Implementación de un prototipo de sistema electrónico de detección de contactos y registro de tiempos como soporte en la determinación de faltas en el circuito de Agility Canino

Heidy Nicoll Mahecha Pérez

Universidad Santo Tomás

Facultad de Ingeniería Electrónica

Bogotá, Colombia

2025

Implementación de un prototipo de sistema electrónico de detección de contactos y registro de tiempos como soporte en la determinación de faltas en el circuito de Agility Canino

Heidy Nicoll Mahecha Pérez

Tesis presentada para obtener el grado de
Ingeniera Electrónica

Director:

Oscar Mauricio Gelvez Lizarazo M.C

Co-Director:

Gustavo Adolfo Higuera Castro M.C

Grupo de Investigación MEM(Modelado-Electrónica-Monitoreo)
Universidad Santo Tomás
Facultad de Ingeniería Electrónica
Bogotá, Colombia
2025

Autoridades de la universidad

RECTOR GENERAL
R.P. FRAY ÁLVARO JOSÉ ARANGO RESTREPO, O.P.

VICERRECTOR ADMINISTRATIVO Y FINANCIERO GENERAL
R.P. FRAY HENDER ALVEIRO RODRÍGUEZ PÉREZ, O.P.

VICERRECTOR ACADÉMICO GENERAL
R.P. FRAY MAURICIO ANTONIO CORTÉS GALLEGO, O.P.

SECRETARIA GENERAL
Dra. LINA MARCELA CAMARGO MARTÍNEZ

DECANO DIVISIÓN DE INGENIERÍAS
R.P. FRAY JAVIER ANTONIO HINCAPIÉ ARDILA, O.P.

SECRETARIA DE DIVISIÓN
E.C. LUZ PATRICIA ROCHA CAICEDO

DECANO FACULTAD DE INGENIERÍA ELECTRÓNICA
PhD. LEONARDO FABIO YEPES ARBELÁEZ

Nota de Aceptación

Firma del Tutor

Firma del jurado

Firma del jurado

BOGOTÁ D.C._____DE 2025

Advertencia

La Universidad Santo Tomás no se hace responsable de las opiniones y conceptos expresados en el trabajo de grado, solo velará por qué no se publique nada contrario al dogma ni a la moral católica y porque el trabajo no tenga ataques personales y únicamente se vea el anhelo de buscar la verdad científica.

Dedicatoria

A mis padres, por ser una gran inspiración, por su amor incondicional y por enseñarme que el esfuerzo y la perseverancia siempre traen grandes recompensas.

A mis docentes, quienes con dedicación y pasión me guiaron en mi formación, inculcando en mí el amor por la ingeniería y la tecnología.

Y a mi familia y amigos, quienes con su compañía y aliento hicieron de este proceso una experiencia llena de aprendizajes y gratitud.

Agradecimientos

En primer lugar, agradezco a mis padres, especialmente a mi madre, quien ha sido mi mayor inspiración y apoyo en cada paso de mi vida. Tu amor, tus sacrificios y tus palabras de aliento me han dado las fuerzas para superar cada obstáculo. Gracias por creer en mí, incluso cuando yo misma dudaba. Este logro también es tuyo, porque sin ti, este camino no habría sido posible.

Además, a mi director de proyecto, Oscar Mauricio Gelvez Lizarazo, y a mi co-director, Gustavo Adolfo Higuera Castro, por su invaluable orientación, dedicación y paciencia durante el desarrollo de este trabajo. Sus enseñanzas y consejos fueron clave para transformar este proyecto en una realidad.

También a mi pareja, por estar siempre a mi lado, por tu paciencia, amor y por ser mi mayor fuente de motivación. Tu confianza en mí ha sido esencial para alcanzar esta meta, y agradezco cada momento en el que me recordaste que soy capaz de todo.

A mis amigos, especialmente a Alejandra Pinzón, Juan Andrés Rivera, y Jorge Palacios, por su compañía, apoyo y confianza en mí. Gracias por escucharme, animarme en los momentos difíciles y celebrar cada pequeño avance como si fuera propio. Su fe en mí y su amistad han sido un regalo invaluable a lo largo de este proceso.

Finalmente a la Universidad Santo Tomás, al grupo de investigación MEM (Modelado-Electrónica-Monitoreo), por proporcionar los recursos, el conocimiento y el espacio para desarrollar este proyecto. Y a todas las personas que, de una u otra forma, contribuyeron a que este trabajo fuera posible. Su apoyo, directo o indirecto, me ayudó a seguir adelante y a alcanzar este importante logro.

Índice general

Índice de cuadros

Índice de figuras

Resumen

El presente trabajo de grado tiene como finalidad el diseño, desarrollo e implementación de un prototipo de sistema electrónico para la detección automatizada de contacto y el registro de tiempo en circuitos de Agility Canino, como herramienta de soporte para la labor del arbitraje en competencias oficiales y entrenamientos. Este sistema reduce el margen de error asociado al juicio visual humano mediante la incorporación de tecnologías de sensado y procesamiento embebido.

La solución propuesta contempla la implementación de una red de sensores piezorresistivos construidos a partir de material Velostat, el cual permite detectar variaciones en la presión superficial ejercida por los perros al atravesar zonas específicas de contacto en los obstáculos, según lo establecido en los reglamentos internacionales. De manera paralela, se integra un sistema de cronometraje basado en sensores ultrasónicos HCSR04, encargados de iniciar y finalizar la medición de tiempo en el recorrido, con una exactitud menor 5ms y sin intervención manual.

Ambos sistemas están conectados a una unidad de procesamiento basada en un microcontrolador ESP32, que permite la adquisición de señales, su acondicionamiento y transmisión de datos en tiempo real. La información recolectada es enviada a una plataforma web desarrollada específicamente para este propósito, la cual permite visualizar los tiempos de recorrido, y el contacto en los obtáculos o la falta del mismo realizado por cada participante. Esta interfaz está diseñada para ser utilizada por jueces, entrenadores y público en general.

El prototipo fue validado en un entorno controlado que simula una competencia en una pista de agility, evaluando parámetros como la exactitud del cronometraje, la sensibilidad de detección de los sensores Velostat, la latencia de procesamiento y la estabilidad de la comunicación entre los módulos. Los resultados obtenidos demuestran un desempeño favorable frente a los métodos tradicionales, con un sistema de bajo costo, y con potencial de escalabilidad.

Esta propuesta aporta al desarrollo de herramientas tecnológicas en el ámbito deportivo, promueve la transparencia en la toma de decisiones durante las competencias y representa un avance hacia la transformación digital del arbitraje en el agility canino.

Palabras clave: Velostat, sensores ultrasónicos, cronometraje digital, ESP32, microcontroladores, detección de contacto, sensado electrónico, agility canino, interfaz web, arbitraje automático.

Abstract

The present undergraduate thesis aims to design, develop, and implement a prototype of an electronic system for automated contact detection and time recording in Canine Agility circuits, serving as a support tool for refereeing during official competitions and training sessions. This system reduces the margin of error associated with human visual judgment through the integration of sensing technologies and embedded processing.

The proposed solution includes the implementation of a network of piezoresistive sensors built using Velostat material, which allows the detection of surface pressure variations exerted by dogs when crossing specific contact zones on the obstacles, as established by international regulations. In parallel, a timing system based on HCSR04 ultrasonic sensors is integrated, responsible for starting and stopping the time measurement of the run, achieving an accuracy better than 5 ms without manual intervention.

Both subsystems are connected to a processing unit based on an ESP32 microcontroller, which enables signal acquisition, conditioning, and real-time data transmission. The collected information is sent to a web platform specifically developed for this purpose, allowing visualization of the run times and contact detection (or lack thereof) for each participant. This interface is designed for use by judges, trainers, and the general audience.

The prototype was validated in a controlled environment simulating a competition on an agility track, evaluating parameters such as timing accuracy, sensor detection sensitivity, processing latency, and communication stability between modules. The obtained results demonstrate favorable performance compared to traditional methods, with a low-cost system and potential for scalability.

This proposal contributes to the development of technological tools in the sports field, promotes transparency in decision-making during competitions, and represents progress toward the digital transformation of refereeing in canine agility.

Keywords: Velostat, ultrasonic sensors, digital timing, ESP32, microcontrollers, contact detection, electronic sensing, canine agility, web interface, automatic refereeing.

Glosario

- Agility Canino: Deporte competitivo en el que un perro debe recorrer una pista con obstáculos en el menor tiempo posible, siguiendo las indicaciones de su guía, sin cometer errores ni penalizaciones.
- Arbitraje Automatizado: Proceso mediante el cual se utiliza tecnología para asistir o reemplazar el juicio humano en la evaluación de eventos deportivos, mejorando la objetividad y precisión de las decisiones.
- Calibración de Sensores: Proceso mediante el cual se comparan las lecturas del sensor con valores de referencia conocidos para determinar su error o desviación.
- Cronometraje Digital: Técnica de medición de tiempo basada en sistemas electrónicos automáticos, eliminando la intervención manual para mejorar la exactitud.
- ESP32: Microcontrolador de alto rendimiento con conectividad Wi-Fi y Bluetooth, utilizado para el procesamiento de señales y la comunicación inalámbrica entre módulos del sistema.
- Interfaz Web: Plataforma digital accesible desde navegadores de internet que permite visualizar, almacenar y gestionar la información recolectada por el sistema de detección y cronometraje.
- Latencia: Tiempo de retraso entre la ocurrencia de un evento físico (como un contacto o cruce) y su detección y procesamiento por parte del sistema.
- Microcontrolador: Dispositivo electrónico programable que integra una unidad de procesamiento, memoria y periféricos, y que permite controlar el funcionamiento de sistemas electrónicos embebidos.
- Sistema Embebido: Sistema computacional diseñado para realizar tareas específicas dentro de un dispositivo, comúnmente con restricciones de tamaño, consumo energético y procesamiento.
- Sensor Piezoresistivo: Dispositivo que detecta variaciones en la resistencia eléctrica provocadas por cambios en la presión o fuerza mecánica sobre su superficie.
- Sensor Ultrasónico: Dispositivo electrónico que mide distancias mediante la emisión de ondas sonoras de alta frecuencia y el análisis del tiempo que tarda el eco en regresar tras reflejarse en un objeto. Se utiliza comúnmente para detección de proximidad y medición sin contacto.
- Velostat: Material conductor de polímero sensible a la presión, cuya resistividad varía al aplicar fuerza mecánica. Es ampliamente utilizado para la construcción de sensores piezorresistivos de bajo costo.

- Zona de Contacto: Área específica dentro de los obstáculos del circuito de agility que el perro debe pisar obligatoriamente; omitirla genera penalización.

Capítulo 1

Introducción

El Agility Canino es una disciplina deportiva que ha ganado gran popularidad a nivel internacional gracias a la destreza, velocidad y compenetración que exige entre el perro y su guía. En este deporte, los perros deben recorrer una pista con diversos obstáculos en el menor tiempo posible, evitando cometer errores que puedan acarrear penalizaciones. A pesar del crecimiento y profesionalización del agility, especialmente en países como Colombia, el arbitraje de este tipo de competencias continúa dependiendo en gran medida del juicio visual de los jueces, lo cual introduce subjetividad y un margen significativo de error en la evaluación.

Actualmente, las pistas de Agility en Colombia no cuentan con sistemas automatizados para la detección de faltas. Esta situación puede afectar la transparencia, equidad y confiabilidad de los resultados, tanto en competencias oficiales como en sesiones de entrenamiento. En un entorno donde la precisión y objetividad son fundamentales, surge la necesidad de desarrollar soluciones tecnológicas que permitan optimizar el proceso de arbitraje mediante métodos automáticos de sensado y registro.

Desde esta perspectiva, el presente proyecto tiene como objetivo principal diseñar e implementar un prototipo funcional de sistema electrónico capaz de detectar automáticamente el contacto de los perros con las zonas reglamentarias de los obstáculos y registrar los tiempos de recorrido mediante sensores especializados. Para ello, se propuso una arquitectura compuesta por sensores piezorresistivos fabricados con Velostat para la detección de contacto, sensores ultrasónicos para el cronometraje y un sistema embebido basado en un microcontrolador ESP32, encargado de procesar y transmitir la información a una plataforma web. Esta interfaz permite la visualización de los datos por parte de jueces, entrenadores y espectadores, promoviendo la transparencia en la evaluación.

Este sistema busca no solo mejorar el control en las competencias de agility, sino también convertirse en una herramienta útil para el entrenamiento, permitiendo identificar con precisión los errores cometidos durante los recorridos. Además, su carácter de bajo costo, flexibilidad y fácil implementación lo convierten en una alternativa viable para ser replicada en diferentes escuelas y escenarios de competencia a nivel nacional e internacional.

La validación del prototipo se realizó en un entorno de pruebas controlado, simulando las condiciones reales de una competencia de agility, lo que permitió evaluar su precisión, sensibilidad, latencia y estabilidad. Los resultados demuestran que el sistema propuesto representa una mejora significativa frente al arbitraje manual tradicional, permitiendo una mayor rigurosidad técnica y reduciendo el margen de error.

En resumen, esta investigación se enmarca en la búsqueda de soluciones tecnológicas innovadoras para mejorar el deporte, fortalecer la objetividad del arbitraje y apoyar el desarrollo de herramientas accesibles para eventos y entrenamientos caninos. El trabajo contribuye al cumplimiento de objetivos de desarrollo sostenible relacionados con el bienestar, el deporte, la innovación y el fortalecimiento de comunidades en torno a la interacción humano-animal.

1.1. Planteamiento del Problema

El Agility es un deporte canino que combina la destreza física, el trabajo en equipo y la coordinación entre el perro y su guía, en el cual los perros deben superar una serie de obstáculos en un circuito cronometrado, incluyendo saltos, túneles, pasarelas, entre otros, en el menor tiempo posible y sin cometer errores [1].

En Colombia, este deporte ha experimentado un crecimiento significativo en los últimos años ya que se considera una buena alternativa para afianzar las relaciones existentes entre las mascotas y sus dueños [2]. Sin embargo, los circuitos de agility no cuentan con un sistema automatizado de verificación de las faltas cometidas por los competidores, y al momento de utilizar la pista, ya sea por entrenamientos o competencias, los responsables de definir las penalizaciones correspondientes son los jueces de los eventos quienes toman la decisión de manera visual, lo cual implica un arbitraje basado en la percepción humana introduciendo un gran número de posibilidades en las tomas de decisiones según el juez adjudicado en el evento [3]; lo anterior conduce a tener una relación inversa en la que se puede deducir mediante un análisis descriptivo que mientras menos perceptible sea la falta cometida por el animal o su entrenador, hay una mayor posibilidad de cometer un error en la elección de la penalización (o ausencia de ella), ya que la falta de rigurosidad en el cumplimiento de las normativas y reglas de la competencia puede conducir a una premiación o descalificación injusta de los competidores.

