio hasta recibir una señal externa.
- Navegación autónoma de regreso a su estación una vez confirmada la descarga.
- Dashboard Online.

▪ **Funcionalidades Excluidas:**

- El proyecto **no incluirá** un mecanismo de descarga o vaciado automático del contenedor. El proceso de vaciado será **realizado manualmente** por un operario una vez que el robot llegue al punto de acopio. La interacción termina cuando el operario confirma, mediante un pulsador, que el robot está vacío.

3. Marco Teórico

3.1. Estado del Arte

El concepto de "basureros inteligentes"(Smart Bins) ha sido explorado en diversas investigaciones [17, 18, 19, 20].

Trabajos como el de Venkadesh, Divya y Vishal (2024) proponen sistemas estáticos que, si bien automatizan la apertura y notifican su estado de llenado, aún dependen del modelo tradicional de recolección donde una persona debe acudir al sitio del contenedor. Nuestro proyecto representa una evolución de este concepto al introducir la movilidad autónoma como parte integral de la logística de recolección. La contribución fundamental no es solo la "inteligencia" para monitorear, sino la capacidad física de actuar sobre esa información, cerrando el ciclo al trasladarse autónomamente al punto de vaciado. Esto lo diferencia conceptualmente de los sistemas estáticos y lo posiciona como una solución logística, no solo como un dispositivo sensorizado.

Cuadro 3: Análisis comparativo de proyectos relacionados con gestión inteligente de residuos

Autor y Año	Propuesta Tecnológica	Resultados Obtenidos	Limitaciones	Diferencias con el presente proyecto
Perumal et al. (2024)	Basurero inteligente con ESP32 y sensores ultrasónicos. Comunicación vía Wi-Fi.	Automatización básica de apertura y cierre de tapa. Comunicación de llenado en tiempo real.	Sin sistema de navegación ni control autónomo del movimiento.	Nuestro prototipo incluye navegación autónoma hasta el punto de acopio.
Pavithra et al. (2023)	Sistema IoT completo con sensores y módulo GSM.	Monitoreo remoto eficiente del nivel de basura en múltiples ubicaciones.	No cuenta con autonomía física (sin robot móvil).	Nuestro sistema se desplaza físicamente hasta el centro de acopio.
Manara et al. (2024)	Simulación multi-agente para optimizar rutas de recolección con IoT.	Reducción del 30% en rutas y costos operativos mediante algoritmos predictivos.	Solo modelo simulado, sin prototipo físico.	Nuestro prototipo es real y no requiere rutas de camiones.
SciTePress (2024)	Ecobot autónomo con separación y recolección mediante visión computacional e IoT.	Clasificación eficiente y navegación autónoma en entornos urbanos.	Costos elevados y alta complejidad técnica.	Nuestro enfoque es más simple y se centra en recolección autónoma.
IJERT (2022)	Diseño de sistema básico de monitoreo con sensores IR, Arduino y comunicación RF.	Registro local de llenado del contenedor con alertas simples.	Falta escalabilidad, sin base IoT robusta.	Nuestro sistema usa Firebase y visualización en tiempo real.
Brahmanandam et al. (2024)	Contenedor con datos enviados a Firebase para control remoto y visualización.	Datos en tiempo real en dashboard propio.	No incluye desplazamiento autónomo.	Integramos línea de seguimiento y desplazamiento físico autónomo.
Venkadesh, Divya y Vishal (2024)	Sistema de monitoreo de basura con panel web y sensores, basado en nodos estáticos.	Medición precisa en tiempo real.	Sistema estático, sin movilidad física.	Nuestro diseño propone movilidad física para recolección, no solo sensores.
Aiswarya et al. (2023)	Prototipo que monitorea el llenado del contenedor con sensores y envía alertas.	Alertas enviadas mediante GSM.	No tiene desplazamiento ni control autónomo.	Nuestro sistema incluye control PID y desplazamiento físico autónomo.

La revisión evidencia que si bien existen múltiples esfuerzos previos en el diseño de sistemas inteligentes para la gestión de residuos, el presente proyecto destaca por integrar funcionalidades avanzadas como la navegación autónoma asistida por PID, el monitoreo en tiempo real vía Firebase y la interacción física automatizada mediante sensores y actuadores. Estas características representan un avance significativo frente a propuestas similares, tanto en términos de autonomía operativa como de escalabilidad.

3.2. Tecnologías de Implementación

3.2.1. Hardware

Microcontrolador ESP32: Un sistema en chip (SoC) de bajo costo y alto rendimiento, basado en la arquitectura Xtensa LX6 de 32 bits. Opera con una CPU de doble núcleo que puede alcanzar hasta 240 MHz, e integra conectividad Wi-Fi 802.11 b/g/n y Bluetooth v4.2.

Característica	Descripción
Microcontrolador	Tensilica Xtensa LX6 de doble núcleo a 240 MHz
Voltaje de operación	3.0V a 3.3V (los pines GPIO no son tolerantes a 5V)
Memoria RAM	520 KB SRAM interna
Wi-Fi	802.11 b/g/n (2.4 GHz) con soporte para modo STA/AP/STA+AP
Bluetooth	Bluetooth 4.2 y Bluetooth Low Energy (BLE)
Entradas analógicas	Hasta 18 canales ADC de 12 bits
Salidas analógicas	2 canales DAC de 8 bits
Entradas táctiles	Hasta 10 entradas táctiles capacitivas
Interfaz de comunicación	UART, SPI, I2C, CAN, I2S, PWM
GPIOs	Hasta 34 GPIOs configurables
Timers	Múltiples temporizadores de hardware de 64 bits
Consumo energético	Modo activo (160 mA), modo de sueño profundo (10 µA)
Voltaje de entrada (Vin)	5V (cuando se alimenta por USB o Vin externo)
Temperatura de operación	-40 °C a +125 °C

Cuadro 4: Características técnicas del ESP32



Figura 3: Microcontrolador ESP32

Sensor Ultrasónico (HC-SR04): Dispositivo que determina la distancia a un objeto mediante la emisión de pulsos sonoros de alta frecuencia (típicamente 40 kHz) y la medición del tiempo de vuelo del eco.

Característica	Descripción
Voltaje de operación	5V DC
Rango de medición	2 cm a 400 cm
Precisión	± 3 mm
Frecuencia ultrasónica	40 kHz
Salida	Pulso digital proporcional a la distancia
Aplicaciones	Evitación de obstáculos, medición de distancia

Cuadro 5: Características del sensor HC-SR04



Figura 4: Sensor ultrasónico HC-SR04

Sensor de Distancia Láser (VL53L0X): Sensor basado en la tecnología de Tiempo de Vuelo (ToF). Emite un pulso láser de 940 nm (infrarrojo, invisible) y mide con alta precisión el tiempo que tarda en reflejarse en un objeto y retornar.

Característica	Descripción
Tecnología	Tiempo de vuelo (ToF)
Voltaje de operación	2.6V a 3.5V (compatible con 3.3V/5V vía I2C)
Rango de medición	Hasta 2 metros (según condiciones)
Resolución	1 mm
Interfaz	I2C
Aplicaciones	Medición precisa de distancia, robots, drones

Cuadro 6: Características del sensor VL53L0X

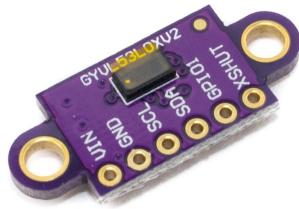


Figura 5: Sensor de distancia láser VL53L0X

Sensor Infrarrojo (Seguidor de Línea TCRT5000): Módulo que consta de un diodo emisor de luz infrarroja y un fototransistor. El principio de funcionamiento se basa en la reflexión de la luz IR.

Característica	Descripción
Tipo	Sensor reflectivo (emisor IR + fototransistor)
Voltaje de operación	3.3V a 5V DC
Rango efectivo	2 mm a 10-15 mm
Salida	Analógica o digital (dependiendo del módulo)
Consumo	5–20 mA
Aplicaciones	Seguidores de línea, detección de bordes

Cuadro 7: Características del sensor TCRT5000



Figura 6: Sensor infrarrojo TCRT5000

Servomotor (SG90): Actuador rotativo que permite un control preciso de la posición angular, controlado mediante una señal de Modulación por Ancho de Pulso (PWM).

Característica	Descripción
Tipo	Servomotor de rotación controlada
Ángulo de giro	0° a 180°
Voltaje de operación	4.8V a 6V
Control	Señal PWM
Torque aproximado	1.8 kg/cm a 4.8V
Aplicaciones	Robótica, brazos mecánicos, control de dirección

Cuadro 8: Características del servomotor SG90



Figura 7: Servomotor SG90

Motores DC: Los motores de corriente directa (DC) convierten energía eléctrica en energía mecánica mediante la interacción entre un campo magnético y una corriente. Son muy utilizados en proyectos de robótica por su simplicidad, bajo costo y facilidad de control.