Diagnóstico de la situación actual

En Colombia, las competencias de Agility continúan dependiendo de un sistema manual de evaluación, sin la incorporación de sensores ni herramientas tecnológicas que permitan medir objetivamente el contacto de los perros con las zonas reglamentarias o registrar con exactitud el tiempo de recorrido. Esta limitación se da a pesar de que el deporte ha ganado espacio dentro del calendario nacional de competencias y ha despertado el interés de entrenadores, jueces y público general. En muchos casos, las decisiones arbitrales se basan

únicamente en la ubicación visual del juez, lo que representa un margen significativo de error, especialmente en recorridos de alta velocidad y con múltiples obstáculos. Además, el cronometraje muchas veces se realiza de forma manual, introduciendo imprecisiones que pueden alterar los resultados finales de los eventos.

Pronóstico de la situación descrita

De mantenerse esta situación, la credibilidad del arbitraje podría verse comprometida, afectando la transparencia de los resultados y la motivación de los participantes. Esto no solo limita la posibilidad de crecimiento profesional del deporte en el país, sino que también restringe la incorporación de Colombia en circuitos internacionales con altos estándares tecnológicos. El uso de nuevas tecnologías permitiría detectar con mayor precisión el cumplimiento de las normas, reducir errores humanos, ofrecer retroalimentación en tiempo real y elevar la calidad de las competencias. Sin una intervención tecnológica, el desarrollo del agility en Colombia continuará limitado por métodos tradicionales que no responden a las demandas actuales de eficiencia, transparencia y modernización.

1.2. Pregunta Problema

¿Cómo las técnicas de sensado electrónico de última generación se pueden emplear para contribuir a robustecer el cumplimiento de normas y reglamentaciones en competencias de agility canino?

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un prototipo de sistema electrónico funcional de determinación de la existencia de contactos y registros de tiempo como soporte en el arbitraje en circuitos de Agility Canino mediante el despliegue de una red de sensores.

1.3.2. Objetivos Específicos

- Diseñar el circuito electrónico que integre los sensores para la determinación de la existencia de contactos y registro de tiempo, así como los elementos de visualización y monitoreo de variables para el arbitraje en un circuito de agility canino.
- Implementar el circuito electrónico a partir del procesamiento de las señales adquiridas utilizando un microcontrolador que facilite la determinación del tiempo de recorrido y la existencia de contactos de los perros durante el circuito.
- Validar el funcionamiento del prototipo de sistema a través de ensayos bajo condiciones controladas, ajustando los parámetros necesarios para la determinación de

precisión y latencia.

1.4. Justificación

Colombia cuenta aproximadamente con 24 escuelas de Agility Canino inscritas ante la ACCC (Asociación Club Canino Colombiano) principalmente ubicadas en Bogotá, Medellín y Cali, las cuales han presentado un incremento en su creación y registro en los últimos años [4]. Además teniendo en cuenta el calendario de competencias estipulado por Agility Colombia, y el calendario de eventos de la Asociación Club Canino Colombiano se tuvieron organizadas 17 competencias y alrededor de 31 eventos diferentes a nivel nacional para el 2024 [5] [6]. Internacionalmente existe un aproximado de 43 campeonatos programados entre los años 2024 y 2027 en países como Bélgica, Italia, Austria, España, entre otros, según el calendario de campeonatos de la *Federation Cynologique Internationale*(FCI) [7].

Teniendo en cuenta el crecimiento positivo que viene presentando el Agility Canino en Colombia, el presente proyecto busca contribuir a la robustez en el proceso de arbitraje de cada participante, además de facilitar la verificación de las faltas de una manera más confiable por medio de un sistema de sensores distribuidos en los obstáculos, reduciendo el porcentaje de error en las penalizaciones a los competidores. Lo anterior permite incrementar los estándares de calidad en los torneos debido a la mejora de la transparencia en el arbitraje, estimulando posiblemente una mayor participación en este deporte, además de brindar mayores oportunidades de reconocimiento a esta forma de actividad física para todos aquellos que disfrutan de la compañía de sus mascotas caninas.

El presente proyecto surge como respuesta a una necesidad concreta identificada en el entorno del agility canino en Colombia: la falta de herramientas tecnológicas que permitan garantizar la transparencia, precisión y eficiencia en la labor de arbitraje durante las competencias y entrenamientos. A pesar del creciente interés en esta disciplina deportiva y su avance a nivel internacional, en el contexto nacional se siguen empleando métodos manuales que dependen casi exclusivamente del juicio visual de los jueces, lo cual introduce subjetividad, inconsistencias y posibilidad de errores en la evaluación del desempeño de los participantes.

Desde el enfoque teórico, este proyecto aporta al estudio y aplicación de tecnologías de sensado en contextos no convencionales, combinando conocimientos en electrónica, sistemas embebidos, interfaz web e Internet de las Cosas (IoT). La utilización de sensores piezorresistivos fabricados con Velostat y sensores ultrasónicos permite profundizar en el análisis del comportamiento físico de materiales sensibles al contacto, así como en la exactitud del cronometraje automático en escenarios dinámicos.

En el plano metodológico, la propuesta incluye el diseño de una arquitectura electrónica modular conectada a una plataforma web, permitiendo validar diferentes combinaciones de sensores, niveles de sensibilidad, rutinas de procesamiento y formas de visualización. Este proceso de desarrollo incluye fases de diseño, implementación, pruebas en entorno controlado y ajuste iterativo, lo que proporciona un modelo replicable para proyectos similares en otros ámbitos deportivos o de control físico de eventos.

Desde una perspectiva práctica, el sistema busca ser una herramienta de bajo costo, portátil, adaptable a distintas pistas y con un nivel de complejidad operativa reducido, lo cual facilita su implementación tanto en competencias oficiales como en entrenamientos. Además, su uso promueve la equidad en la toma de decisiones, ya que disminuye el margen de error y mejora la experiencia tanto de los participantes como del público observador.

Este trabajo beneficia directamente a jueces, entrenadores y organizadores de eventos de Agility, quienes contarán con un soporte tecnológico para evaluar el desempeño de los perros de forma más justa y precisa. De igual forma, impacta positivamente en los dueños y guías, quienes tendrán acceso a información objetiva y detallada que les permitirá mejorar sus estrategias de entrenamiento.

Finalmente, este proyecto contribuye al fortalecimiento de la cultura de innovación en el deporte al integrar herramientas tecnológicas que favorecen la transformación digital de las prácticas tradicionales. Así mismo, representa una experiencia formativa integral para la autora, al aplicar de forma práctica los conocimientos adquiridos durante su proceso académico en ingeniería electrónica, y al aportar una solución concreta con potencial de crecimiento y escalabilidad.

1.5. Impacto Social

El presente proyecto cuenta con un enfoque que beneficia a los organizadores de eventos de Agility a nivel nacional e internacional ofreciendo un sistema electrónico de fácil implementación y bajo costo, que ayude a un arbitraje más eficaz según el cumplimiento de las normas y reglamentaciones en las competencias, para brindar unas mejores condiciones de evaluación contribuyendo a su transparencia.

También se busca brindar un apoyo a los jueces de los eventos por medio del manejo de un cronómetro digital para el registro del tiempo durante el recorrido de una manera automática, y utiliza un sistema con sensores ubicados estratégicamente en los diferentes obstáculos para una correcta verificación del paso que tiene el animal por el circuito, donde se utilizan las señales tomadas por los sensores y se envían a los jueces para permitir un arbitraje más acertado hacia cada uno de los competidores.

Éste contribuye a los entrenadores y participantes a mejorar la identificación y precisión del recorrido de sus perros por medio de la práctica en una pista que tenga el sistema implementado y promueve el sentido de justicia para los participantes y espectadores en cada una de las decisiones tomadas durante las competencias.

Según los objetivos de desarrollo sostenible, el proyecto abarca el objetivo de desarrollo sostenible número 3 de salud y bienestar, el cual plantea la idea de garantizar una vida sana y promover el bienestar para todos en todas las edades, debido a la búsqueda de un estilo de vida que se mantenga con el tiempo [8]; se ha demostrado que tener un perro permite liberar oxitocina, incluso mucho más que cuando se habla e interactúa con un humano, además, tener un perro trae bastantes beneficios respecto al bienestar de una persona, reduce la presión arterial, el cortisol, el estrés, ayuda a mejorar y fortalecer el sistema inmunológico, promueven la actividad física, el deporte, una vida social activa, y funciona como un apoyo terapéutico y emocional, brindan seguridad y fomentan el sentido de la responsabilidad [9].

Adicionalmente, el proyecto contribuye al objetivo de desarrollo sostenible número 9 de Industria, innovación e infraestructura, el cual busca fomentar la innovación, la investigación y el desarrollo de tecnologías, mediante la creación de una infraestructura tecnológica accesible y adaptada a este deporte, busca complementar y robustecer el arbitraje de Agility canino por medio de un sistema novedoso [10].

En consecuencia, la propuesta impacta positivamente en el fortalecimiento de la cultura deportiva, la innovación tecnológica local y la construcción de comunidades que valoran la transparencia, el respeto y el bienestar animal.

Capítulo 2

Estado del Arte

En el artículo "Diseño, desarrollo y validación de un cronómetro con un ordenador de placa reducida"[11] se plantea la construcción de un cronómetro de bajo coste para eventos deportivos, específicamente para el patinaje mediante el empleo de un dispositivo Raspberry Pi, el cual debe ser portátil, autosuficiente, fácil de montar y resistente; al producto obtenido se le realiza una validación en la precisión de la medición por medio de un péndulo físico, cuya longitud de cuerda es conocida, al que se le mide el periodo en distintas condiciones, concretamente la temperatura ambiente, el tiempo de uso, así como el tiempo de espera desde el encendido, los resultados obtenidos en el método de validación realizado determinan que el dispositivo se mantiene estable a lo largo de su uso, no generando valores atípicos o aislados apreciables, por lo tanto, se afirma que el cronómetro soporta largas sesiones de uso sin que su estabilidad se vea comprometida. Igualmente, no se detectan variaciones causadas por el calentamiento de los elementos electrónicos del cronómetro, por lo que no se precisa de la instalación de ningún sistema de renovación de aire interno para la evacuación del calor desprendido por los mismos.

Según el artículo "Desarrollo de un sistema de detección de movimiento usando un sensor infrarrojo pasivo para implementación en robótica", [12] se muestra un proyecto que propone la caracterización, junto al desarrollo de un sistema de detección de movimiento de los sensores PIR (PassiveInfrared). Generalmente el sensor da como respuesta una variable eléctrica con respecto a una variable de entrada, normalmente de naturaleza física, o viceversa; es decir, calcula la tasa de cambio de una variable de entrada cuando tiene cierta señal de salida. Una buena caracterización produce mediciones con gran precisión al momento de determinar el movimiento de personas, lo que posteriormente se utiliza para todo tipo de robótica movil.

Al continuar con la investigación se encontró el artículo "Diagnóstico de sensores de detección de movimiento con ayuda de la tecnología arduino en la AA.HH Villa María" en la cual se dice que los sensores de detección de movimiento son los dispositivos que captan el movimiento en un área determinada mediante las señales que puede emitir y recibir,

para que así pueda reconocer el movimiento físico en un área limitada, esto lo utilizan con el objetivo de mejorar la Seguridad de Viviendas; dice además que los sensores de detección de movimiento poseen varios tipos de sensores como el ultrasonido, el infrarrojo, y el multi-tecnológico, estos detectores [13].

En el artículo "Medida de la fuerza sobre el cuerpo humano mediante sensores piezorresistivos de tipo film", [14] se busca la posibilidad de medir la fuerza o presión ejercida, este tipo de sensores presentan una disminución de resistencia cuando se incrementa la fuerza que actúa sobre su superficie, en el trabajo se ha realizado un estudio de las distintas tecnologías (shuntmode y truemode) y sensores disponibles en el mercado. Se han seleccionado tres fabricantes diferentes (Interlink, Sensitronics y Tekscan) y se han realizado un estudio comparando sus principales características: Linealidad, histéresis, drift o deriva (respuesta del sensor en el tiempo). Dejando como conclusión que las fuerzas cortantes tienen efecto sobre la medida de los sensores, en este trabajo se ha comprobado que la colocación de materiales deformables como son las espumas sobre la superficie de los sensores transmiten una fuerza cortante aunque la fuerza ejercida sea normal a la superficie del sensor. Este efecto ya está incluido en la calibración específica que se ha realizado en este trabajo, pues los sensores han sido calibrados con espumas colocadas sobre su superficie. La curvatura de los sensores también tiene un efecto sobre su medida, pero al igual que con las fuerzas cortantes la curvatura se incluye en la calibración especifica de los sensores.

En el artículo "Implementación de un sistema para la medición de fuerza basado en el efecto piezorresistivo" se plantea realizar un software en LabView que muestre de manera gráfica y numérica el valor de una medida de fuerza realizada con su correspondiente unidad de medida, para esto se implementa el sensor piezorresistivo FlexiForce modelo A201, el cual fue escogido debido a sus múltiples propiedades físicas y de fun-cionamiento, y posteriormente se transforma la variación de resistencia del sensor piezorresistivo en una variación de voltaje [15].

En el proyecto "Centro de capacitación y adiestramiento canino en el distrito de San Juan de Miraflores" ,[16] se explica la importancia de espacios óptimos e integrales dirigidos a los canes y sus dueños, en este se realizan espacios abiertos para las actividades de entrenamiento, exhibición de lo aprendido, aulas para charlas de capacitación a personas, tiendas grooming, espacio para veterinaria básica, etc.; contemplando dentro de la programación zonas de: adiestramiento, capacitación y educación, hospedaje, recreación, veterinaria, servicios complementarios y administrativos. Éste funciona como justificación del presente proyecto, ya que muestra la importancia de tener un lugar adecuado para el entretenimiento y deporte de los perros en el cual puedan compartir con sus dueños.

En la tesis "Desarrollo de un sistema de adquisición de señales de fuerzas plantares de reacción del suelo en los tres planos del espacio" [17] se realiza un sistema de adquisición

que mide las fuerzas plantares en cinco puntos diferentes, el cual permite adquirir wearable de señales de fuerzas plantares en tres ejes ortogonales (el plano vertical, antero- posterior y medio- lateral) para su uso en un ensayo de entrenamiento de voleibol. Además el sistema de adquisición de señales consigue satisfactoriamente muestrear la señal de fuerza detectada en el sensor a 500 Hz y transmitirla inalámbricamente a un teléfono móvil con sistema operativo Android y permite observarlas en tiempo real mediante aplicación móvil.

En la tesis "Sensor de presión flexible basado en nanoalambres de plata" [18] se describe el desarrollo de un sensor de presión flexible basado en nanoalambres de plata, como parte de las investigaciones que se están llevando actualmente para el avance de la electrónica flexible y las aplicaciones de nanomateriales como sustitutos de materiales que se utilizan actualmente en la fabricación de componentes electrónicos rígidos con el objetivo de implementar en un futuro cercano este tipo de electrónica en dispositivos no solo en la industria sino en la vida cotidiana. Este tipo de sensores. El funcionamiento del sensor se basa en el cambio de resistencia en las terminales del electrodo al existir una fuerza o presión aplicada sobre el sensor, donde un incremento en la fuerza aplicada mostrará una disminución en la resistencia medida, debido a un mayor contacto entre los nanoalambres de las películas resistiva y el cobre en el electrodo.

Teniendo en cuenta el proyecto de "Diseño y construcción de un equipo para laboratorio de física de movimiento de caída libre operado mediante software" [19] donde se realiza un sistema electrónico y de control del equipo de laboratorio de caída libre haciendo uso de módulos de sensor infrarrojo fc-51 para la detección del inicio de la caída del balín, y la posición de tambor de lanzamiento, además se utilizó un módulo de sonido ky-036 para la detección de impacto y finales de carrera para la posición de altura y del sistema de realimentación de balines, finalmente la comunicación para el control del equipo es de tipo serial por medio de un puerto USB, este equipo permite realizar las mediciones necesarias en un laboratorio de práctica de caida libre.

En el trabajo de grado "Diseño y prototipado de una plantilla sensorizada para la monitorización de la pisada"[20] se realiza el diseño y prototipado de una plantilla sensorizada para la monitorización de la pisada, lo que permite obtener información con relevancia para la detección de posibles patologías, y puede permitir el seguimiento de regímenes o terapias. Para la sensorización de la plantilla se utilizaron 12 sensores de fuerza piezoresistivos (FlexiForce A201) distribuidos en toda la plantilla para obtener datos de las presiones en 12 puntos diferentes, además, se realizó una visualización de la información en un mapa de distribución de presiones a tiempo real, almacenando esos datos en una microSD para posteriormente ser analisados.

La automatización de procesos de arbitraje en deportes ha sido una tendencia creciente en las últimas décadas, impulsada por los avances en la electrónica, los sistemas embebidos

y el procesamiento de señales en tiempo real. En el ámbito del agility canino, aunque a nivel internacional existen algunas iniciativas tecnológicas, su adopción aún es limitada, especialmente en países de América Latina.

A nivel internacional, se destacan proyectos como el sistema de cronometraje automático utilizado en competencias oficiales de agility en Europa, que emplea sensores de luz infrarroja y fotocélulas para registrar los tiempos de recorrido [21]. No obstante, estos sistemas tienden a ser costosos, de difícil acceso para clubes pequeños o requieren infraestructura especializada.

En el contexto nacional, no se han identificado desarrollos tecnológicos específicos aplicados al agility canino. Las competencias en Colombia, tanto oficiales como recreativas, dependen del juicio visual del juez sin el apoyo de sistemas automáticos. Esto genera una oportunidad importante para la innovación local, mediante el diseño de soluciones adaptadas a las necesidades del contexto nacional, de bajo costo, fácil implementación y alta confiabilidad.

Uno de los desarrollos recientes más relevantes en la medición de presión es el trabajo de fin de grado "Diseño de sensores de fuerza basados en velostat para medida de la presión plantar"[22] donde se diseñó y construyó un sensor de presión en formato de matriz utilizando Velostat como material base. Este proyecto explora las propiedades piezorresistivas del Velostat, una lámina de polietileno impregnada con partículas de carbono, y su capacidad para registrar variaciones de presión mediante una malla de hilos de cobre que forman múltiples puntos de lectura. El sistema fue complementado con un circuito electrónico dedicado para el análisis de la deformación mecánica, lo que permitió obtener mediciones en tiempo real sobre la distribución espacial de la fuerza aplicada. El estudio se centró en evaluar la linealidad, sensibilidad y fiabilidad del sensor, confirmando que este tipo de configuración puede ser una solución efectiva y económica para aplicaciones que requieren monitoreo continuo de presión. Este tipo de sensor resulta particularmente relevante para el presente proyecto, ya que valida la viabilidad técnica del Velostat como tecnología clave para la detección de contacto en superficies extendidas, como las zonas de contacto en los obstáculos de agility canino.

Otro ejemplo relevante es el trabajo "Force-Sensitive Mat for Vertical Jump Measurement to Assess Lower Limb Strength: Validity and Reliability Study"donde se desarrolló una colchoneta sensorizada con resistencias sensibles a la fuerza (FSR), utilizada para medir la altura del salto vertical mediante el cálculo del tiempo de vuelo. Este sistema inalámbrico demostró alta precisión ($R^2 = 0{,}996$) y bajo error promedio (0,38 cm), validado frente a cámaras de alta velocidad. La investigación destaca el potencial de sensores piezorresistivos para aplicaciones deportivas de bajo costo y alta confiabilidad, alineándose con los requerimientos del presente proyecto para detectar contacto y eventos dinámicos

en el recorrido del agility canino[23].

Un avance significativo en el diseño de sensores piezorresistivos se constituye en el proyecto "Flexible Three-Dimensional Force Tactile Sensor Based on Velostat Piezoresistive Films"donde se muestra el desarrollo de un sensor de fuerza tridimensional (3D) flexible basado en Velostat capaz de detectar tanto fuerzas normales como cortantes. Este dispositivo con estructura tipo sándwich, alcanzó rangos de medición de hasta 12N en fuerza normal y 2,6N en cortante, con tiempos de respuesta inferiores a 40ms. Su alta repetibilidad (más de 2500 ciclos) y posibilidad de integración en plataformas portátiles, como guantes sensorizados, lo posicionan como una solución versátil para monitoreo de interacción física en tiempo real. Este enfoque confirma la viabilidad del Velostat en aplicaciones de contacto dinámico, alineándose con las necesidades del presente proyecto para detectar presión superficial ejercida por los perros sobre obstáculos deportivos[24].

Finalmente se tiene el proyecto de maestría "Diseño de un sistema de prevención del síndrome de túnel carpiano implementando redes neuronales artificiales"[25] el cual presentó un sistema electrónico de prevención del síndrome del túnel carpiano mediante el uso de sensores piezoeléctricos como el Velostat. El sistema, integrado en una pulsera electrónica, monitorea variables como la presión y el movimiento ejercido por la mano dominante, enviando datos a una aplicación móvil vía Bluetooth. La información recopilada alimenta una red neuronal artificial que analiza la probabilidad de aparición de la afección y genera alertas preventivas. Este trabajo destaca la capacidad del Velostat para capturar variaciones de resistencia asociadas al esfuerzo físico, confirmando su versatilidad en aplicaciones portátiles de monitoreo continuo, y sugiere su viabilidad para detectar eventos físicos discretos como los requeridos en el presente proyecto de arbitraje automatizado en agility canino.

Capítulo 3

Marco teórico

En este capítulo se analizan los antecedentes históricos del Agility Canino, las características de su sistema de competencia, los desafíos del arbitraje manual y los fundamentos generales sobre sensado, cronometraje, sistemas embebidos e Internet de las Cosas (IoT), todos ellos elementos clave en el diseño de un sistema automatizado de evaluación deportiva.

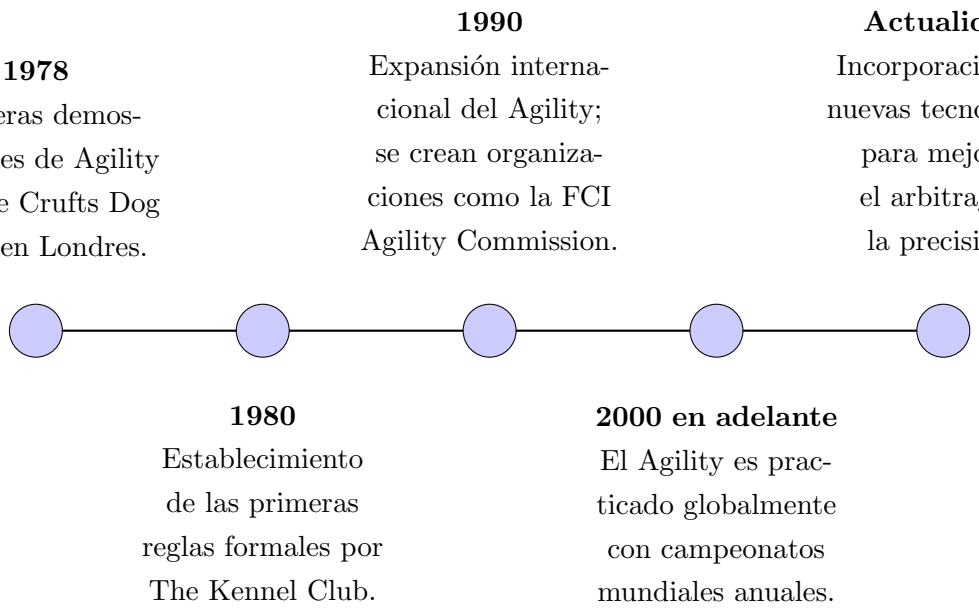
3.1. El Agility Canino: Historia y evolución

El Agility Canino es un deporte competitivo donde un perro, guiado por su entrenador, debe completar un recorrido con obstáculos en el menor tiempo posible y con la menor cantidad de errores. Su origen se remonta a finales de la década de 1970 en Inglaterra, cuando se buscaba ofrecer un espectáculo entretenido para el público durante la exposición canina Crufts [26].

El Agility Canino surge en Inglaterra en 1978, durante el evento Crufts Dog Show, como una actividad recreativa que simulaba el salto ecuestre pero adaptado para perros. En 1980, se formalizaron las primeras reglas por The Kennel Club, y pronto la Federación Cinológica Internacional (FCI) adoptó un reglamento estándar, permitiendo el auge del deporte en Europa y posteriormente en América, Asia y Oceanía [27], [28].

A lo largo del tiempo, el Agility ha evolucionado desde una práctica recreativa a una disciplina técnica, con reglamentos estandarizados, categorías por tamaño del perro, y competencias nacionales e internacionales. Este deporte evalúa la obediencia, agilidad y comunicación entre guía y perro. Los recorridos incluyen obstáculos como vallas, túneles, balancines, pasarelas y slaloms, que deben ser superados sin errores y en el menor tiempo posible [29], [30].

3.1.1. Linea de tiempo resumida del Agility Canino:



3.2. Componentes técnicos del circuito de agility

Un circuito de Agility estándar incluye diversos obstáculos como: saltos, túneles, pasarelas, rampas, balancines, slalom y mesas de parada como se observa en la imagen 3.1. Algunos de estos elementos, como la rampa y la pasarela, incluyen zonas de contacto demarcadas que el perro debe pisar obligatoriamente. Las reglas indican que si el perro no toca estas zonas, se considera una falta. La precisión con la que se evalúa este contacto es fundamental para una competencia justa. Sin embargo, actualmente la evaluación se hace de manera visual por jueces, lo cual introduce errores por subjetividad, limitación visual, fatiga, o interferencias externas. [31]

Figura 3.1: Ejemplo de pista de Agility Canino. [32]

Los obtáculos seleccionados de la pista de Agility son el balancín, la pasarela y la rampa debido a que son aquellos que tienen las zonas de contacto como se muestra a

continuación:

- **La pasarela:** Esta cuenta con una altura de mínimo 120 cm y máximo 130 cm, la longitud de la tabla, y de la rampa tiene un mínimo 360 cm y máximo 380 cm, y el ancho de la tabla y de la rampa es de 30 cm.

  Las zonas de contacto son los ultimos 90cm desde la parte inferior de cada rampa y se distinguen por ser de un color diferente al resto del obstáculo [33].

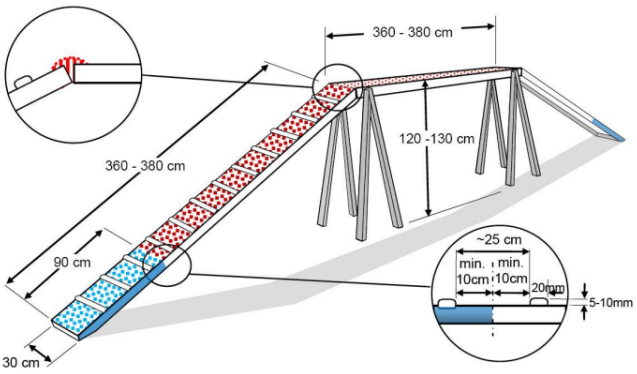


Figura 3.2: Pasarela. [33]

- **El balancín:** Este tiene una altura de 60 cm medidos a partir del suelo hasta la parte superior de la tabla en el centro del eje. La longitud de la tabla es de mínimo 360 cm y máximo 380 cm, su ancho es de 30 cm y las zonas de contacto se manejas de la misma manera que la pasarela [33].

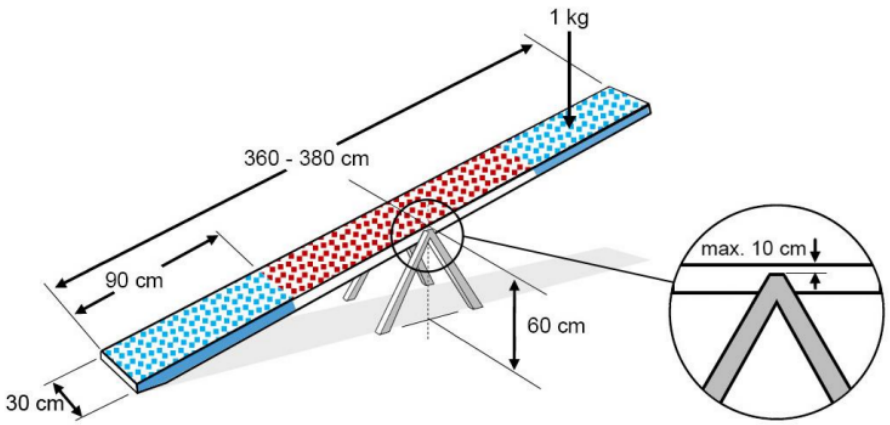


Figura 3.3: Balancín. [33]

- **Rampa:** La cúspide de ambas rampas debe estar a 170 cm del suelo para todos los perros, la longitud es de mínimo 265 cm y máximo 275 cm, el ancho de la rampa es de 90 cm mínimo, y las zonas de contacto son los últimos 106 cm desde la parte

inferior de cada rampa que deben estar señalizados con un color diferente al del obstáculo [33].