Característica	Descripción
Tensión de operación	3V a 12V DC
Relación de reducción	Variable (comúnmente 1:30, 1:50, 1:100, 1:150, hasta 1:1000)
Velocidad sin carga	6V a 300 RPM (relación 1:50), 6V a 100 RPM (relación 1:150)
Corriente sin carga	40mA a 100mA (típico a 6V)
Torque	Aprox. 0.5 a 1.2kg · cm (dependiendo de reducción y voltaje)
Diseño	12mm × 10mm × 26mm (cuerpo del motor sin eje)
Aplicaciones	Minicarros seguidores de línea, brazos robóticos, sistemas de engranajes, cerraduras electrónicas, pequeños sistemas mecatrónicos.

Cuadro 9: Características del Micro Motorreductor



Figura 8: Motores DC

Controlador L298N: El L298N es un controlador de tipo puente H dual que permite controlar la velocidad y dirección de dos motores DC de forma independiente. Es ideal para proyectos educativos y robóticos, ya que soporta hasta 2A por canal y puede controlarse fácilmente desde microcontroladores como Arduino.

Característica	Descripción
Voltaje de operación (V_{IN})	5V a 35V DC
Voltaje lógico (V_{logic})	5V
Corriente máxima por canal	2A (con disipador adecuado)
Número de canales	2 (permite controlar 2 motores DC o 1 motor paso a paso)
Salidas (V_{OUT})	Dos pares de salidas para motores A y B
Protección térmica	Sí, integrada en el chip
Control de velocidad	Mediante señal PWM
Aplicaciones	Robótica, control de movimiento, vehículos autónomos

Cuadro 10: Características del driver L298N

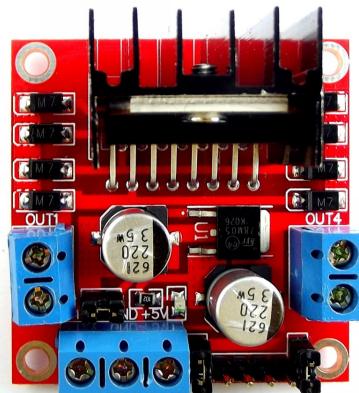


Figura 9: Driver L298N para motores DC

Sensor Inercial (MPU-6500): Sensor MEMS de 6 ejes que combina un acelerómetro y un giroscopio de 3 ejes cada uno. Permite medir aceleración lineal y velocidad angular con alta precisión, ideal para aplicaciones de detección de movimiento.

Característica	Descripción
Tecnología	MEMS (Sistema Microelectromecánico)
Voltaje de operación	2.4V a 3.6V (típico: 3.3V)
Sensores integrados	Acelerómetro de 3 ejes y giroscopio de 3 ejes
Rango de aceleración	$\pm 2g$, $\pm 4g$, $\pm 8g$, $\pm 16g$ (configurable)
Rango de giro	± 250 , ± 500 , ± 1000 , ± 2000 °/s (configurable)
Interfaz	I2C y SPI
Frecuencia de muestreo	Hasta 1 kHz
Aplicaciones	Detección de movimiento, estabilización, navegación inercial, robótica, drones

Cuadro 11: Características del sensor inercial MPU-6500



Figura 10: Sensor MPU-6500

Módulo Regulador de Voltaje (XL6009): Convertidor DC-DC tipo Step-Up (elevador) capaz de aumentar tensiones de entrada entre 3V y 32V hasta un rango de salida ajustable de 5V a 35V. Soporta hasta 4A de corriente con disipación, siendo ideal para alimentar motores, microcontroladores y módulos electrónicos que requieren voltajes estables y superiores.

Característica	Descripción
Tensión de operación	Entrada de 3V a 32V DC
Relación de conversión	Step-Up (elevador); convierte tensiones más bajas en salidas más altas
Voltaje de salida	Ajustable de 5V a 35V DC
Corriente de salida	Hasta 4A (uso continuo recomendado < 3A con disipador)
Diseño	Módulo compacto con potenciómetro para ajuste de voltaje. Dimensiones: 43mm × 21mm × 14mm
Aplicaciones	Fuentes de alimentación reguladas, cargadores portátiles, sistemas con baterías Li-ion, motores DC, microcontroladores

Cuadro 12: Características técnicas del módulo elevador XL6009 Step-Up



Figura 11: Módulo regulador de voltaje XL6009

3.2.2. Software y Plataformas

Firebase Realtime Database: Es un servicio de base de datos NoSQL alojado en la nube que permite almacenar y sincronizar datos entre usuarios en tiempo real. Los datos se almacenan como un único y gran árbol JSON.

Entorno de Desarrollo (PlatformIO): El firmware del ESP32 se desarrolla utilizando PlatformIO, un ecosistema de desarrollo profesional para software embebido. Se eligió sobre el IDE estándar de Arduino por sus ventajas significativas, como la gestión automática de librerías y dependencias (evitando conflictos), la integración con el editor de código Visual Studio Code, y herramientas avanzadas de compilación y depuración que agilizan el ciclo de desarrollo.

3.3. Tecnologías y Arquitecturas de Software

3.3.1. Patrones de Arquitectura de Software

Single Page Application (SPA) es un tipo de arquitectura de aplicación web que opera dentro de una única página HTML, donde todo el contenido necesario se carga inicialmente o de manera incremental sin necesidad de recargar toda la interfaz durante la navegación. A diferencia de las aplicaciones tradicionales de múltiples páginas (MPA), las SPA utilizan tecnologías como JavaScript, AJAX, y frameworks como React, Vue o Angular para gestionar dinámicamente las vistas, enrutamiento interno y comunicación con servicios backend mediante APIs REST o GraphQL. Este enfoque permite una experiencia de usuario más fluida y rápida, simulando el comportamiento de una aplicación nativa, ya que las transiciones entre secciones no requieren cargas completas del servidor.

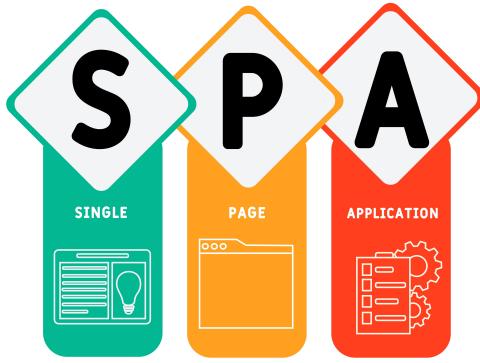


Figura 12: Solicitud de una sola página (SPA)

3.3.2. Tecnologías del Frontend

React: Biblioteca de JavaScript para construir interfaces de usuario interactivas y reutilizables mediante componentes. Facilita el desarrollo de aplicaciones web modernas y responsivas.

TypeScript: Superconjunto de JavaScript que añade tipado estático. Mejora la robustez del código y facilita el mantenimiento, especialmente en aplicaciones a gran escala.

Vite: Herramienta de construcción y desarrollo rápido que permite compilar archivos `.tsx` a JavaScript. Integra un servidor de desarrollo eficiente y genera paquetes optimizados para producción.

Chart.js: Biblioteca JavaScript para la creación de gráficos dinámicos y visualizaciones interactivas en el dashboard. Compatible con múltiples tipos de gráficos como barras, líneas, radar y pastel.

3.4. Modelo Cinemático y Control

Robot Móvil Diferencial: El chasis del robot se basa en una configuración de tracción diferencial, con dos ruedas coaxiales accionadas por motores independientes, lo que permite una alta maniobrabilidad.

Modelo Cinemático Directo: Describe el movimiento del robot en función de la velocidad de sus ruedas. Dado un robot con dos ruedas de radio r , separadas por una distancia L , y con velocidades angulares ω_r (derecha) y ω_l (izquierda), la velocidad lineal v y la velocidad angular ω del centro del robot se definen por:

$$v = \frac{r(\omega_r + \omega_l)}{2} \quad (1)$$

$$\omega = \frac{r(\omega_r - \omega_l)}{L} \quad (2)$$

Control de Navegación (Algoritmo PID): Para un seguimiento de línea suave, se implementa un controlador Proporcional-Integral-Derivativo (PID). Este algoritmo de control de lazo cerrado calcula un error (desviación del robot respecto a la línea) y ajusta la velocidad de los motores para corregirlo. La salida del controlador $u(t)$ es:

$$\omega = (K_p \cdot \text{error}) + (K_i \cdot \sum \text{error}) + (K_d \cdot (\text{error}_{\text{actual}} - \text{error}_{\text{anterior}})) \quad (3)$$

Esquema del funcionamiento del controlador PID para el seguimiento de línea

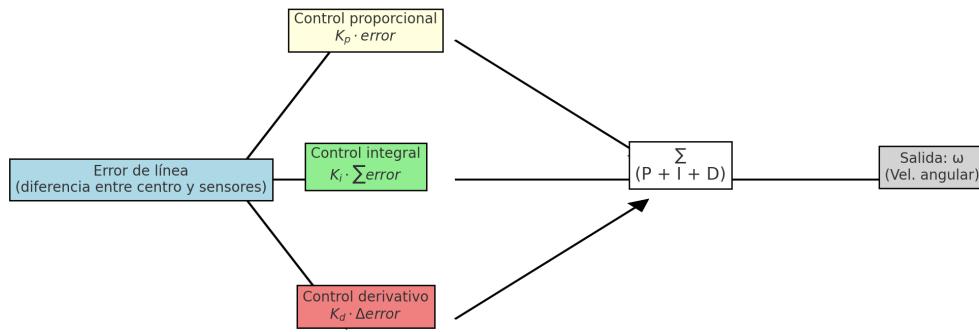


Figura 13: Control PID

3.5. Protocolos de Comunicación

3.5.1. Pila de Protocolos TCP/IP

La comunicación del dispositivo se basa en la pila TCP/IP estándar. El ESP32 se conecta a la red local a través de Wi-Fi (IEEE 802.11), obteniendo una dirección IP mediante DHCP.

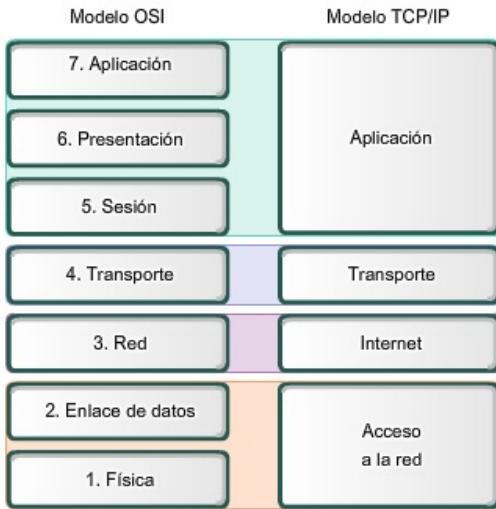


Figura 14: Protocolo TCP/IP

3.5.2. HTTPS (Hypertext Transfer Protocol Secure)

Es el protocolo principal utilizado para la comunicación entre el ESP32 y Firebase Realtime Database. La librería FirebaseESP32.h encapsula solicitudes RESTful (como PUT y POST) sobre una conexión TLS/SSL, garantizando que los datos viajen de forma encriptada y segura.

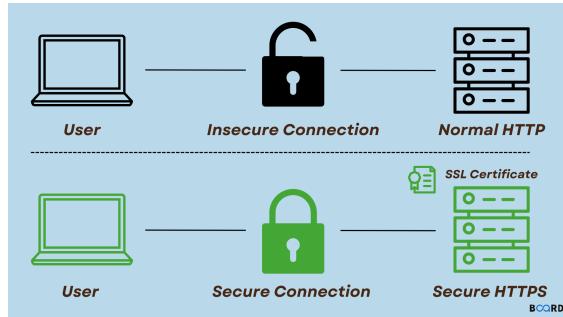


Figura 15: Protocolo HTTPS

3.5.3. NTP (Network Time Protocol)

Para asegurar la integridad y el orden cronológico de los datos históricos, el sistema utiliza el protocolo NTP. Al iniciar, el ESP32 se conecta a un servidor de tiempo público (ej. pool.ntp.org) para sincronizar su reloj interno.

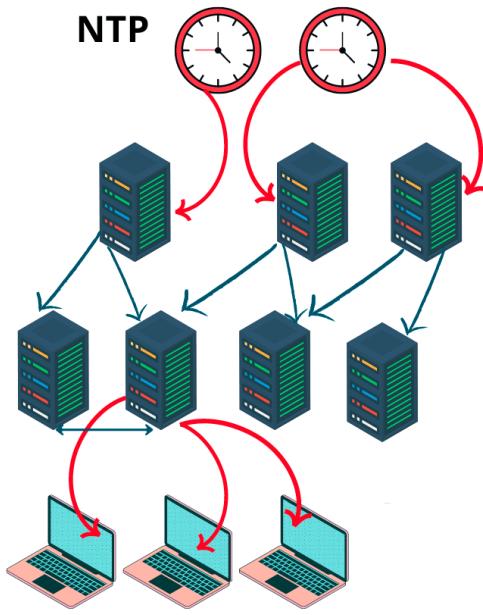


Figura 16: Protocolo NTP

3.5.4. WebSocket para Comunicación en Tiempo Real

Para la comunicación entre la base de datos Firebase y el dashboard, se emplea la tecnología WebSocket, gestionada automáticamente por el SDK de Firebase para JavaScript. A diferencia de HTTP, WebSocket establece un canal de comunicación bidireccional y persistente entre el cliente (navegador web) y el servidor. Sus beneficios en este proyecto son:

Comunicación en Tiempo Real (Push): Cuando el ESP32 actualiza un dato, el servidor de Firebase lo empuja inmediatamente a través del WebSocket al dashboard, eliminando la necesidad de que el usuario recargue la página.

Baja Latencia y Eficiencia: Al mantener una conexión abierta, se reduce significativamente la latencia y el tráfico de red asociado a múltiples peticiones HTTP, resultando en una experiencia de usuario fluida y reactiva.

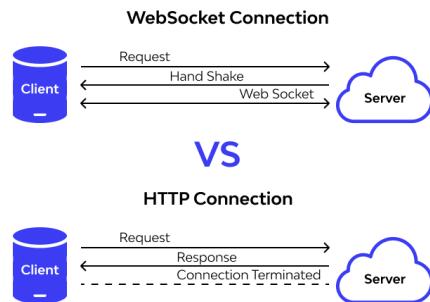


Figura 17: WebSocket para Comunicación en Tiempo Real

4. Diseño de la Propuesta

Para definir el alcance y los criterios de éxito del proyecto, se ha elaborado una lista detallada de requisitos funcionales, que describen qué debe hacer el sistema, y requisitos no funcionales, que definen cómo debe comportarse en términos de calidad y rendimiento. La siguiente tabla establece las métricas específicas que se utilizarán para validar el cumplimiento de cada requisito durante la fase de pruebas.

4.1. Requerimientos funcionales

Cuadro 13: Requisitos Funcionales (RF) y Métricas de Éxito

Código	Descripción del Requisito	Métrica de Cumplimiento
Subsistema: Gestión de Tapa y Residuos		
RF-01	Apertura de tapa por proximidad: El sistema debe detectar a un usuario y abrir la tapa automáticamente.	La tapa se abrirá al ángulo definido (80°) en menos de 1.5s cuando un objeto sea detectado a una distancia 36 cm. Se requiere una tasa de éxito $\geq 95\%$.
RF-02	Cierre automático de tapa: Tras la detección, si el usuario se aleja, la tapa debe cerrarse automáticamente.	La tapa iniciará su cierre 2 segundos después de que el sensor de proximidad deje de detectar al usuario.
RF-03	Medición de nivel de llenado: El sistema debe medir continuamente el nivel de residuos.	El dato del porcentaje de llenado se actualizará internamente cada 2 segundos, con un error máximo de $\pm 5\%$ respecto a la altura real.
Subsistema: Navegación Autónoma		
RF-04	Transición a modo autónomo: Al llenarse, el sistema debe iniciar el ciclo de navegación.	El robot activará el estado VIAJE_A_DESTINO cuando porcentajeLlenado sea 99% .
RF-05	Navegación guiada por línea: El robot debe ser capaz de seguir la trayectoria designada.	El robot debe completar el circuito de prueba sin desviarse en al menos 8 de 10 intentos (Tasa de éxito del 80%).
RF-06	Maniobra de llegada y giro: Al llegar al destino, el robot debe detenerse y girar 180° .	El giro de 180° debe completarse con una precisión de $\pm 5^\circ$.
RF-07	Ciclo de retorno a base: Tras el vaciado, el robot debe regresar a su punto de origen.	El robot iniciará el viaje de regreso tras 20 segundos de espera y se repositionará en su base.
Subsistema: Dashboard y Monitoreo Remoto		
RF-08	Visualización de estado en tiempo real: El dashboard web debe mostrar el estado actual del robot.	Los datos en el dashboard se actualizarán con una latencia máxima de 5 segundos.
RF-09	Visualización de historial: El sistema debe mostrar datos históricos de llenado.	El dashboard debe renderizar un gráfico con los registros históricos almacenados en Firebase.

4.2. Requerimientos no funcionales

Cuadro 14: Requisitos No Funcionales (RNF) y Métricas de Calidad

Código	Atributo de Calidad	Métrica de Cumplimiento
RNF-01	Rendimiento	El sobreimpulso en el seguimiento de línea debe ser inferior al 15 % para garantizar un movimiento estable.
RNF-02	Fiabilidad	El sistema debe operar de manera continua durante 1 hora sin fallos ni reinicios inesperados.
RNF-03	Usabilidad	La operación de depósito de residuos debe ser 100 % sin contacto físico.
RNF-04	Usabilidad	La interfaz del dashboard debe ser clara y fácilmente interpretable por un usuario no técnico.
RNF-05	Mantenibilidad	El código fuente debe estar estructurado y comentado para facilitar futuras modificaciones.
RNF-06	Seguridad	La comunicación con la base de datos en la nube debe realizarse exclusivamente mediante un protocolo cifrado (HTTPS).
RNF-07	Eficiencia	El sistema debe utilizar componentes de bajo consumo para maximizar la autonomía.