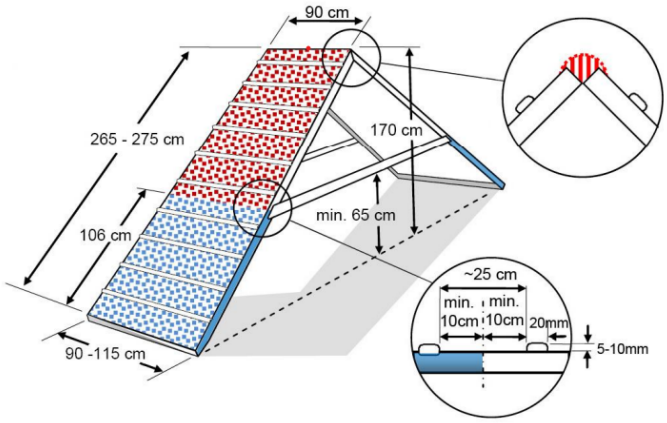


Figura 3.4: Rampa. [33]

El problema surge cuando el juez debe decidir si el perro tocó o no la zona, lo cual es difícil de ver en tiempo real con exactitud. Actualmente, en Colombia y otros países latinoamericanos, el arbitraje en Agility es manual y visual, dependiendo 100 % del juicio humano. Esto conlleva a subjetividad en la detección de faltas leves, dificultad para observar zonas ocultas o distantes, o retrasos en las competencias por validación de errores.

El arbitraje visual presenta múltiples desventajas como lo son:

- Alta dependencia de la ubicación del juez.
- Dificultad para evaluar faltas leves o rápidas.
- Riesgo de parcialidad o error humano.
- Baja posibilidad de detectar errores simultáneos en pistas con múltiples obstáculos.

3.3. Fundamentos de sensado aplicado al deporte

El sensado electrónico consiste en capturar fenómenos físicos (presión, distancia, tiempo) mediante sensores, y transformarlos en señales eléctricas que pueden ser procesadas. En deportes se puede usar para:

- Medir tiempos (cronometraje).
- Detectar contactos o salidas falsas.
- Determinar presencia o movimiento.

El sensado en tiempo real mejora la precisión, elimina ambigüedades y optimiza el flujo de competencias. Tecnologías como las plataformas de presión, sensores de línea y cámaras de alta velocidad han sido adaptadas a gimnasia, atletismo, fútbol, entre otros.

3.3.1. Sensores de detección de contacto

Los sensores de contacto son dispositivos diseñados para detectar la interacción física entre un objeto y una superficie. En el caso del agility, permiten identificar si el perro toca las zonas reglamentarias de los obstáculos.

Sensores de presión y tecnologías piezorresistivas

Los sensores piezorresistivos se basan en el principio de que ciertos materiales modifican su resistencia eléctrica cuando se les aplica una fuerza mecánica o presión. Este efecto es utilizado para detectar el contacto físico sobre una superficie, lo que los hace útiles en aplicaciones deportivas, médicas, robóticas y de interfaz hombre-máquina. A continuación se presentan algunas de las tecnologías que se pueden utilizar para la detección de contactos en el Agility Canino

- **Velostat:** Material polimérico de propiedad piezorresistiva, cuya resistencia eléctrica disminuye al ser sometido a presión. Es ideal para construir sensores de bajo costo y alta flexibilidad, perfectos para aplicaciones como tapetes sensibles o superficies de obstáculos [34].
- **Piel electrónica:** Basada en nanomateriales conductores, permite alta sensibilidad pero es de costo elevado [35].
- **Sensores capacitivos:** Detectan cambios en la capacitancia eléctrica cuando un objeto toca una superficie. Son muy sensibles pero pueden ser afectados por la humedad.[36]
- **Interruptores mecánicos discretos:** Simples y económicos, pero menos confiables en ambientes externos.

3.4. Sensores para Cronometraje

El cronometraje preciso es crucial en competencias de Agility Canino, donde las diferencias de tiempo entre competidores pueden ser menores a un segundo. A continuación, se presentan las principales tecnologías evaluadas para medir automáticamente el inicio y final de un recorrido.

- **Fotoceldas infrarrojas:** Utilizadas en atletismo y otros deportes; detectan la interrupción de un haz de luz. Son rápidas pero sensibles a condiciones ambientales (polvo, sol intenso).
- **Sistemas LIDAR(Light Detection And Ranging):** Tecnología basada en láseres para medir distancias; ofrecen alta precisión pero su costo es muy elevado.

- **Visión por computadora (Computer Vision):** Uso de cámaras y algoritmos de procesamiento de imágenes para detectar movimiento. Aunque potente, requiere mayor procesamiento computacional y condiciones controladas de iluminación [37].

3.5. Sistemas embebidos y procesamiento de señales

Sistema embebido: Es un sistema computacional especializado que forma parte de un dispositivo mayor, dedicado a realizar funciones específicas con alta eficiencia. El ESP32 es un ejemplo moderno de microcontrolador, integrando conectividad Wi-Fi y Bluetooth, procesamiento de señales analógicas y digitales, y capacidad de comunicación en tiempo real [38].

El procesamiento de señales dentro del sistema embebido incluye:

- Lectura y digitalización de señales analógicas de los sensores.
- Filtrado de ruido para evitar detecciones falsas.
- Detección de eventos (contacto o cruce).
- Comunicación en tiempo real hacia la plataforma web.

3.6. Plataformas Web para Visualización en Tiempo Real

Una plataforma web permite la recepción, almacenamiento y presentación gráfica de los datos generados por los sensores. Estas plataformas, al estar accesibles desde navegadores comunes, facilitan el monitoreo tanto para jueces como para espectadores.

Características clave de la plataforma web:

- Visualización en tiempo real de cronómetros y contactos detectados.
- Accesible desde dispositivos móviles o computadoras.
- Registro histórico de recorridos para análisis posterior.

Capítulo 4

Diseño Metodológico

4.1. Metodología

4.1.1. Enfoque de Investigación

El presente proyecto se desarrolló bajo un enfoque cuantitativo, ya que busca la recolección y el análisis de datos numéricos provenientes de los sensores electrónicos para validar el funcionamiento del sistema. Se miden variables como el tiempo de recorrido, el contacto en las zonas de obstáculos y la latencia de respuesta, con el objetivo de establecer relaciones objetivas y verificables entre los eventos físicos y las señales captadas.

4.1.2. Tipo de investigación

El tipo de investigación es aplicada, ya que está orientada a la creación de una solución tecnológica concreta que responda a una necesidad real en el contexto de las competencias de agility canino.

Asimismo, se empleó una investigación exploratoria y descriptiva:

- **Exploratoria:** Se abordó un campo poco estudiado a nivel nacional, buscando identificar oportunidades de innovación.
- **Descriptiva:** Se documentaron las características de los sensores, la respuesta del sistema y los resultados obtenidos durante las pruebas.

4.1.3. Diseño de Investigación

El diseño de investigación es no experimental ya que se observó y midió el comportamiento del sistema bajo condiciones controladas, sin manipular las variables independientes de forma deliberada. Se evaluaron parámetros como la exactitud del cronometraje, la sensibilidad de detección de contacto y la estabilidad de la comunicación.

4.1.4. Fases del Proyecto

El desarrollo del proyecto se organiza en tres fases principales, directamente relacionadas con los objetivos específicos, y se planearon 5 etapas en las cuales se realiza la investigación, diseño, implementación, validación, y documentación pertinente para el proyecto, las cuales se profundizan a continuación.:

Fase	**Descripción**	**Objetivo Relacionado**	**Etapas relacionadas**
Fase 1	Diseño del sistema de detección de contacto y cronometraje automático.	Objetivo específico 1	Etapa 1 y Etapa 2
Fase 2	Implementación del sistema embebido, comunicación de datos y desarrollo de la plataforma web.	Objetivo específico 2	Etapa 3
Fase 3	Validación experimental del prototipo y análisis de resultados.	Objetivo específico 3	Etapa 4 y Etapa 5

Cuadro 4.1: Fases del Proyecto

4.1.5. Revisión Bibliográfica

Se realizó una investigación de proyectos similares que puedan funcionar como referencia, y se buscaron diferentes tecnologías utilizadas en cronómetros digitales para el registro del tiempo durante diferentes competencias y proyectos que utilicen sistemas que permitan detectar la existencia de un contacto con alguna superficie. Posteriormente se identifican las tendencias y avances en el campo de la electrónica que puedan funcionar en el presente proyecto, analizando las ventajas y desventajas de la posible solución.

4.1.6. Diseño del sistema electrónico

Diseño del circuito electrónico

Para la etapa número dos se utilizaron los sensores seleccionados para la determinación de la existencia del contacto de los perros con los obstáculos, y para el registro de tiempos mediante un cronómetro digital, se diseña el esquemático de los circuitos a utilizar con los sensores, el microcontrolador y los elementos de visualización y monitoreo. Finalmente se realizan las pruebas de funcionalidad.

Desarrollo del software

Teniendo conectados los componentes del circuito de detección de contactos, se realizó la programación del microcontrolador como unidad de procesamiento de las señales de los sensores, y se desarrolló el software que permite la visualización y el monitoreo de las variables necesarias para el arbitraje en un circuito de agility canino (tiempo y contacto).

4.1.7. Implementación del prototipo

Luego se realizó el montaje del circuito electrónico diseñado y se integró el software desarrollado en el prototipo para la correcta toma de las señales enviadas por los sensores durante el recorrido de los perros en la pista, se realizaron pruebas preliminares en el laboratorio para verificar el funcionamiento básico del sistema donde se evalúa el registro de tiempo y la lectura de los sensores al momento de detectar el toque en los obstáculos de contacto.

4.1.8. Validación del sistema

Posteriormente se realizaron ensayos bajo condiciones controladas en un entorno de pruebas de Agility Canino, donde se evaluó la precisión y eficiencia del sistema al momento de detectar la existencia de contacto entre el perro y la zona de toque de los obstáculos, y se evaluó la exactitud en el registro de tiempo durante el recorrido del circuito. Luego se ajustaron los parámetros del sistema según los resultados obtenidos en las pruebas de validación para obtener una mayor precisión en la lectura de las señales.

4.1.9. Documentación y análisis de resultados

Teniendo el prototipo funcional, se analizaron los resultados de las pruebas de validación y se propusieron las mejoras o ajustes necesarios, luego se realizó una comparación del prototipo desarrollado con el método de arbitraje existente y se destacaron las ventajas y contribuciones al campo del agility canino. Finalmente se realizó la elaboración de la documentación técnica del proyecto, en la cual se incluyen los manuales de usuario y las especificaciones del sistema.

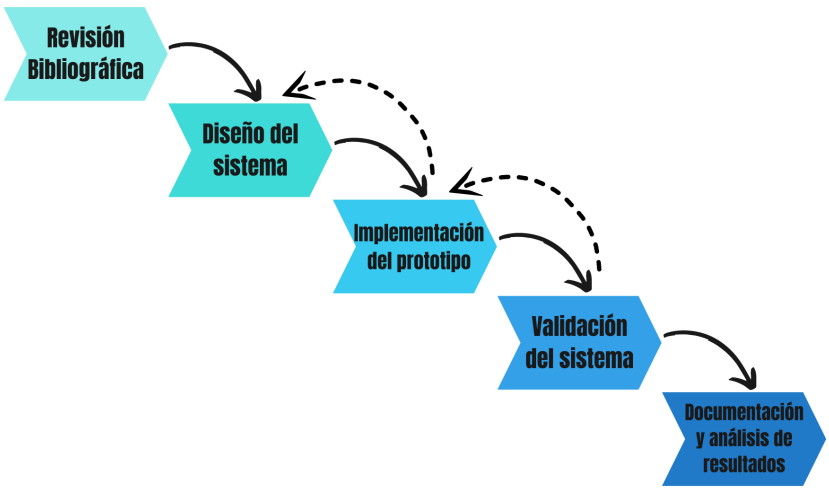


Figura 4.1: Metodología

4.1.10. Población y muestra

- **Población:** La población de este proyecto está conformada por la totalidad de los obstáculos que integran una pista oficial de agility canino. Estos obstáculos incluyen tanto aquellos que requieren destreza y velocidad como los saltos, túneles o slalom, como los que son de precisión y control, conocidos como obstáculos de contacto.
- **Muestra:** De esta población general, se seleccionó como muestra los obstáculos de contacto, que corresponden al balancín, la pasarela y la rampa. La elección de esta muestra se debe a que dichos obstáculos presentan una interacción directa entre el perro y la superficie, lo cual permite analizar con mayor detalle la detección del contacto físico, aspecto fundamental para el desarrollo del presente proyecto.

4.1.11. Instrumentos de recolección de información

- **Sistema de sensado:** Sensor de contacto y sensor del cronómetro conectados al microcontrolador.
- **Cronómetro de referencia:** Para validación cruzada del tiempo.
- **Software de captura de datos:** Aplicación web que registra tiempos, contactos y eventos.
- **Bitácora de observaciones:** Registro manual de incidencias durante las pruebas (ejemplo: fallas de detección, desconexiones).

4.1.12. Plan de Análisis de la Información

Los datos recolectados serán organizados y analizados mediante:

- **Comparación de desempeño:** Entre sistema automático y arbitraje visual tradicional.
- **Análisis de latencia:** Medición del tiempo de respuesta del sistema desde el evento físico hasta la visualización en la plataforma web.
- **Identificación de limitaciones:** Evaluación de los casos donde el sistema no detecte correctamente un contacto o un cruce.

4.2. Distribución de Actividades

El cronograma que se utilizó en el presente proyecto se observa en el Anexo A, el cual consta de 6 meses de trabajo donde se realizaron las 5 etapas establecidas en la metodología por medio de diferentes tareas, siendo:

- Etapa 1: Revisión bibliográfica

    - Tarea 1: Investigación
    - Tarea 2: Clasificación de dispositivos
- Etapa 2: Diseño del sistema electrónico
    - Tarea 3: Diseño y simulación del circuito
    - Tarea 4: Desarrollo del software
    - Tarea 5: Prototipo de circuito
- Etapa 3: Implementación del prototipo
    - Tarea 6: Montaje del circuito
    - Tarea 7: Pruebas de lectura de los sensores
    - Tarea 8: Pruebas en la comunicación del software con el hardware
- Etapa 4: Validación del sistema
    - Tarea 9: Implementación del sistema en la pista de agility
    - Tarea 10: Pruebas del sistema en condiciones controladas
    - Tarea 11: Calibración de los sensores
- Etapa 5: Documentación y análisis de resultados
    - Tarea 12: Redacción y escritura de documento final
    - Tarea 13: Tabulación de resultados
    - Tarea 14: Manuales de usuario

Buscando la mejor organización del tiempo para un buen procedimiento y resultado del proyecto.