4.3. Arquitectura del sistema

La solución propuesta se basa en una arquitectura de tres capas: física, de datos y de presentación. Esta estructura modular permite la interacción eficiente entre el hardware (sensores y actuadores), el almacenamiento en la nube (Firebase) y una interfaz web amigable desarrollada en React. Gracias a esta integración, el sistema puede operar en tiempo real, automatizar decisiones y brindar al usuario información clara y actualizada sobre el estado del robot y la gestión de residuos.

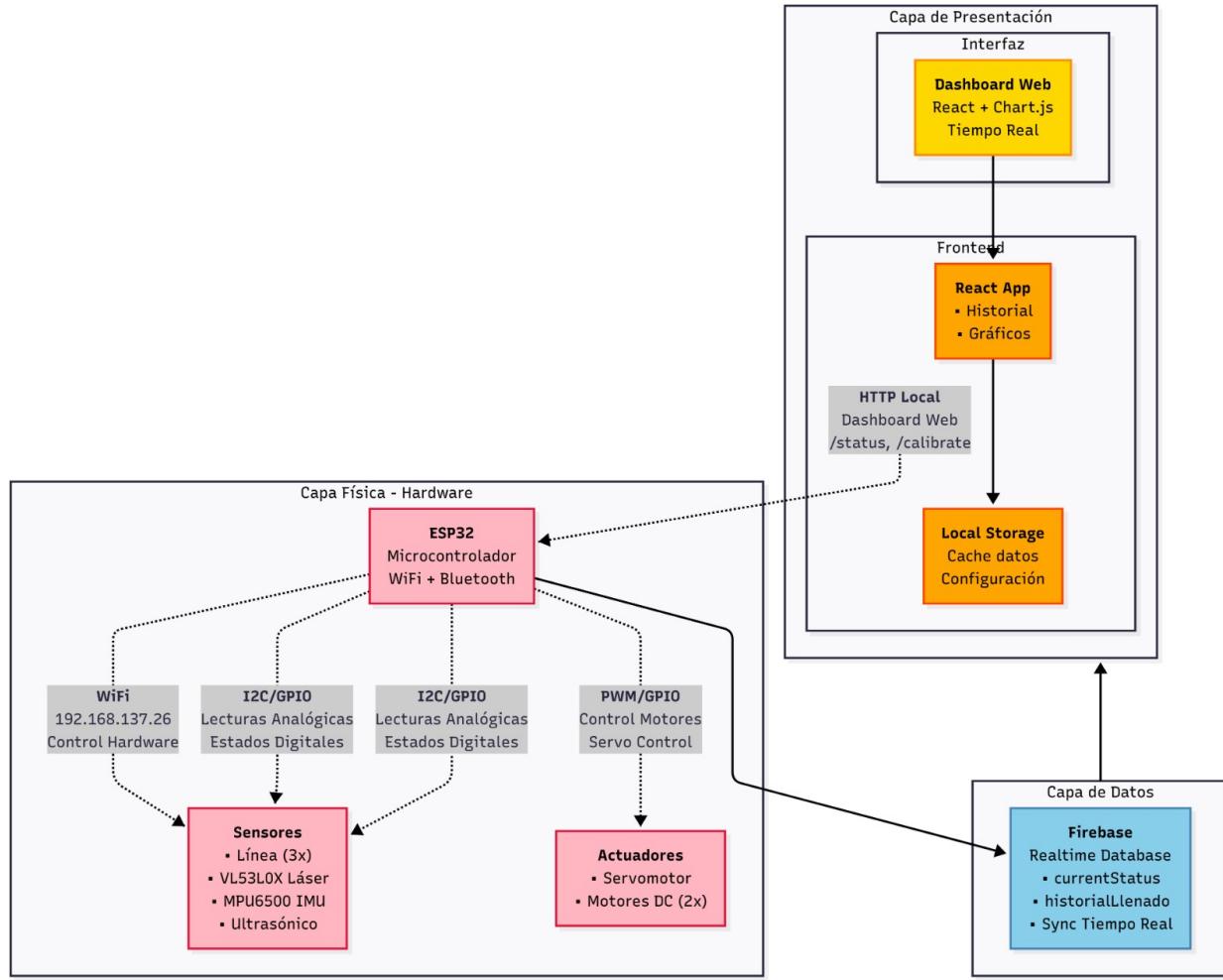


Figura 18: Esquema de Arquitectura de Sistema

4.4. Implementación

4.4.1. Apertura automática de la tapa

Para el sistema de apertura automatizada se instaló un sensor ultrasónico HC-SR04 adelante del tacho, encargado de detectar la presencia de los usuarios. Al activarse, el sistema acciona el servomotor que abre la tapa del basurero de forma automática. Además, la incorporación de un sensor de alcance en el interior monitorea el nivel de llenado, optimizando la recolección y evitando desbordes.



Figura 19: Esquema de Arquitectura del Sistema.

4.4.2. Implementación del chasis y navegación

Se integró el microcontrolador ESP32 y los sensores sobre un chasis de dos ruedas motrices y una loca. Los cuales activan la navegación autónoma hacia el punto de acopio cuando el sensor de alcance detecta que el contenedor está lleno. La estructura asegura estabilidad y una distribución eficiente de los componentes.



Figura 20: Chasis con sensores y ESP32 integrados

4.4.3. Implementación de sensores IR para navegación

En la base del chasis se instalaron sensores infrarrojos (IR) encargados de detectar contrastes entre la línea negra del recorrido y el fondo claro. Estos sensores permiten al robot seguir trayectorias predefinidas mediante un algoritmo de control PID, activando los motores de forma diferencial según la desviación detectada. Su disposición frontal garantiza una lectura precisa durante el desplazamiento autónomo.

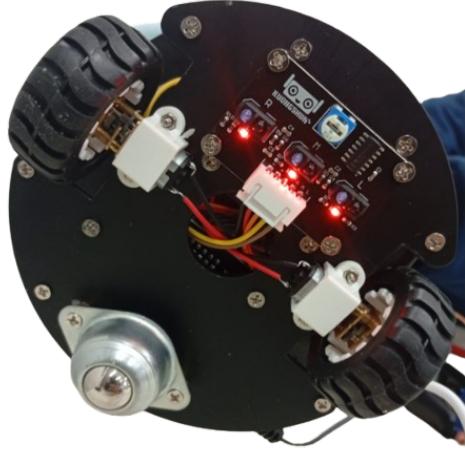


Figura 21: Sensores infrarrojos montados para la detección de línea.

4.4.4. Modelo Cinemático del Robot Diferencial

El robot utiliza una configuración de tracción diferencial, donde la velocidad y dirección se controlan variando la velocidad de rotación de cada una de sus dos ruedas motrices de forma independiente. El modelo cinemático relaciona la velocidad lineal del robot (V) y su velocidad angular (ω) con las velocidades lineales individuales de la rueda izquierda (v_L) y la rueda derecha (v_R).

La velocidad angular ω es la variable clave para corregir la trayectoria. Si ω es positiva, el robot gira a la izquierda; si es negativa, gira a la derecha. La relación se define por las siguientes ecuaciones de cinemática inversa:

$$v_L = V - \frac{L \cdot \omega}{2} \quad (4)$$

$$v_R = V + \frac{L \cdot \omega}{2} \quad (5)$$

Donde:

- V es la velocidad lineal base del robot, definida en el código como `BASE_LINEAR_VELOCITY`.
- ω es la velocidad angular correctiva, calculada por el controlador PID.
- L es la distancia entre las dos ruedas motrices (`DISTANCE_BETWEEN_WHEELS`).

En nuestro sistema, el controlador PID se encarga de calcular la ω necesaria para mantener el robot en la línea, y estas ecuaciones la traducen en las velocidades que cada rueda debe tener.

Código de la cinemática implementado:

Listing 1: Cálculo de velocidades de ruedas basado en cinemática diferencial

```

1 float vL = BASE_LINEAR_VELOCITY - (DISTANCE_BETWEEN_WHEELS / 2.0) * omega;
2 float vR = BASE_LINEAR_VELOCITY + (DISTANCE_BETWEEN_WHEELS / 2.0) * omega;

```

Estas expresiones, directamente extraídas del código fuente, implementan el modelo de cinemática diferencial. La variable `omega` representa la salida del controlador PID, es decir, la corrección angular necesaria para seguir la línea negra. Dicho valor se traduce en velocidades lineales diferenciadas para las ruedas izquierda y derecha, según la distancia entre ruedas (`DISTANCE_BETWEEN_WHEELS`), asegurando así la navegación precisa del robot.