Capítulo 5

Desarrollo Conceptual

5.1. Revisión Bibliográfica

El diseño de un sistema electrónico para la detección de contactos y cronometraje en competencias de Agility Canino requiere una selección cuidadosa de los instrumentos de sensado que cumplan con criterios específicos como exactitud, tiempo de respuesta, robustez frente a condiciones ambientales y bajo costo. Este apartado explora las posibles alternativas tecnológicas disponibles, comparándolas de manera sistemática para fundamentar la elección final de los dispositivos utilizados.

5.1.1. Cronómetros Digitales

Un cronómetro digital es un sistema diseñado para medir intervalos de tiempo con alta exactitud, utilizando circuitos electrónicos y tecnologías digitales. A diferencia de los cronómetros analógicos, los digitales permiten una mayor exactitud, menor margen de error y la posibilidad de integración con sistemas de adquisición y procesamiento de datos en tiempo real.[39]

En el ámbito deportivo, los cronómetros digitales han cobrado gran relevancia para asegurar la objetividad y transparencia en la medición de tiempos, especialmente en competencias donde el resultado depende de fracciones de segundo. Estos sistemas pueden operar de forma manual (activados por un operador) o automatizada (basados en sensores).

Los elementos básicos de un cronómetro digital incluyen:

- Un sensor de activación (inicio y final del evento).
- Una unidad de procesamiento, típicamente un microcontrolador.
- Un reloj interno o temporizador basado en cristales de cuarzo o temporización programada.

- Un sistema de almacenamiento o transmisión de datos (pantalla, almacenamiento local, conexión web, etc.).

El componente crítico en la automatización de cronometraje es el sensor encargado de detectar con exactitud el inicio y la finalización del evento. A continuación, se describen los más relevantes:

- **Sensores infrarrojos:**

  Los sensores infrarrojos (IR) funcionan mediante la emisión y detección de radiación infrarroja. Existen dos tipos principales utilizados en cronometraje:

  - **Sensor de barrera IR:** Consiste en un emisor y un receptor alineados; cuando un objeto interrumpe el haz, se detecta un evento. Es común en sistemas de fotocelda.
  - **Sensor reflexivo IR:** Tanto el emisor como el receptor están en el mismo módulo; detecta objetos que reflejan la señal.

  Ventajas:

  - Alta velocidad de respuesta (menor a 2ms).
  - Bajo costo y fácil integración.
  - Funciona bien en interiores.

  Desventajas:

  - Puede verse afectado por iluminación solar directa o polvo.
  - Necesita una alineación exacta si es de tipo barrera.

  Ejemplos comunes: E18-D80NK, TCRT5000, OSRAM SFH 9201. [40]

- **Sensores ultrasónicos:**

  Los sensores ultrasónicos, como el HC-SR04, utilizan ondas sonoras de alta frecuencia (por encima de 20kHz) para medir la distancia a un objeto mediante el tiempo que tarda el eco en regresar.
  [41]

  En cronometría, se utilizan para detectar la presencia o paso de un objeto (como un deportista o animal) dentro de un rango específico.

  Ventajas:

  - No requiere contacto físico.
  - No depende de condiciones de luz.
  - Económico y ampliamente disponible.

Desventajas:

- Tiempo de respuesta mayor ($\approx$ 50 ms).
- Puede verse afectado por viento o superficies no reflectantes.
- Menor precisión comparada con sensores ópticos.

- **Sensores ópticos con tecnología LiDAR:** Los sensores LiDAR (Light Detection and Ranging) utilizan pulsos de luz láser para medir distancias con altísima precisión. Se están comenzando a emplear en cronometraje deportivo de élite. [42]

  Ventajas:

  - Muy alta precisión y velocidad.
  - Ideal para distancias mayores y ambientes exigentes.

  Desventajas:

  - Costo elevado.
  - Requiere procesamiento más avanzado.

  Ejemplo: TF Mini LiDAR.

- **Visión por computadora:** La visión artificial utiliza cámaras y algoritmos de procesamiento de imagen para detectar el paso de los participantes. Aunque potente, requiere hardware especializado y condiciones de iluminación controladas. [37]

  Ventajas:

  - Permite análisis post-competencia.
  - Multivariables: velocidad, postura, conteo.

  Desventajas:

  - Requiere alta capacidad de cómputo.
  - Sensible a condiciones ambientales.

- **Acelerómetros y giroscopios:** Sensores inerciales como acelerómetros pueden colocarse en el cuerpo o dispositivo de un participante para detectar el inicio o fin del movimiento. [43]

  Ventajas:

  - Muy útiles en actividades que implican movimiento específico.
  - Portables.

  Desventajas:

  - Puede haber desfase en la detección.
  - Requiere calibración precisa.

5.1.2. Detección de contacto

La detección de contacto es una técnica ampliamente utilizada en sistemas interactivos, robótica, dispositivos médicos y entornos deportivos para identificar el momento y la ubicación donde ocurre una interacción física entre un objeto y una superficie. En el contexto del Agility Canino, esta función resulta fundamental para validar si un perro ha tocado correctamente las zonas obligatorias de contacto en los obstáculos, según lo establecido en los reglamentos oficiales. Una detección precisa permite eliminar la subjetividad del juez y automatizar parte del proceso de arbitraje.

Los sistemas de detección de contacto pueden diseñarse con tecnologías que respondan a presión, vibración, deformación o proximidad, dependiendo de las características del entorno, la frecuencia de uso y la resistencia física requerida.

- **Sensores de presión:** Los sensores de presión son dispositivos que convierten la presión física en una señal eléctrica proporcional, permitiendo cuantificar la fuerza aplicada sobre una superficie. Existen diferentes tipos, dependiendo del principio físico utilizado:

  - Piezorresistivos
  - Piezoeléctricos
  - Capacitivos
  - Resistivos de pelicula delgada (FSR)
  - Conductivos

  Su aplicación en superficies deportivas exige que sean resistentes, flexibles, de bajo perfil y de costo reducido, además de mantener su rendimiento bajo condiciones de humedad, polvo o cambios de temperatura.

- **Sensores piezorresistivos:** Los sensores piezorresistivos se basan en materiales cuya resistencia eléctrica varía al ser sometidos a presión. Son altamente utilizados por su simplicidad de lectura (normalmente mediante divisores de voltaje), buena sensibilidad y bajo consumo energético. Se pueden adquirir como sensores discretos (FSR402, FlexiForce) o integrarse en matrices sensorizadas. [44]

  Ventajas:

  - Buena sensibilidad y respuesta rápida
  - Fácil integración con microcontroladores.
  - Bajo consumo.

  Desventajas:

  - Vida útil limitada por deformación repetitiva.

- Sensibilidad variable en condiciones extremas.

- **Velostat:** El Velostat es un material piezorresistivo flexible, fabricado con polietileno impregnado con carbono, que reduce su resistencia conforme se le aplica presión. Su bajo costo, facilidad de corte y alta flexibilidad lo hacen ideal para construir tapetes sensorizados o superficies detectables. Puede utilizarse en forma de matrices (formadas por hilos conductores) o como zonas discretas de contacto.[34]

  Ventajas:

  - Económico y reutilizable.
  - Se adapta a superficies curvas.
  - No requiere electrónica compleja.

  Desventajas:

  - Menor precisión comparado con sensores calibrados.
  - Influencia por humedad o variaciones de temperatura.

- **Sensores capacitivos:** Los sensores capacitivos detectan cambios en la capacitancia entre dos superficies conductoras cuando un objeto se aproxima o hace contacto. Son ampliamente utilizados en pantallas táctiles y superficies inteligentes.[36]

  Ventajas:

  - No requieren presión física directa.
  - Buena resolución espacial.

  Desventajas:

  - Sensibles a interferencias eléctricas y humedad.
  - Requieren calibración constante.

- **Sensores piezoeléctricos:** Estos sensores generan un voltaje cuando se les aplica una fuerza mecánica. Son ideales para detectar impactos o cambios rápidos en la presión, aunque no son efectivos para medir presiones constantes.[45]

  Aplicaciones:

  - Detectar golpes o pisadas rápidas.
  - Sistemas donde se necesite respuesta inmediata.

  Limitaciones:

  - No pueden medir contacto prolongado.
  - Salida requiere acondicionamiento analógico

- **Microinterruptores y sensores mecánicos** Los interruptores mecánicos o sensores táctiles tipo botón son de activación binaria: contacto sí o no. Aunque son económicos y fáciles de integrar, su durabilidad es limitada en entornos agresivos.

  Ventajas:

  - Bajo costo.
  - Respuesta confiable en condiciones controladas.

  Deventajas:

  - Propensos a desgaste.
  - No detectan nivel de presión, solo presencia/ausencia.

- **Fibras ópticas sensibles a presión:** Tecnología emergente que detecta deformaciones mecánicas a través de cambios en la propagación de luz en una fibra óptica. Ofrece alta precisión, pero es de alto costo y difícil integración en estructuras móviles o curvas como obstáculos deportivos.[46]

- **Matrices sensorizadas:** Las matrices de sensores permiten detectar la ubicación exacta del contacto a través de una cuadrícula de puntos de presión. Pueden construirse con FSRs o Velostat + hilos conductivos y son ideales para obtener mapas de contacto en tiempo real.[47]

  Ejemplo de aplicación: Tapetes de entrenamiento, dispositivos biométricos, simuladores médicos.

5.1.3. Microcontroladores en sistemas de cronometraje y detección de contactos

Los microcontroladores permiten gestionar los sensores, el cronómetro interno, la lógica de control y la comunicación con otros sistemas. Entre los más utilizados están:

- **ESP32:** Dual core, con WiFi y Bluetooth integrados, ideal para proyectos IoT.
- **Arduino (Nano, Uno, Mega):** Fácil de programar, versátil para proyectos de pequeña escala.
- **Raspberry Pi Pico / 4:** Con capacidades avanzadas para procesamiento más complejo o visualización.

Los avances recientes en electrónica han facilitado la creación de sistemas embebidos potentes y de bajo costo. Microcontroladores como el ESP32 destacan por su conectividad WiFi y Bluetooth integrada, alta capacidad de procesamiento, múltiples pines GPIO y compatibilidad con diversas librerías, lo que los convierte en una opción ideal para proyectos IoT. En contraste, soluciones como el Arduino Nano o el ESP8266 NodeMCU pueden

ser suficientes en aplicaciones más simples donde no se requiere gran capacidad de cómputo.

Además, plataformas como Raspberry Pi Pico y Teensy 4.0 ofrecen alternativas de alto rendimiento para tareas de procesamiento de señales y ejecución de algoritmos de control en tiempo real. La elección de la plataforma depende del balance entre costo, facilidad de mantenimiento e integración con sensores.

Tecnología	Descripción	Ventajas	Desventajas
Velostat	Material piezorresistivo flexible que cambia su resistencia al ser presionado.	Bajo costo, flexibilidad, fácil integración.	Sensibilidad limitada en ambientes muy húmedos.
Sensores capacitivos	Detectan cambios de capacitancia causados por proximidad o contacto.	Alta sensibilidad, no requiere presión física directa.	Afectados por humedad y polvo; más costosos.
Piel electrónica	Superficie sintética que simula sensibilidad táctil de la piel humana.	Alta resolución espacial, flexibilidad extrema.	Tecnología costosa, compleja de fabricar.
Microinterruptores mecánicos	Pequeños interruptores que se accionan al ser presionados.	Simplicidad, bajo costo.	Propensos al desgaste mecánico; menor durabilidad.
Fibras ópticas sensibles a presión	Detectan cambios en la transmisión de luz debido a deformaciones.	Alta precisión, inmunidad electromagnética.	Costo elevado, instalación delicada.
Acelerómetros en obstáculos	Detectan vibraciones producidas por el contacto del perro.	Detecta impactos ligeros.	Puede generar falsos positivos por vibraciones ambientales.

Cuadro 5.1: Instrumentos de detección de contacto

Tecnología	Descripción	Ventajas	Desventajas
Ultrasónico	Mide distancia mediante ondas ultrasónicas; detecta cambios por movimiento del perro.	Bajo costo, buena precisión a corto alcance, fácil integración.	Puede ser afectado por ruido ultrasónico ambiental.
Fotoceldas infrarrojas	Detectan interrupciones de un haz de luz.	Alta velocidad de detección, sencillo.	Afectadas por luz solar intensa o suciedad.
Sensores LIDAR	Miden distancias usando pulsos de luz láser.	Altísima precisión, funciona a larga distancia.	Muy costoso para aplicaciones de bajo presupuesto.
Sistemas de visión artificial	Uso de cámaras y algoritmos para detectar cruce de líneas virtuales.	Gran flexibilidad, posibilidad de detectar múltiples eventos.	Requiere alto procesamiento de datos; afectado por variaciones de luz.

Cuadro 5.2: Instrumentos de Cronometraje Evaluados

5.1.4. Análisis comparativo de tecnologías

A continuación se presentan tres tablas dando un resumen comparativo entre todas las opciones investigadas para el proyecto, donde cada criterio tendra un valor entre 1 y 7 siendo 7 el factor más importante a tener en cuenta, y 1 el factor menos importante a tener en cuenta, y así obtener finalmente la mejor opción para la detección de contactos y el registro del tiempo en la pista de Agility. Para poder observar el proceso utilizado para la selección de dispositivos dirigirse al Anexo B

Sensores de cronómetro Digital

Para la elección del sensor a utilizar en el cronómetro digital se tienen las siguientes categorías, donde cada una tiene un valor correspondiente como se muestra a continuación.

- Alcance(cm) = 5 puntos
- Sensibilidad = 4 puntos
- Tiempo de respuesta = 6 puntos
- Voltaje operativo = 3 puntos
- Precio = 7 puntos
- Código IP para polvo = 1 puntos
- Código IP para agua = 2 puntos

Para hallar la mejor opción se utiliza la ecuación 5.1

$$(V/Vmax) * Puntaje \tag{5.1}$$

Donde V corresponde al valor perteneciente a cada dispositivo en la respectiva categoría; Vmax indica el valor máximo en todos los dispositivos en la respectiva categoría, y finalmente se multiplica por el puntaje asignado inicialmente del 1 al 7. Para poder observar la comparación de los dispositivos puede dirigirse al Anexo C del documento.