4.4.5. Controlador PID para Seguimiento de Línea

Para que el robot siga la línea de forma autónoma, se implementa un controlador Proporcional-Integral-Derivativo (PID). El propósito de este controlador es calcular la velocidad angular correctiva (ω) necesaria para minimizar el error de posición del robot con respecto a la línea.

Cálculo del Error El error se determina a partir de las lecturas de los tres sensores infrarrojos. Se le asigna un valor numérico discreto según la posición del robot:

- Error = -2 o -1: El robot está desviado a la derecha de la línea.
- Error = 0: El robot está centrado sobre la línea.
- Error = 1 o 2: El robot está desviado a la izquierda de la línea.

Fórmula PID Discreta Dado que el microcontrolador opera en tiempo discreto, se utiliza la versión discreta de la fórmula PID para calcular la salida del controlador, ω :

$$\omega = (K_p \cdot \text{error}) + (K_i \cdot \sum \text{error}) + (K_d \cdot (\text{error}_{\text{actual}} - \text{error}_{\text{anterior}})) \quad (6)$$

Este cálculo se corresponde directamente con el siguiente fragmento de la función `seguirLineaPID()`:

```

integral += error;
int derivative = error - lastError;
float omega = Kp * error + Ki * integral + Kd * derivative;
lastError = error;

```

La ω resultante es la entrada para el modelo cinemático, que determinará las velocidades v_L y v_R .

4.4.6. Calibración de Motores: De Velocidad a PWM

El modelo cinemático y el PID operan con unidades de velocidad (mm/s), pero los motores se controlan con señales de Modulación por Ancho de Pulso (PWM). Se necesita un modelo que traduzca la velocidad deseada a un valor PWM. Para ello, se realizó una calibración empírica para obtener una función lineal: $\text{PWM} = m \cdot \text{Velocidad} + c$.

Obtención del Modelo Lineal Se tomaron mediciones para definir la recta, registrando la velocidad obtenida a diferentes valores de PWM en una distancia de 600 mm. Los puntos clave para el cálculo fueron:

Cuadro 15: Puntos de Datos para la Calibración Velocidad-PWM

Prueba	PWM Aplicado	Tiempo (s)	Distancia (mm)	Velocidad (mm/s)
1	100	14.00	600	42.86
2	200	6.54	600	91.74

Estos datos corresponden a los puntos (42.86, 100) y (91.74, 200) en un plano (Velocidad, PWM).

Cálculo de la Pendiente (m) La pendiente (m) representa el aumento de PWM necesario por cada mm/s de velocidad.

$$m = \frac{\Delta \text{PWM}}{\Delta \text{Velocidad}} = \frac{200 - 100}{91,74 - 42,86} = \frac{100}{48,88} \approx 2,046 \quad (7)$$

Este valor se define como la constante `VEL_TO_PWM_SLOPE`.

Cálculo del Intercepto (c) El intercepto (c) es el PWM teórico necesario para vencer la fricción e iniciar el movimiento. Usando el primer punto:

$$c = \text{PWM}_1 - (m \cdot \text{Velocidad}_1) = 100 - (2,046 \cdot 42,86) \approx 12,31 \quad (8)$$

Este valor se define como la constante `VEL_TO_PWM_INTERCEPT`.

Resultado Final Con este modelo, el firmware convierte las velocidades v_L y v_R calculadas por el modelo cinemático en los valores PWM que se envían a los motores, cerrando el ciclo de control.

$$\text{PWM}_{\text{calculado}} = (2,046 \cdot \text{Velocidad}_{\text{deseada}}) + 12,31 \quad (9)$$

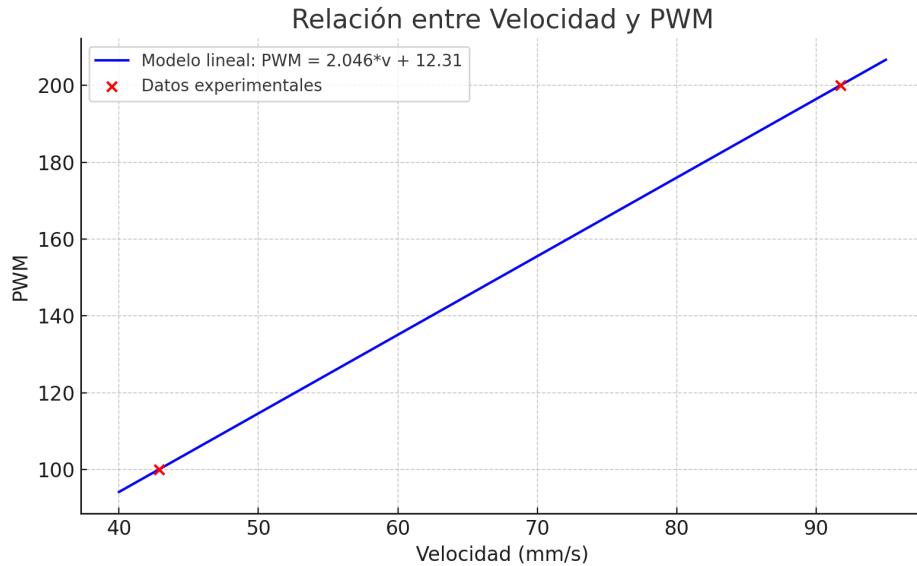


Figura 22: Relación entre velocidad y PWM

Conversión de Velocidad a PWM en Código: Para convertir la velocidad lineal calculada por el modelo cinemático en señales PWM aplicables a los motores, se utilizó la función lineal obtenida. Los siguientes parámetros se definen en el código:

```
const float VEL_TO_PWM_SLOPE = 2.046;
const float VEL_TO_PWM_INTERCEPT = 12.31;
```

Luego, en la función de seguimiento de línea PID, se realiza la conversión:

```
int pwmLeft  = constrain((int)(vL * VEL_TO_PWM_SLOPE + VEL_TO_PWM_INTERCEPT),
0, MAX_SPEED);
int pwmRight = constrain((int)(vR * VEL_TO_PWM_SLOPE + VEL_TO_PWM_INTERCEPT),
0, MAX_SPEED);
```

Este fragmento garantiza que las velocidades físicas calculadas se traduzcan en señales PWM compatibles con los controladores de motor, permitiendo el movimiento real del robot.

4.4.7. SPA Dashboard - Monitoreo

: El robot envia datos al servidor Firebase, Este es leido y mostrado por una web mediante un dashboard en la cual se puede ver el nivel de llenado del basurero, así mismo un historial de llenado, detección de personas, el estado del carro y si la tapa se encuentra abierta o cerrada.

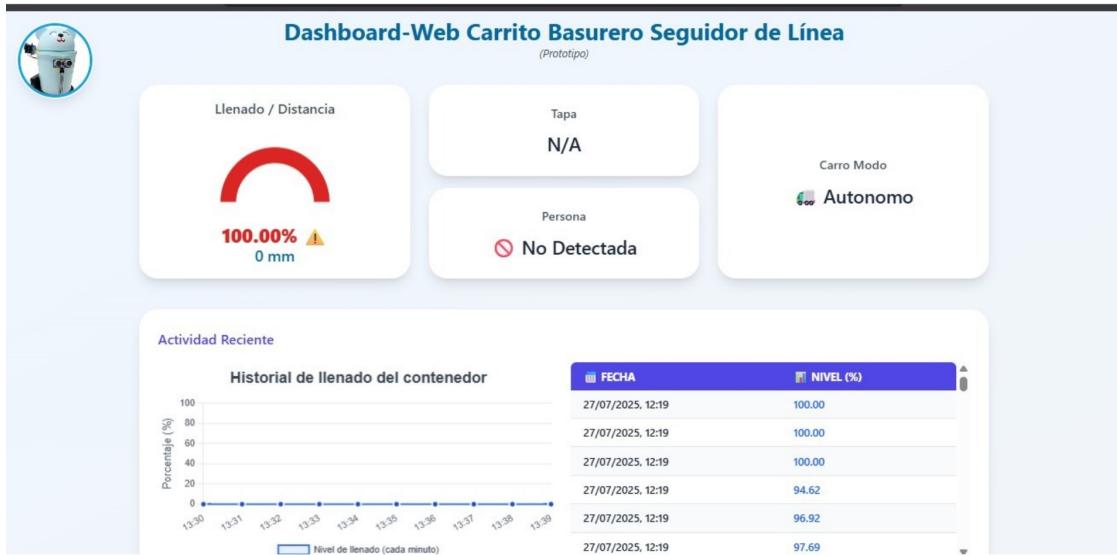


Figura 23: Panel de monitoreo desde la interfaz web integrada.

5. Pruebas y Resultados

Para la validación del prototipo, se diseñó un conjunto de experimentos específicos para verificar el cumplimiento de cada objetivo, asegurando la correcta operación tanto de los componentes individuales como del sistema integrado.