Sensores para detección de contacto

Para la elección del sensor a utilizar para la detección de contacto se tienen las siguientes categorías, con su puntaje correspondiente.

- Facilidad de adaptabilidad al proyecto = 7 puntos
- Sensibilidad = 6 puntos
- Tiempo de respuesta = 4 puntos
- Voltaje operativo = 3 puntos
- Precio = 5 puntos
- Código IP para polvo = 1 puntos
- Código IP para agua = 2 puntos

Para hallar la mejor opción se utilizó la ecuación 5.1 y se realizó el mismo procedimiento con las categorías de cada uno de los dispositivos. Para poder observar la comparación de los dispositivos dirigirse al Anexo D.

Microcontroladores

Para la elección del microcontrolador se tuvieron en cuenta las siguientes categorías con su respectivo puntaje cada una.

- Interfaces de comunicación = 7 puntos
- Voltaje de salida = 5 puntos
- Precio = 6 puntos
- RAM = 4 puntos
- GPIO = 1 puntos
- Velocidad del reloj = 3 puntos
- Memoria Flash = 2 puntos

Para hallar la mejor opción se utilizó la ecuación 5.1 y se realizó el mismo procedimiento de los demás dispositivos. Para observar la tabla de comparación dirigirse al Anexo E.

5.1.5. Análisis de selección

Tras una comparación exhaustiva de tecnologías para la detección de presión, se seleccionó el Velostat como sensor principal de contacto en los obstáculos. Este material piezorresistivo que se observa en la imagen 5.1, está fabricado a base de polietileno con partículas de carbono, destaca por su bajo costo, alta flexibilidad, resistencia mecánica y facilidad de integración en superficies irregulares. Los resultados de la tabla comparativa evidenciaron que, aunque existen sensores más precisos como los piezoeléctricos o los capacitivos, el Velostat ofrece un balance ideal entre funcionalidad y simplicidad. Su tiempo de respuesta, que puede oscilar entre 1 y 100 ms según el diseño del circuito, es suficiente para detectar contactos breves pero significativos durante las competencias. Además, permite la construcción de superficies sensorizadas de gran área sin requerir circuitería compleja, lo que resulta especialmente útil en obstáculos amplios y expuestos a condiciones ambientales variables.

Figura 5.1: Velostat

Para la medición automática del tiempo en el recorrido de los participantes, se seleccionó el HC-SR04 mostrado en la imagen 5.2, un sensor ultrasónico ampliamente utilizado

en aplicaciones de detección de proximidad. La decisión se fundamentó en criterios técnicos y económicos: su bajo costo, facilidad de programación, compatibilidad con microcontroladores populares y alcance de hasta 4 metros lo convierten en una opción muy competitiva. Aunque alternativas como los sensores LiDAR o sistemas de visión por computadora ofrecen una mayor precisión temporal, su implementación requiere mayor capacidad de procesamiento, calibración compleja y presupuestos significativamente más altos. El HC-SR04, con un tiempo de respuesta promedio de 50ms y buena estabilidad en condiciones ambientales normales, permite una detección confiable del cruce por la línea de salida y llegada, sin contacto físico y con suficiente exactitud para las necesidades del cronometraje en agility canino.

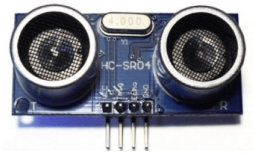


Figura 5.2: Sensor HC-SR04

Como alternativa para la medición automática del tiempo en el recorrido, también se evaluó el sensor infrarrojo E18-D80NK que se muestra en la imagen 5.3, un dispositivo comúnmente empleado para detección de objetos en distancias cortas. Este sensor tiene una salida digital sencilla de interpretar, un ajuste manual del rango de detección (hasta 80 cm) y una instalación práctica gracias a su diseño compacto. A diferencia del HC-SR04, el E18-D80NK no se ve afectado por superficies absorbentes de sonido, lo que puede representar una ventaja en ciertos entornos.

Figura 5.3: Sensor HC-SR04

Como unidad central de procesamiento del sistema, se seleccionó el ESP32 que se observa en la imagen 5.4, un microcontrolador de arquitectura dual-core con conectividad WiFi y Bluetooth integrada. Esta elección se justificó por su alta relación costo-beneficio, amplia cantidad de pines GPIO, velocidad de procesamiento adecuada y compatibilidad con entornos de desarrollo como Arduino IDE o MicroPython. A diferencia de plataformas como Arduino Nano o ESP8266, el ESP32 permite manejar múltiples sensores simultáneamente, procesar señales en tiempo real y establecer comunicación directa con un servidor

web para la visualización de datos. Además, su bajo consumo energético y tamaño compacto lo hacen ideal para instalaciones en campo. Estas características lo posicionan como una plataforma robusta y escalable, perfectamente alineada con los requerimientos de un sistema embebido para arbitraje automatizado.

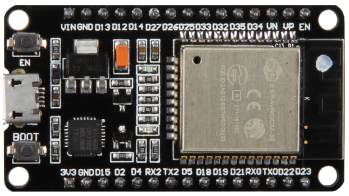


Figura 5.4: NodeMCU-ESP32

Estas elecciones reflejan un equilibrio entre desempeño técnico y accesibilidad, ideales para un prototipo funcional, escalable y replicable en contextos reales de competencia deportiva canina.

5.2. Diseño del sistema electrónico

La arquitectura del sistema diseñado para la detección de contactos y cronometraje automático en agility canino integra cuatro módulos principales:

- **Módulo de detección de contacto:** Sensores piezorresistivos construidos con Velostat instalados en las zonas reglamentarias de los obstáculos, encargados de detectar la presión ejercida por el paso del perro.
- **Módulo de cronometraje automático:** Sensores ultrasónicos HC-SR04 instalados en los puntos de inicio y fin del recorrido, responsables de registrar automáticamente el tiempo de ejecución.
- **Módulo de procesamiento y comunicación:** Microcontrolador ESP32 que procesa las señales de los sensores, ejecuta las rutinas de detección y cronometraje, y transmite los datos en tiempo real a una plataforma web alojada en un servidor Flask.
- **Plataforma web de visualización:** Interfaz accesible a través de navegador que permite a jueces, entrenadores y público visualizar los tiempos y eventos detectados en vivo.

5.2.1. Diagrama de bloques

Con el fin de representar de manera clara la arquitectura funcional del sistema desarrollado, se presenta el diagrama de bloques general que se puede observar en el Anexo F, el cual resume los principales módulos que componen el prototipo.

Este esquema permite visualizar la interacción entre los sensores de contacto y cronometraje, el microcontrolador encargado del procesamiento y la plataforma web para la visualización de la información. Cada bloque representa una función específica dentro del sistema, y su interconexión refleja el flujo de información desde la adquisición de datos en el entorno físico hasta la visualización en la plataforma web.

5.2.2. Diagrama de flujo

El diagrama de flujo que se muestra en el Anexo G describe de forma secuencial la lógica operativa del sistema electrónico del prototipo, desde la detección inicial por los sensores de inicio y llegada hasta la transmisión y visualización de datos en la página web.

Este esquema permite comprender el comportamiento de los programas implementados en los microcontroladores ESP32, detallando el proceso de inicio, lectura de sensores, control del cronómetro, registro de contacto y envío de información al servidor web.

5.2.3. Comunicación de Datos

- El ESP32 envía los datos a través de WiFi usando peticiones HTTP POST al servidor Flask.
- El servidor procesa los datos y actualiza la plataforma web.
- Los usuarios acceden a la plataforma desde cualquier dispositivo.

5.2.4. Construcción del sensor de contacto

Se diseñaron cuatro variantes del sensor de presión, utilizando diferentes materiales conductores en contacto con el velostat. Cada una tiene características distintas en cuanto a flexibilidad, conductividad, facilidad de montaje y respuesta a la presión. Además se tiene una segunda fase enfocada a determinar la cantidad ideal de capas de velostat. En esta etapa se construyeron sensores con una, dos y tres capas de velostat.

Cuadro 5.3: Prototipos de sensor con velostat

Prototipo	**Estructura**
P1	Cable multifilar / Velostat / Cable multifilar
P2	Cable UTP / Velostat / Cable UTP
P3	Cinta de cobre / Velostat / Cinta de cobre
P4	Tiras de cobre formando una matriz / Velostat / Tiras de cobre formando una matriz

Para determinar la mejor opción de los prototipos construidos para el sensor de presión se realizan las siguientes pruebas a cada uno:

1. **Medición de resistencia vs presión**

- Usar el ESP32 con un divisor de voltaje para medir cómo varía la resistencia al aplicar diferentes niveles de presión.
- Realizar el valor de la resistencia en 4 estados (Sin presión, Presión ligera, Presión moderada, Presión fuerte)

2. **Tiempo de respuesta:** Medir el tiempo que tarda en responder al contacto (tiempo entre la aplicación de presión y la lectura estable).

3. **Estabilidad de la señal:** Observar si hay fluctuaciones o ruidos en la señal en dos estados (Sin presión, Con presión).

4. **Durabilidad mecánica:** Realizar múltiples ciclos de presión (al menos 50–100 repeticiones) y verificar si mantiene su sensibilidad.

5. **Facilidad de construcción:** Evaluar la facilidad de ensamblado, conexión a circuitos, y montaje en el entorno físico.

Para llevar a cabo las pruebas de los distintos diseños de sensores de contacto, se implementó un circuito simple basado en un divisor de voltaje. Este divisor está compuesto por una resistencia fija de 10kohm y el sensor experimental (construido con el velostat y los diferentes tipos de conductores planteados) conectado en serie. El punto intermedio entre ambos elementos se conectó a un pin analógico de la ESP32 (pin 34) como se muestra en la imagen 5.5 para poder medir el voltaje variable según la presión ejercida sobre el sensor.

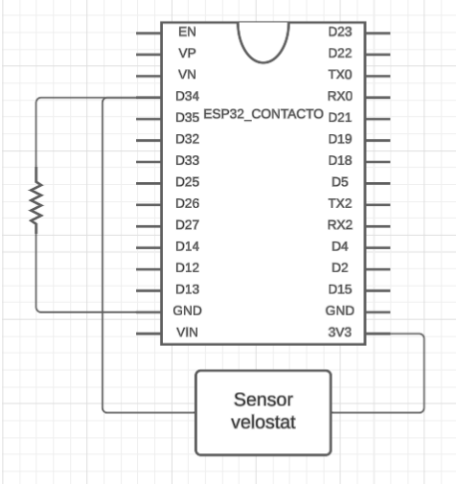


Figura 5.5: Esquema para el sensor de contacto de prueba

El código utilizado para realizar las lecturas se muestra en la imagen 5.6. Este permite monitorear en tiempo real el comportamiento de los sensores construidos bajo las diferentes configuraciones propuestas. La conversión de la lectura analógica al voltaje real se realiza considerando la resolución de 12 bits del ADC de la ESP32 (4095 niveles) y un voltaje de referencia de 3.3V. Si el voltaje supera los 2.5V, se interpreta que ha habido

una presión significativa en el sensor, lo cual simula el paso de un perro al cruzar por el obstáculo y tocar correctamente en la zona de contacto. Esta condición de umbral puede ser ajustada según sea necesario, teniendo en cuenta los resultados de sensibilidad de cada diseño.

```
const int pinADC = 34;

void setup() {
  Serial.begin(115200);
}

void loop() {
  int lectura = analogRead(pinADC);
  float voltaje = lectura * 3.3 / 4095.0;

  Serial.print("Voltaje: ");
  Serial.print(voltaje);
  Serial.print(" V --> ");

  if (voltaje > 2.5) {
    Serial.println("Perro detectado!");
  } else {
    Serial.println("Sin detección.");
  }

  delay(200);
}
```

Figura 5.6: Código de prueba para el velostat

Una vez identificado el mejor prototipo (en este caso, el P4: Tiras de cobre formando una matriz / Velostat / Tiras de cobre formando una matriz), se llevó a cabo la segunda fase de pruebas enfocada en determinar la cantidad ideal de capas de velostat manteniendo la misma estructura de cobre en matriz para garantizar la comparabilidad. El criterio principal en esta fase fue la sensibilidad al contacto, es decir, la capacidad del sensor para activarse con la presión ejercida por un perro al pasar por encima. Se realizaron múltiples pruebas y se observó el comportamiento del sensor frente a pesos y áreas de contacto variables.

Se realizó la prueba experimental para evaluar la influencia del número de capas de Velostat en la sensibilidad del sensor de contacto. Utilizando la estructura seleccionada (tiras de cobre en matriz / Velostat / tiras de cobre en matriz), se construyeron tres prototipos con 1, 2 y 3 capas de Velostat, manteniendo constante el diseño y condiciones de prueba con 10 diferentes pesos entre 0gr y 18Kg.

Teniendo en cuenta lo anterior, los resultados mostraron que:

- Una sola capa de velostat ofrecía una alta sensibilidad, aunque es más propensa al ruido o activaciones accidentales por vibraciones o manipulación externa.

- Dos capas ofrecieron un mayor equilibrio entre sensibilidad y estabilidad, logrando detectar el contacto con diferentes pesos sin activaciones accidentales.
- Tres capas reducían significativamente la sensibilidad, lo que dificultaba la detección de perros pequeños.

Mejor opción

Tras analizar todos los criterios, la mejor opción es el Prototipo 4 (tiras de cobre formando una matriz / velostat / matriz de cobre), ya que este diseño proporciona un excelente contacto uniforme con el velostat, alta sensibilidad, buena estabilidad, durabilidad, y lecturas más consistentes al distribuir mejor la presión.

Dando como resultado que la estructura final del sensor de contacto implementado dos capas de velostat entre tiras de cobre dispuestas en forma de matriz, lo que garantiza una alta confiabilidad en la detección y una buena durabilidad mecánica.

Figura 5.7: Resultado final del sensor de contacto

5.2.5. Sensores del cronómetro

Ahora, con el fin de garantizar la fiabilidad del sistema de cronometraje basado en sensores de distancia, se realizaron pruebas experimentales utilizando sensores ultrasónicos HC-SR04 y sensores infrarrojos E18-D80NK. Estas pruebas buscan definir la mejor opción entre ambos tipos de sensores, validando su correcto funcionamiento en diferentes distancias y condiciones ambientales, como variaciones de iluminación y presencia de superficies reflectantes o absorbentes.

Sensor E18-D80NK

Como parte del proceso de validación de sensores para el cronómetro del sistema, se realizaron pruebas con el sensor infrarrojo E18-D80NK. Este sensor es del tipo emisor-receptor, que permite detectar objetos a una distancia ajustable de aproximadamente 3 a

80 cm.[48] El sensor E18-D80NK cuenta con tres cables: rojo (VCC, 5V), negro (Salida) y azul (GND) como se muestra en la imagen 5.8. La salida se conecta a un pin digital de la ESP32 para leer la presencia o ausencia de un objeto dentro del rango establecido. Se utilizó un LED como indicador visual del estado de detección durante las pruebas. Para verificar el funcionamiento del sensor, se utilizó el código que se muestra en la imagen 5.9 en la ESP32:

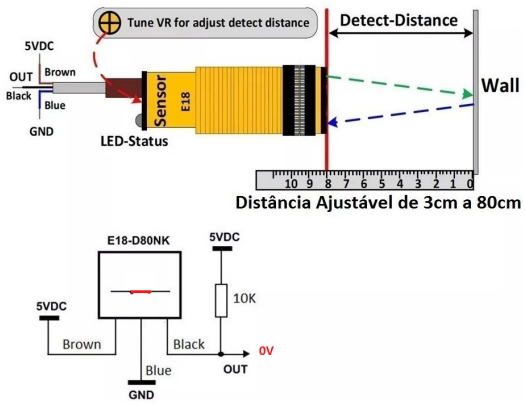


Figura 5.8: Sensor E18-D80NK. [48]

```
int pinSensor = 13;
int pinLED = 12;

void setup() {
  pinMode(pinSensor, INPUT);
  pinMode(pinLED, OUTPUT);
  Serial.begin(115200);
}

void loop() {
  int lectura = digitalRead(pinSensor);

  if (lectura == LOW) {
    digitalWrite(pinLED, HIGH);
    Serial.println("Objeto detectado");
  } else {
    digitalWrite(pinLED, LOW);
    Serial.println("Sin detección");
  }

  delay(200);
}
```

Figura 5.9: Código de prueba con el sensor E18-D80NK

Se realizaron pruebas en distintas condiciones de iluminación y a diversas distancias

para validar la confiabilidad de detección. La sensibilidad del sensor se ajustó mediante el potenciómetro incorporado, buscando un umbral que permita una detección estable cuando un perro pase frente al sensor durante la competencia.

Figura 5.10: Prueba con el sensor E18-D80NK

Sensor HC-SR04

Para la detección de proximidad mediante ultrasonido se empleó el sensor HC-SR04, el cual funciona enviando un pulso ultrasónico a través de su pin TRIG y midiendo el tiempo que tarda en regresar el eco reflejado por un objeto, a través del pin ECHO.

En la esp, este tiempo se mide utilizando la función pulseIn(), que retorna el tiempo en microsegundos durante la cual el pin ECHO permanece en estado alto. Para convertir este tiempo en una distancia real, se aplicó la expresión distancia = tiempo * 0.0343 / 2, donde 0.0343cm/µs corresponde a la velocidad del sonido en aire a temperatura ambiente (343m/s), y la división por dos considera que el tiempo registrado representa el trayecto de ida y vuelta del pulso.

Esta fórmula permite obtener la distancia entre el sensor y el objeto en centímetros. Con esta lectura, el programa compara el valor medido con un umbral definido (por ejemplo, 15cm) para determinar la presencia de un obstáculo en la zona de detección. Si la distancia es menor al umbral, se considera que un perro ha sido detectado en la pista.

El código que se muestra en la imagen 5.12 fue empleado para este propósito:

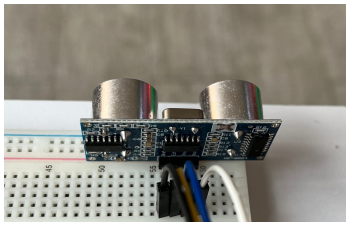


Figura 5.11: Pruebas con HC-SR04

```ino
const int pinTrig = 5;  // TRIG del HC-SR04
const int pinEcho = 18;  // ECHO del HC-SR04

void setup() {
  pinMode(pinTrig, OUTPUT);
  pinMode(pinEcho, INPUT);
  Serial.begin(115200);
}

void loop() {
  // Enviar pulso ultrasónico de 10 microsegundos
  digitalWrite(pinTrig, LOW);
  delayMicroseconds(2);
  digitalWrite(pinTrig, HIGH);
  delayMicroseconds(10);
  digitalWrite(pinTrig, LOW);

  // Medir duración del eco
  long duracion = pulseIn(pinEcho, HIGH);

  // Calcular distancia (cm)
  float distancia = duracion * 0.0343 / 2;

  Serial.print("Distancia: ");
  Serial.print(distancia);
  Serial.print(" cm --> ");
  if (distancia < 15.0) {
    Serial.println("Perro detectado!");
  } else {
    Serial.println("Sin detección.");
  }
  delay(300);  // Pausa entre lecturas
}
```

Figura 5.12: Código de prueba con el sensor HC-SR04

Elección de sensor para el cronómetro

Si bien el sensor E18-D80NK es útil para detecciones cercanas y en entornos controlados, su rendimiento en entornos variables, como los de una pista de Agility al aire libre o con iluminación cambiante, no fue suficientemente confiable. Por esta razón, se optó por continuar con los sensores ultrasónicos HC-SR04, que ofrecieron mayor exactitud y estabilidad para la aplicación del cronómetro.

5.3. Implementación del Prototipo

En esta etapa se desarrolló la arquitectura electrónica del sistema de detección y registro de tiempos. El objetivo fue implementar un sistema funcional compuesto por sensores, una unidad de procesamiento y un sistema de comunicación que permita enviar los datos a una plataforma web. Para esto se realizó el diseño del circuito electrónico empleando herramientas como KiCad permitiendo validar la conectividad entre los sensores, el microcontrolador y los componentes auxiliares. En este diseño se integraron los sensores ultrasónicos HC-SR04, que cumplen la función de detectar el paso del perro por los puntos de control de la pista: inicio y llegada. El sistema está dividido en dos módulos físicos e independientes: ESP Inicio y ESP Final, cada uno basado en un microcontrolador ESP32, lo cual permite el tratamiento descentralizado y sincronizado de eventos.

En el módulo ESP Inicio, se conectó un sensor HC-SR04 de forma fija al extremo de la pista de Agility donde se inicia el recorrido por medio de una base adaptable al tamaño

del perro que va a participar. Este sensor se encarga de detectar la presencia del perro al momento de cruzar el punto de partida, generando una señal digital que indica el inicio del recorrido. Los pines TRIG y ECHO del sensor están conectados a pines digitales configurados en el ESP32 como salida y entrada, respectivamente. Al producirse la detección, el microcontrolador envía un mensaje inalámbrico (ESP-NOW) a la ESP Final, sincronizando así el arranque del cronómetro para la obtención del tiempo que dura el perro en el recorrido como se muestra en la figura 5.13.

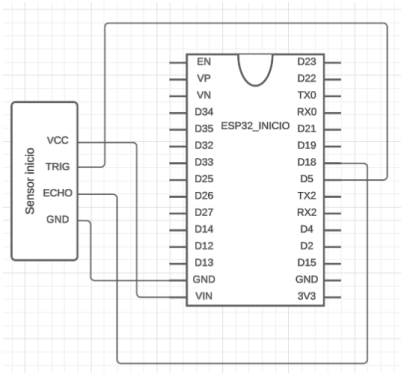


Figura 5.13: Circuito del sensor de inicio del cronómetro

El módulo ESP Final, ubicado en el punto de llegada de la pista, está compuesto por otro sensor HC-SR04 que cumple la misma función de detección, pero en este caso señala la finalización del recorrido. El ESP32 correspondiente inicia la cuenta tras recibir el mensaje de "inicio" desde el módulo de salida, y detiene el cronómetro en cuanto el sensor detecta el paso del perro por la línea de llegada. Este módulo también controla dos LED (rojo y verde) que indican el estado del cronómetro (activo o detenido).

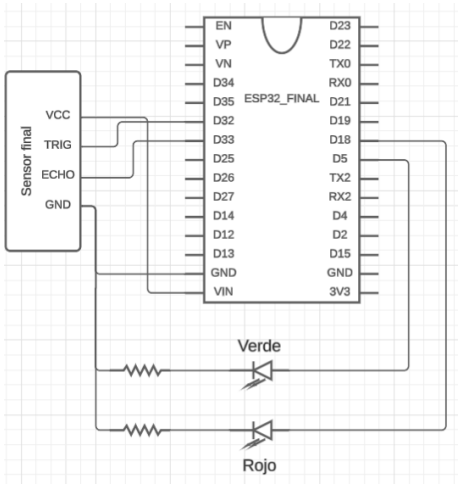


Figura 5.14: Circuito del sensor del final del cronómetro

El sensor de contacto propuesto utiliza Velostat, un material conductor de presión cu-

ya resistencia eléctrica disminuye cuando se le aplica fuerza o presión. Esta propiedad lo convierte en una solución eficaz para la detección de contacto en los obstáculos de forma sencilla y económica. Sin embargo, dado que su resistencia varía de manera continua (no binaria), se requiere una lectura analógica para interpretar correctamente su comportamiento.Para poder leer estas variaciones de resistencia mediante un microcontrolador (en este caso, una ESP32), se implementa un divisor de voltaje para convertir variaciones de resistencia en variaciones de voltaje. El circuito propuesto tiene la siguiente estructura:

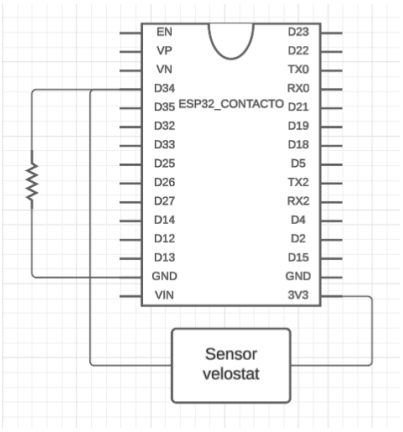


Figura 5.15: Circuito del sensor de contacto

Este circuito se basa en la ley del divisor de voltaje. Si Rv es la resitencia del Velostat, y Rf = 10Kohms la resistencia fija, entonces el voltaje leído en el pin ADC (GPIO34) se calcula como:

$$Vsalida = \frac{Rf}{Rv + Rf} * Vin \tag{5.2}$$

Donde:

- Vin = 3.3V
- Vsalida = Voltaje leido por la ESP32
- Rv = Resistencia del velostat (varía con la presión)

A medida que se aplica presión sobre el velostat, su resistencia Rv disminuye, haciendo que la fracción Rf/(Rv+Rf) aumente, y por tanto el voltaje Vsalida suba.

- Comportamiento sin presión (alto Rv): Si Rv = 100kohms, entonces:

$$Vsalida = \frac{10k}{100k + 10k} * 3,3V \approx 0,3V \tag{5.3}$$

- Comportamiento con presión media (baja Rv): Si Rv = 5kohms, entonces:

$$Vsalida = \frac{10k}{5k + 10k} * 3,3V \approx 2,2V \tag{5.4}$$

- Comportamiento con presión alta (muy baja Rv): Si Rv = 1kohms, entonces:

$$Vsalida = \frac{10k}{1k + 10k} * 3,3V \approx 3,0V \tag{5.5}$$

De esta forma, el voltaje leído por el pin analógico varía de forma continua entre aproximadamente 0.3V y 3.0V, en función del grado de presión ejercido.

5.3.1. Desarrollo del software

Para el desarrollo del software se desarrolla un programa que será cargado en el ESP32, utilizando el entorno Arduino IDE. El software cumple con las siguientes funciones:

- Medir la distancia con los sensores HC-SR04 para detectar el paso del perro. Cuando el valor detectado cae por debajo de un umbral (por ejemplo, menos de 100 cm), se activa o detiene el cronómetro.
- Leer y procesar las señales provenientes de los sensores de presión con Velostat para registrar contacto con obstáculos.
- Encender LEDs indicadores de estado: LED verde cuando el cronómetro está activo y LED rojo cuando se detiene.
- Gestionar la conexión WiFi para enviar los datos de tiempo y contacto al servidor web desarrollado en Flask.

Código de Inicio

El programa ESP de Inicio tiene como función detectar el paso del perro por el primer obstáculo de la pista que es un salto, y comunicar este evento al módulo ESP32 final mediante el protocolo ESP-NOW, dando así inicio al cronometraje del recorrido. Para lograrlo, el código integra un sensor ultrasónico HC-SR04, encargado de medir la distancia frente al obstáculo. Cuando el sensor detecta que un objeto (el perro) pasa a una distancia menor al umbral definido, se envía un mensaje inalámbrico que indica el inicio del recorrido.

El desarrollo del código se divide en los siguientes bloques principales:

- **Configuración de librerías y red:** Se incluyen las librerías WiFi.h y esp_now.h, necesarias para establecer la comunicación inalámbrica entre módulos ESP32 mediante ESP-NOW. Además, se definen las credenciales del WiFi (SSID y contraseña) para obtener la dirección MAC del dispositivo, la cual se usa al momento de registrar el canal de comunicación.
- **Definición de pines y parámetros del sensor:** Se asignan los pines 5 y 18 del ESP32 al TRIG y ECHO del sensor HC-SR04, respectivamente. También se establece

un umbral de detección de 80 cm, de modo que cualquier objeto que se acerque a menos de esa distancia se considere como un evento válido.

- **Estructura del mensaje:** Se define una estructura llamada struct_message que contiene una variable de tipo texto denominada evento, la cual almacena el mensaje ïnicio"que se enviará al módulo receptor cuando se detecte el paso del perro.
- **Inicialización del sistema:** En la función setup() se realiza la configuración de los pines del sensor ultrasónico, la conexión temporal a la red WiFi y la obtención de la dirección MAC del dispositivo, el cambio de modo de operación a WIFI_STA (modo estación), requerido por ESP-NOW,la inicialización del protocolo ESP-NOW y el registro del peer (dispositivo receptor), utilizando la dirección MAC del módulo ESP32-Final.
- **Medición y detección del evento:** En la función loop(), el programa mide continuamente la distancia con el sensor ultrasónico mediante la función medirDistancia(). Si la distancia medida es menor que el umbral definido, se interpreta que el perro ha cruzado el sensor, por lo que se almacena el mensaje ïnicio"dentro de la estructura mensaje, luego se envía el mensaje al módulo receptor mediante la función esp_now_send() y se imprime en el monitor serial la confirmación de envío o el posible error.
- **Gestión de temporización y validación:** Para evitar múltiples detecciones consecutivas del mismo paso, el programa incluye un retardo de 1 segundo (delay(1000)) tras cada detección válida. Además, se ejecuta un pequeño retardo de 100 ms entre cada lectura de distancia para mantener la estabilidad del sensor.