5.1. Pruebas Experimentales

Se describen a continuación las metodologías de prueba aplicadas para cada subsistema.

5.1.1. Validación del Sistema de Tapa Automática (Objetivo 1)

- **Prueba de Calibración de Ángulos del Servomotor:** Se ejecutó un script de prueba para mover el servomotor SG90 en incrementos de 5 grados, desde 0° hasta 90°. Se observó visualmente el comportamiento mecánico de la tapa para identificar los ángulos exactos que correspondían a las posiciones “completamente cerrada” y “completamente abierta” sin forzar el motor ni el mecanismo.



Figura 24: Calibración del servomotor

- **Prueba de Fiabilidad del Sensor de Proximidad:** Se realizaron 30 ensayos en los que un operario se aproximaba al robot desde diferentes ángulos (frontal, lateral) y a distintas velocidades. El objetivo era verificar la activación consistente del sensor ultrasónico HC-SR04 y el umbral de distancia de 36 cm definido en el firmware.

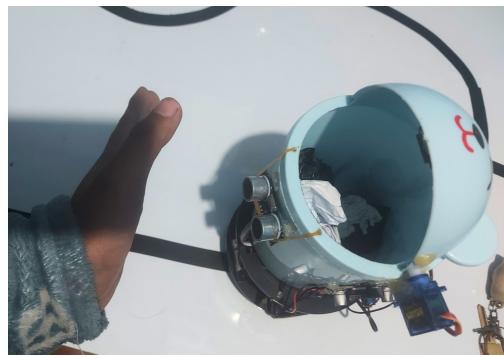


Figura 25: Prueba de proximidad de basura

5.1.2. Validación del Monitoreo de Llenado (Objetivo 2)

- **Prueba de Precisión del Sensor ToF:** El contenedor se llenó progresivamente con objetos de volumen conocido (bolas de papel arrugado) en incrementos del 20 % de su capacidad total. En cada nivel, se tomaron 5 mediciones con el sensor VL53L0X para promediarlas y se compararon con la distancia real medida manualmente con una cinta métrica. Esto permitió calcular el error porcentual del sensor.

- **Prueba de Latencia del Dashboard:** Con el sistema en línea, se provocaron cambios de estado (ej. depositar basura, acercarse al robot) y se cronometró el tiempo transcurrido hasta que la actualización correspondiente (cambio en el porcentaje de llenado, estado de la tapa) se reflejaba en el dashboard web.

5.1.3. Validación de la Navegación Autónoma (Objetivo 3)

- **Prueba de Calibración de Sensores de Línea:** Se ejecutó la rutina de calibración del firmware, exponiendo los tres sensores TCRT5000 primero a una superficie blanca y luego a la línea negra de la pista, para establecer los umbrales de detección óptimos.
- **Prueba de Calibración Velocidad-PWM:** Se midió el tiempo que tardaba el robot en recorrer una distancia fija (600 mm) con diferentes valores de PWM (100 y 200). A partir de estos datos (distancia y tiempo), se calculó la velocidad lineal real (mm/s) para cada nivel de PWM, lo que permitió establecer el modelo de regresión lineal para la conversión.
- **Prueba de Sintonización del Controlador PID:** Se utilizó un método de sintonización manual iterativo en la pista de pruebas. Se comenzó con K_i y K_d en cero y se ajustó K_p hasta obtener una respuesta oscilatoria. Luego, se ajustó K_d para amortiguar las oscilaciones y, finalmente, se introdujo un valor pequeño de K_i para corregir errores de estado estacionario. Se documentaron los comportamientos para diferentes combinaciones de constantes.
- **Prueba de Precisión de Giro (IMU):** Se colocó el robot en una marca de inicio y se activó la rutina de giro de 180°. Se utilizó un transportador en el suelo para medir el ángulo final real del robot y compararlo con el objetivo, repitiendo la prueba 10 veces para evaluar la consistencia.

5.2. Resultados Obtenidos

Los datos recopilados en las pruebas validan la efectividad del diseño.

- **Calibración de Actuadores y Sensores:**
 - Los ángulos óptimos para el servomotor se establecieron en **5° para cerrado** y **80° para abierto**, valores que se fijaron en el firmware como `SERVO_CERRADO` y `SERVO_ABIERTO`.
 - La prueba de fiabilidad del sensor de proximidad arrojó una **tasa de éxito del 96.7%** (29 de 30 activaciones correctas). El único fallo ocurrió por una aproximación excesivamente rápida y oblicua.
 - El sensor ToF para el nivel de llenado mostró un **error medio de $\pm 3\%$** , considerado altamente aceptable para esta aplicación. La latencia del dashboard fue consistentemente **inferior a 3 segundos**.
- **Rendimiento del Control PID:** El proceso de sintonización fue crucial. La Tabla 16 muestra un resumen del proceso. Las constantes finales que proporcionaron el

mejor rendimiento (seguimiento suave, sin oscilaciones y rápida recuperación) fueron **K_p=2.0, K_i=0.001, K_d=6.0**.

Cuadro 16: Proceso de Sintonización de Constantes PID

K_p	K_i	K_d	Comportamiento Observado
1.0	0	0	Lento, se sale en curvas cerradas.
3.5	0	0	Reacciona rápido pero oscila violentamente (inestable).
2.0	0	4.0	Oscilaciones reducidas, pero aún sobrepasa la línea.
2.0	0.001	6.0	Óptimo: Seguimiento estable, suave y preciso.

Con las constantes óptimas, se realizaron 25 pruebas de navegación completas en el circuito de 2 metros, obteniendo una **tasa de éxito del 100 %**, como se detalla en la Tabla 17. La precisión del giro con la IMU MPU-6500 registró una desviación promedio de solo $\pm 4^\circ$ sobre el objetivo de 180° .

Cuadro 17: Resultados Finales de Navegación del Controlador PID

Iteración	Resultado	Observaciones
1–5	Fallo Parcial	El robot presentaba oscilaciones y se desviaba en curvas (fase de sintonización inicial).
6–30	Éxito	Seguimiento estable y preciso de la línea. El robot navegó fluidamente por toda la pista, incluyendo tramos rectos y curvos.
Tasa de éxito general (incluyendo sintonización): 83.3 %		
Tasa de éxito post-sintonización: 100 %		

6. Discusión

Los resultados obtenidos confirman la viabilidad del prototipo como una solución integral para la gestión automatizada de residuos. La discusión se centra en la interpretación de estos resultados, su comparación con el estado del arte y las implicaciones de las decisiones de diseño.

Análisis del Rendimiento. La descripción detallada de las pruebas y sus resultados cuantitativos fundamenta la robustez del sistema. La calibración metódica de los ángulos del servo y las constantes del PID no fue un paso trivial, sino un requisito indispensable para pasar de un modelo teórico a un sistema funcional en el mundo real. El éxito del 100 % en la navegación post-sintonización demuestra que un enfoque de control clásico es altamente efectivo y computacionalmente eficiente para la tarea de seguimiento de línea

en un microcontrolador como el ESP32, adaptándose a las no linealidades del sistema real (fricción, variaciones de voltaje de la batería, etc.).

Comparación con el Estado del Arte. El presente proyecto se diferencia significativamente de otras propuestas de "basureros inteligentes". Trabajos como los de Pavithra et al. (2023) y Perumal et al. (2024) se centran en contenedores estáticos que, si bien notifican su estado, aún dependen de la recolección manual tradicional. Nuestro sistema introduce una innovación logística clave: **la movilidad autónoma**. Al desplazar el contenedor hacia el personal, se invierte el paradigma de recolección, optimizando la mano de obra y eliminando los tiempos muertos de revisión de contenedores vacíos. Esta capacidad de cerrar el ciclo"(monitorear, decidir y actuar físicamente) posiciona a nuestro prototipo como una solución logística más avanzada y no solo como un dispositivo de monitoreo.

Decisiones de Diseño y Limitaciones. La arquitectura de software basada en una máquina de estados finitos (FSM) fue crucial para gestionar la complejidad del sistema de forma ordenada y predecible. Cada estado (`ESPERANDO_EN_BASE`, `VIAJE_A_DESTINO`, etc.) encapsula una lógica específica, lo que facilitó la depuración y previene comportamientos inesperados.

Una limitación observada fue la sensibilidad de los sensores infrarrojos a cambios drásticos de iluminación ambiental y la del sensor ToF a superficies de residuos muy irregulares. Aunque no representaron un problema en el entorno controlado de prueba, en un despliegue real podrían requerir la implementación de filtros de software (ej. media móvil) para promediar las lecturas y aumentar la robustez del sistema. Asimismo, el seguimiento de línea, aunque fiable, limita al robot a una ruta fija.

En resumen, la discusión de los resultados evidencia que el prototipo no solo cumple con sus objetivos, sino que representa un avance conceptual sobre los sistemas existentes, sentando las bases para una gestión de residuos más dinámica y eficiente.

7. Conclusiones

Este proyecto culminó con el diseño, construcción y validación exitosa de un prototipo funcional de robot basurero autónomo, cumpliendo con el objetivo general propuesto. Se demostró la integración efectiva de tecnologías de sensado, actuación, control y comunicación IoT para crear una solución innovadora a la gestión de residuos en interiores.

Las principales conclusiones son:

1. **Se logró un sistema de recolección autónomo y funcional.** El robot integra exitosamente el monitoreo de estado (nivel de llenado, presencia de usuarios) con la acción física (apertura de tapa, navegación autónoma). La implementación de una máquina de estados finitos fue clave para orquestar el ciclo completo de operación de manera robusta y fiable.

2. **El control PID es una estrategia eficaz y eficiente para la navegación guiada.**
La correcta implementación y sintonización del controlador PID, combinada con el modelo cinemático diferencial y la calibración motor-PWM, permitió un seguimiento de línea preciso y estable, validando esta técnica como una solución idónea para sistemas embebidos con recursos computacionales limitados.
3. **La arquitectura IoT basada en ESP32 y Firebase es robusta y escalable.** La plataforma seleccionada demostró ser una solución costo-eficiente para la transmisión, almacenamiento y visualización de datos en tiempo real. El dashboard desarrollado en React proporcionó una interfaz de usuario intuitiva y efectiva para el monitoreo remoto del sistema.
4. **El proyecto representa una innovación en la logística de gestión de residuos.**
A diferencia de los basureros inteligentes estáticos, este prototipo introduce un nuevo paradigma donde el contenedor se desplaza autónomamente al punto de acopio. Esta capacidad de "limpieza por demanda" tiene el potencial de optimizar significativamente los recursos humanos y mejorar la eficiencia operativa en entornos como oficinas, hospitales o centros educativos.

En definitiva, este trabajo no solo valida la viabilidad técnica de la solución propuesta, sino que también sirve como una sólida prueba de concepto para futuras investigaciones y desarrollos en el campo de la robótica de servicios y las ciudades inteligentes.

8. Recomendaciones y Trabajo Futuro

Gestión de Energía: Integrar un sistema de monitoreo de batería y programar un comportamiento de retorno autónomo a una estación de carga.

Navegación Avanzada: Sustituir el seguidor de línea por técnicas de SLAM (Localización y Mapeo Simultáneos) usando un sensor LiDAR o una cámara de profundidad para una navegación más flexible en entornos dinámicos.

Inteligencia Artificial en el Borde (Edge AI): Incorporar una cámara y un microcontrolador más potente (como un ESP32-S3) para ejecutar modelos de clasificación de imágenes y separar automáticamente los residuos.

(Otras fuentes: datasheets de componentes, libros de robótica, documentación de Firebase, etc.)

9. Anexos

Anexo A: Datos del enviados por el robot basurero en formato json y en el firebase.

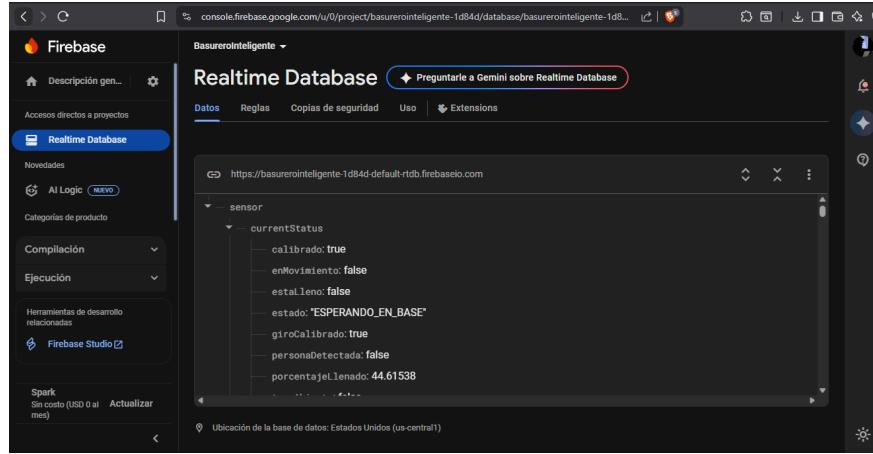


Figura 26: Datos robot visto en el firebase

```

1  {
2   "sensor": {
3     "currentStatus": {
4       "calibrado": true,
5       "enMovimiento": false,
6       "estalleno": false,
7       "estado": "ESPERANDO_EN_BASE",
8       "giroCalibrado": true,
9       "personaDetectada": false,
10      "porcentajeLlenado": 44.61538,
11      "tapaAbierta": false,
12      "timestamp": 1753879798
13    },
14    "historialllenado": [
15      "-OwBh4px32.lokz0iFv": {
16        "porcentajeLlenado": 0,
17        "timestamp": 1753633545
18      },
19      "-OwBh4xamRfNsNxDnybz": {
20        "porcentajeLlenado": 0,
21        "timestamp": 1753633545
22      },
23      "-OwBh78SiSNpmCWSVyi": {
24        "porcentajeLlenado": 0,
25        "timestamp": 1753633554
26      },
27      "-OwBh9mxIxuq2gMmYci": {
28        "porcentajeLlenado": 3.84615,
29        "timestamp": 1753633565
30      },
31      "-OwBhAhsogK9Gnx2hLWg": {
32        "porcentajeLlenado": -7.60231
33      }
34    ]
35  }
36}

```

Figura 27: Datos robot en formato json

Referencias

- [1] Venkadesh, P., Divya, S. V., & Vishal, N. (2024). Smart Dustbin using ESP32 for Waste Management. *IRO Journal on Sustainable Wireless Systems*, 6(4), 309-317.
- [2] Shobha, K.S., Chethana B., Harshitha B., Divyashree N., & Kavya M. (2021). Garbage Collection Robot Using Wireless Communication. *International Journal for Research in Applied Science and Engineering Technology*, 9(V), 1845-1849. doi:10.22214/ijraset.2021.34691
- [3] Raju, D. L. N., Gopal, N. B., Kumar, P. P., Murthy, G. L. N., & Sadulla, Sk. (2022). Smart Waste Management with Intelligent Autonomous Robot for Monitoring Dustbins and Obstacle Avoidance (SmartBin Rover). *International Journal for Research in Applied Science & Engineering Technology*, 10(VI). doi:10.22214/ijraset.2022.43586
- [4] Pavithra, G., Venkatesan, V., Karthikeyan, P., Bharathi, M., & Govindaraj, R. (2023). IoT Based Automated Smart Waste Management System. *International Journal of Research Publication and Reviews*, 4(3), 1331–1336. doi: 10.55248/gengpi.2023.4.31354
- [5] Perumal, M., Kiruthika, M., & Subalakshmi, R. (2024). Smart Dustbin using ESP32 for Waste Management. *International Journal of Creative Research Thoughts (IJCRT)*, 12(1), 2171–2174. doi: 10.5281/zenodo.10689080
- [6] Proyecto ecoTacho. (s.f.). *Informe técnico del prototipo de gestión de residuos para instituciones educativas*. Documento no publicado, Lima, Perú.
- [7] Gestión de residuos por medio de sensores inalámbricos. (s.f.). *Informe académico interno*. Universidad Tecnológica, Lima, Perú. Documento inédito.
- [8] Perumal, M. et al. (2024). *Smart Dustbin using ESP32 for Waste Management*. International Journal of Engineering Trends and Technology. DOI: 10.14445/22315381/IJETT-V72I5P219
- [9] Pavithra, P. et al. (2023). *IoT based Garbage Monitoring System using GSM*. International Journal of Computer Trends and Technology. DOI: 10.14445/22312803/IJCTT-V74I7P105
- [10] Manara, D. et al. (2024). *Multiagent simulation to optimize garbage collection routes using IoT*. Procedia Computer Science. DOI: 10.1016/j.procs.2023.12.015
- [11] SciTePress. (2024). *Autonomous Ecobot with Visual Separation using Computational Vision and IoT*. Proceedings of the 21st International Conference on Informatics.
- [12] IJERT. (2022). *Design of Basic Garbage Monitoring System using IR, Arduino and RF*. International Journal of Engineering Research Technology.
- [13] Brahmanandam, R. et al. (2024). *Design of IoT-Based Smart Trash Bin Monitoring Using ESP32 and Firebase*. International Journal of Advanced Research in Engineering and Technology. DOI: 10.1007/s40003-023-01024-0

- [14] Venkadesh, S., Divya, R., & Vishal, S. (2024). *IoT-Based Smart Waste Management System (SWIMS)*. International Journal of Engineering Trends and Technology. DOI: 10.14445/22312803/IJETT-V7I5P212
- [15] Aiswarya, R. et al. (2023). *IoT BASED SMART WASTE BIN MODEL*. International Research Journal of Engineering and Technology (IRJET). DOI: 10.5281/zenodo.10293152
- [16] Perumal, M., Kiruthika, M., & Subalakshmi, R. (2024). Smart Dustbin using ESP32 for Waste Management. *International Journal of Creative Research Thoughts (IJCRT)*, 12(1), 2171–2174. doi: 10.5281/zenodo.10689080
- [17] Kaur, J., Singh, J., & Kaur, K. (2020). A Review of Smart Waste Management Systems using IoT and Embedded Technologies. *International Journal of Engineering Research & Technology (IJERT)*, 9(07). doi: 10.17577/IJERTV9IS070104
- [18] Singh, A., Singh, N., & Yadav, K. (2022). IoT Based Smart Waste Bin Model to Optimize the Waste Management Process. *Journal of Emerging Technologies and Innovative Research (JETIR)*, 9(5), c401-c405. Recuperado de https://www.researchgate.net/publication/360834384_IOT_BASED_SMART_WASTE_BIN_MODEL_TO_OPTIMIZE_THE_WASTE_MANAGEMENT_PROCESS
- [19] Ngure, S. N., Mutuku, J. K., & Wambua, P. M. (2025). An Internet of Things (IoT) Based Smart Waste Bin for Smart Cities. *International Journal of Engineering and Management Technology*, 11(1), 220-231. Recuperado de <https://iiardjournals.org/get/IJEMT/VOL.%2011%20NO.%201%202025/AN%20INTERNET%20OF%20THINGS%202020-231.pdf>
- [20] Farooq, U., Malik, S. H., Zeeshan, M., & Ahmad, S. (2024). IoT-Based Smart Waste Management System (SWIMS). *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 37(1), 374-386. doi: 10.37934/araset.37.1.374386
- [21] Saini, A., Goyal, N. (2022). IoT BASED GARBAGE MONITORING SYSTEM. *International Research Journal of Multidisciplinary Engineering, Technology & Science (IRJMETS)*, 8(4), 16-20. Recuperado de https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2022/20705/final/fin_irjmets1649864569.pdf
- [22] Adesta, E., Rahman, N. K., Hidayah, N. N., & Faliq, F. (2024). Design of IoT-Based Smart Trash Bin Monitoring Using ESP32 and Firebase. *Jurnal Rekayasa Elektrika*, 20(2). doi: 10.21107/jre.v20i2.21312
- [23] Pattnaik, S., Jena, P. K., & Mohapatra, P. C. (2022). A Smart Waste Management System Framework Using IoT and Machine Learning. *International Journal of Recent Innovations in Technology and Computer Science (IJRITCC)*, 5(2), 346-352. doi: 10.51475/ijritcc.2022.5209
- [24] Salman, H. M., Hamad, Y. M., & Hameed, M. A. (2023). IoT-Enabled Solid Waste Management in Smart Cities. *Journal of Engineering and Applied Sciences*, 20(1). doi: 10.46827/jeas.v20i1.350

- [25] Shewale, R. T., Kulkarni, V. B., Jadhav, A. K., & Jadhav, P. A. (2019). WASTE MANAGEMENT SYSTEM WITH THINGSPEAK. *International Research Journal of Engineering and Technology (IRJET)*, 6(6), 3959-3962. Recuperado de <https://www.irjet.net/archives/V6/i6/IRJET-V6I6775.pdf>
- [26] Ahmed, S. A., Ramachandran, S. A., & Basha, A. S. (2024). Smart Waste Collecting Robot Integration With IoT and Machine Learning. *Journal of Robotics and Automation Systems*, 1(1). Recuperado de <https://www.scribd.com/document/716908666/Smart-Waste-Collecting-Robot-Integration-With-IoT-and-Machine-Learning-1>
- [27] Singh, A., Singh, B. K., & Singh, S. K. (2024). Garbage Collection Robot Using Wireless Communication. *International Journal of Research Publication and Review (IJRPR)*, 6(4), 4347-4351. Recuperado de <https://ijrpr.com/uploads/V6ISSUE4/IJRPR43470.pdf>
- [28] Madhura, B. M., Darshan, K. V., Harshitha, G., & Likitha, S. (2024). IOT BASED DIGITAL WASTE MANAGEMENT ROBOT. *The International Journal of Engineering Research (TIJER)*, 9(4). Recuperado de <https://tijer.org/tijer/papers/TIJERC001325.pdf>
- [29] Lopes, G. G., & Caldas, J. M. (2024). Ecobot: Autonomous Trash Collection and Segregation System. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART)*, 2, 983-990. SciTePress. doi: 10.5220/0012297100003639
- [30] Rajesh, D., Saravanan, S. M., Ramakrishnan, S., & Mohanavel, V. (2022). Development of Smart Waste Bin for Solid Waste Management. *Journal of Smart Cities and Society*, 1(1), 1-8. Recuperado de https://www.researchgate.net/publication/357433081_Development_of_Smart_Waste_Bin_for_Solid_Waste_Management
- [31] Pathan, K. J., Patil, D. P., & Pathan, M. J. (2018). Smart Dustbin: An Efficient Garbage Management Approach for a Healthy Society. *International Journal of Engineering Science and Computing*, 8(11), 16409-16412. Recuperado de https://www.researchgate.net/publication/328995044_Smart_Dustbin_An_Efficient_Garbage_Management_Approach_for_a_Healthy_Society
- [32] Al-Hafi, N., Elhosseny, M., & Ghaleb, H. (2022). IoT-Based Waste Management System in Formal and Informal Public Areas in Mecca. *Sensors*, 22(20), 7935. doi: 10.3390/s22207935
- [33] Gómez-Meza, A., et al. (2021). Análisis medioambiental del manejo de residuos sólidos de los mercados abiertos en Perú, una revisión narrativa. *Ciencia y Tecnología para el Desarrollo Sostenible*, 1(2). doi: 10.55169/cites.v1n2.2021.5
- [34] Díaz, C. P., & Rivas, M. E. (2022). Implementación de automatización de apertura en un basurero, en un área específica. *Revista Científica Unanchay*, 1(1), 60-69. doi: 10.37955/ru.v1i1.14
- [35] Hernández, C. A. C., & Torres, N. G. Z. (2022). Diseño e implementación de un robot móvil para la recolección de desperdicios en la playa. Univer-

sidad Ricardo Palma. Recuperado de <https://www.urp.edu.pe/pdf/id/41571/n/robot-movil-para-la-recolección-de-desperdicios-en-la-playa>

- [36] Al-Rashid, M., et al. (2023). Smart waste collection management system based on IoT and optimization algorithms. *Environmental Science and Pollution Research*. doi: 10.1007/s11356-023-28919-y
- [37] Random Nerd Tutorials. (2025). *Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino*. Recuperado de <https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>
- [38] STMicroelectronics. (2025). *VL53L0X - Time-of-Flight (ToF) ranging sensor*. Recuperado de <https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html>
- [39] Madhusanka, T. (2025). *Real-Time IoT Dashboard with Next.js, ESP32, Firebase, and AWS*. Medium. Recuperado de <https://medium.com/@tharindumadhusanka99/real-time-iot-dashboard-with-next-js-esp32-firebase-and-aws-1b4e0db452b6>
- [40] Rajeshwari, K. V. (2025). Progress Tracking Dashboard using React.JS Review. *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*. Recuperado de <https://ijarsct.co.in/Paper16990.pdf>
- [41] Ahmed, S. E. (2014). PID CONTROL OF LINE FOLLOWERS. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(9), 834-839. Recuperado de <https://core.ac.uk/download/pdf/53189911.pdf>
- [42] Kim, D., & Park, Y. (2023). Design and Implementation of PID Control-based FSM Algorithm on Line Following Robot. *Journal of Korea Institute of Information, Electronics, and Communication Technology*, 16(6), 727-735. doi: 10.17661/jkiect.2023.16.6.727
- [43] Balogh, R., et al. (2018). Using Finite State Machines in Introductory Robotics: Methods and Applications for Teaching and Learning. In *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE. doi: 10.1109/FIE.2018.8659183
- [44] Rajasekar, R., & Ramachandran, K. (2022). Design and Implementation of an Autonomous Line Following Robot. *International Journal of Engineering Research & Technology (IJERT)*, 11(04). Recuperado de <http://www.diva-portal.org/smash/get/diva2:1908071/FULLTEXT01.pdf>
- [45] Alam, M. S., Roy, S., & Hasan, M. A. (2023). Design, Construction and Performance Comparison of Fuzzy Logic Controller and PID Controller for two-Wheel Balance Robot. *Liverpool Journal of Engineering Research*, 6(1), 10-18. Recuperado de <https://journalspress.uk/index.php/LJER/article/view/1222>
- [46] Park, Y. W., & Hong, D. K. (2014). Kinematic model of a differential drive mobile robot. *Journal of Sensor Science and Technology*, 23(6), 464-468. doi: 10.5369/JSSST.2014.23.6.464