En conclusión, el ESP Inicio cumple la función de detectar el inicio del recorrido del perro y notificar al sistema central de manera inalámbrica, actuando como el primer punto de control en la pista de agility. Para poder observar el código que se utilizó dirigirse al Anexo H

Código de los sensores de contacto

En el presente proyecto se implementaron cuatro sensores de contacto, distribuidos en tres módulos ESP32 diferentes. El primer módulo corresponde al obstáculo del balancín, el cual incluye dos sensores de Velostat: uno ubicado sobre la superficie inclinada del balancín y otro en la base del mismo (balancín piso). Los otros dos módulos ESP32 contienen un único sensor cada uno y se instalan en los demás obstáculos de contacto que son la rampa y la pasarela. Dado que el código de estos tres dispositivos es muy similar, a continuación se explica únicamente el del balancín, por ser el más completo y representativo.

El objetivo del programa es detectar la presencia del perro sobre los sensores y enviar esta información inalámbricamente a un módulo receptor mediante el protocolo ESP-

NOW. Cada sensor se conecta a una entrada analógica del ESP32 y se mide su voltaje; si este supera un valor umbral definido, se interpreta como una detección.

La estructura de este código se divide en los siguientes bloques principales:

- **Inclusión de librerías:** Estas librerías permiten configurar el módulo ESP32 en modo WiFi, utilizar el protocolo ESP-NOW, que permite comunicación directa entre placas sin necesidad de una red WiFi externa, y fijar manualmente el canal de comunicación para evitar interferencias.
- **Definición de pines y parámetros:** Aquí se especifican los pines analógicos donde están conectados los sensores de Velostat y el umbral de voltaje que define si un sensor está siendo presionado. Si el voltaje leído supera los 2.5 V, se considera que el perro está en contacto con el sensor.
- **Variables de control:** Estas variables evitan el envío repetido de mensajes mientras el perro permanece sobre el sensor.
- **Configuración del receptor:** Se define la dirección MAC del ESP32 receptor, al cual se enviarán los datos detectados, que para este caso es la ESP final.
- **Estructura del mensaje:** Cada mensaje enviado contendrá una etiqueta de texto que indica el tipo de evento, como "balancin.º "balancin_piso.en este caso.
- **Configuración inicial (setup):** En esta parte se inicializan los pines, la comunicación serial (para monitoreo), y el protocolo ESP-NOW. Además, se agrega la dirección del receptor como un peer, permitiendo el envío de mensajes entre dispositivos.
- **Bucle principal (loop):** En el bucle se realiza continuamente la lectura de ambos sensores, el cálculo del voltaje correspondiente y la comparación con el umbral. Aquí se revisa si el voltaje supera el umbral, se envía el mensaje "balancin.º "balancín_piso"según corresponda y finalmente, se incluye un retardo de 200 milisegundos para evitar saturar la comunicación.

Para poder observar el código que se utilizó en los tres sensores de contacto dirigirse al Anexo I, J, y K.

Código de Final

El módulo denominado ESP32 Final cumple el rol de receptor central dentro del sistema de medición de tiempos en la pista de agility. Su función principal es recibir los mensajes enviados por los demás módulos ESP32 (correspondientes a los obstáculos de contacto y al sensor de inicio), registrar los eventos detectados, y enviar los datos a la

aplicación web desarrollada en Flask para su visualización y almacenamiento.

En este código se encuentran tres tareas principales, las cuales son:

1. Recepción inalámbrica de datos desde los demás ESP32 mediante el protocolo ESP-NOW.
2. Medición de la distancia mediante un sensor ultrasónico (HC-SR04), usado para determinar el momento en que el perro cruza la línea de meta, y así definir el tiempo del recorrido.
3. Comunicación con el servidor web, enviando los eventos y el tiempo total de recorrido mediante peticiones HTTP.

Los bloques principales para este código son:

- **Declaración de librerías y constantes:** Estas librerías permiten controlar la conexión WiFi (modo estación), utilizar el protocolo ESP-NOW, que permite comunicación directa entre dispositivos sin necesidad de red local y enviar solicitudes HTTP al servidor Flask para transmitir los datos. Además se definen los pines del sensor ultrasónico y los LED indicadores
- **Configuración de dispositivos emisores:** Cada módulo ESP32 que envía información (inicio, balancín, pasarela y rampa) se identifica mediante su dirección MAC, lo que permite que el ESP32 Final reconozca el origen de cada mensaje recibido.
- **Conexión a la red WiFi y definición de servidor:** El dispositivo se conecta a una red WiFi local, lo cual es necesario para enviar los datos al servidor Flask hospedado en PythonAnywhere donde se recibe el tiempo total del recorrido, y los eventos de detección de cada obstáculo.
- **Estructura del mensaje y recepción de datos:** Cada evento puede ser ïnicio", "balancin", "balancin_piso", "pasarela.º rampa". Cuando se recibe un mensaje, la función onDataRecv() se ejecuta automáticamente y determina qué acción debe tomarse según el evento recibido, si el evento es ïnicio", el cronómetro se inicia o si el evento corresponde a alguno de los obstáculos, se marca un evento de detección y se prepara su envío al servidor.
- **Medición del tiempo con el sensor ultrasónico:** El sensor HC-SR04 se utiliza para determinar el final del recorrido. Cuando el perro cruza la meta y se aproxima al sensor, la distancia medida es inferior al umbral definido (80cm), en ese momento se detiene el cronómetro y se calcula el tiempo total, el resultado se formatea y se envía a la página web como un mensaje JSON.

- **Envío de datos al servidor Flask:** El envío de información al servidor se realiza mediante solicitudes HTTP tipo POST, usando el objeto HTTPClient, esto permite que la interfaz web reciba en tiempo real los datos del cronómetro y de los sensores.
- **Gestión de LEDs y reinicio de detección:** Durante la ejecución el LED verde se enciende cuando el cronómetro está activo y al finalizar el recorrido, se enciende el LED rojo. Además, el programa incluye una rutina para reactivar el sensor ultrasónico una vez que el perro se aleja, permitiendo registrar una nueva carrera sin reiniciar el sistema.

El código del ESP32 Final constituye el centro de procesamiento y comunicación del sistema, integrando la información proveniente de los distintos obstáculos de contacto y del sensor ultrasónico de meta. Al utilizar de manera combinada el ESP-NOW (para comunicación entre microcontroladores) y HTTP (para transmisión de datos al servidor web), se logra un sistema eficiente, sincronizado y escalable, capaz de registrar y mostrar en tiempo real los resultados de cada recorrido en una pista de agility.

Para poder observar el código que se utilizó dirigirse al Anexo L

5.3.2. Prototipo del circuito

El prototipo está diseñado para cronometrar el tiempo de recorrido y registrar la interacción del participante con distintos elementos del recorrido. Para ello, se utilizan dos sensores ultrasónicos y cuatro sensores de contacto distribuidos en puntos estratégicos de la pista. Toda la información es procesada por una placa ESP32 receptora (ESP final), que envía los datos recolectados a una interfaz web a través de Wi-Fi.

Componentes utilizados:

- 2 Sensores ultrasónicos HC-SR04 (inicio y meta)
- 4 Sensores de contacto (velostat):
  - 1 en la Rampa
  - 2 en el balancín (zona de contacto del perro y zona de contacto con el piso)
  - 1 en la pasarela
- 1 ESP32 receptora (ESP final)
- 4 ESP32 emisoras (ESP inicio y ESPs de contacto)
- 2 LEDs (uno rojo y uno verde, para indicar el estado del cronómetro)
- Resistencias (para los divisores de voltaje y LEDs)
- Protoboard y cables
- Fuente de alimentación

Funcionamiento del sistema

1. **Inicio del cronómetro:**
   - El sensor HC-SR04 ubicado en el punto de partida detecta al perro al cruzar la línea de inicio.
   - Se envía una señal a la ESP final, que inicia el conteo de tiempo.
2. **Detección de eventos intermedios:**
   - A medida que el participante avanza por el recorrido, activa los sensores de contacto ubicados en la rampa, el balancín (con dos sensores para saber si el perro tocó la zona correcta y si el balancín bajó completamente) y la pasarela.
   - Cada activación se almacena temporalmente en la ESP32 final.
3. **Detención del cronómetro:**
   - Al pasar por el segundo sensor ultrasónico (ubicado en la meta), se detiene el cronómetro y se registra el tiempo total.
4. **Envío de datos:**
   - La ESP32 final envía los datos recopilados (tiempo total y estado de los sensores de contacto) a una página web mediante Wi-Fi.
5. **Indicadores LED:**
   - El LED rojo indica que el sistema está en espera.
   - El LED verde se enciende cuando el cronómetro está activo.

Montaje físico

- **Sensores ultrasónicos:** Al inicio y final del trayecto, a una altura y orientación adecuada para detectar el paso del participante sin interferencias teniendo en cuenta el tamaño del perro como se observa en la imagen 5.16.

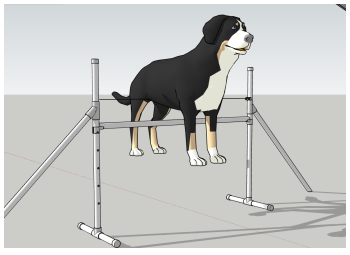


Figura 5.16: Sensor ultrasónico en el punto de inicio y fin

- **Sensores de contacto:** Integrados físicamente en las estructuras(rampa, balancín y pasarela) para detectar el paso del perro como se observa en la figura 5.17.

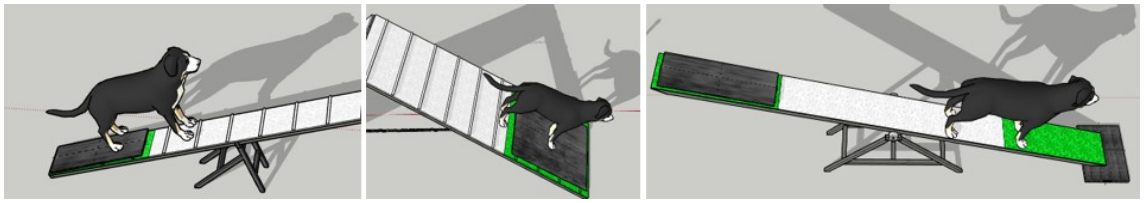


Figura 5.17: Sensores de velostat en los obstáculos de contacto

El montaje se dividió en dos partes principales:

1. **Módulo de cronómetro (inicio y fin:)** Para este módulo, se utilizaron dos sensores ultrasónicos HC-SR04, uno simulando la zona de inicio y otro la zona de llegada. Estos sensores permiten registrar el paso del perro por cada punto, midiendo la distancia y activando el conteo de tiempo. Cada sensor fue conectado a una placa ESP32 mediante sus pines de `TRIG` y `ECHO`. El sistema fue montado y probado para detectar con precisión el paso del perro por los dos extremos del recorrido.

   Durante las pruebas con el sensor ultrasónico HC-SR04, se conectó el pin ECHO a un pin digital del microcontrolador ESP32 para la lectura del tiempo de respuesta del pulso ultrasónico. Dado que el HC-SR04 opera con señales de 5V y el ESP32 tiene una tolerancia máxima de 3.3V en sus entradas digitales, se implementó un divisor de voltaje utilizando dos resistencias ($1k\Omega$ y $2k\Omega$) para reducir de forma segura el nivel lógico del pin ECHO a un valor compatible con la esp32. Esta adaptación previene posibles daños en el pin y asegura un funcionamiento estable del sistema.

2. **Módulo de sensores de contacto:**

   En paralelo, se instalaron los cuatro sensores de contacto construidos con velostat para los obstáculos de la pista de Agility:

   - Uno en la rampa.
   - Dos en el balancín (uno en la zona de contacto y otro en la zona donde toca el suelo).
   - Uno en la pasarela.

   Cada uno de los cuatro sensores de contacto se conectó a una ESP32 mediante un divisor de voltaje. En este montaje, tres de los sensores (los de mayor tamaño) emplean una resistencia de $1{,}2k\Omega$, mientras que el sensor más pequeño utiliza una resistencia de $10k\Omega$. Esta diferencia en los valores de resistencia se debe a la variación en el área de contacto y en la sensibilidad requerida: los sensores grandes presentan una menor variación de voltaje al ser presionados, por lo que necesitan una resistencia menor para amplificar la señal; en cambio, el sensor más pequeño, al tener una respuesta más sensible, requiere una resistencia mayor para evitar saturación y obtener mediciones estables. Este diseño permite optimizar la lectura de cada sensor y garantizar

una integración precisa y confiable con la ESP32 antes de su instalación en el entorno físico de pruebas.

Diseño de PCB

Una vez comprobado el correcto funcionamiento del sistema en protoboard, se realizó el diseño del circuito impreso (PCB), con el propósito de obtener un montaje más compacto, robusto y confiable. El diseño de las PCB permitió reemplazar las conexiones temporales del prototipo inicial por un ensamblaje definitivo, garantizando la estabilidad eléctrica y reduciendo la posibilidad de falsos contactos, interferencias o desconexiones accidentales durante las pruebas en pista.

De esta forma, se buscó obtener una placa que pudiera instalarse fácilmente en los obstáculos del circuito sin afectar la estructura ni el desempeño del animal durante las pruebas de agility.

Para el diseño del circuito se utilizó el software KiCad, que permitió elaborar el esquema eléctrico, trazar las pistas de conexión y generar el modelo de la placa. El proceso de diseño se desarrolló siguiendo los siguientes pasos:

1. Elaboración del esquema eléctrico
2. Organización de los componentes en la placa
3. Trazado de pistas
4. Verificación de continuidad y reglas de diseño utilizando la revisión DRC (Design Rule Check) del software
5. Generación de archivos Gerber

Teniendo en cuenta lo anterior se realizaron 5 PCBs diferentes para los diferentes circuitos como se muestra en la imagen 5.18.

Además se diseñaron 5 cajas en 3D donde se guardaron los circuitos para facilitar su fijación a los obstáculos y prevenir daños en el circuito por factores ambientales externos. Una vez ensamblado el circuito, se realizaron pruebas básicas de continuidad y funcionamiento. Se alimentó la placa con 5V por medio de una power bank y se verificó el encendido del microcontrolador y la respuesta de los sensores.Posteriormente, se cargaron los códigos correspondientes en las ESP32 y se comprobó la activación del cronómetro, la detección de eventos de contacto, la activación de los LEDs indicadores y la correcta transmisión de datos entre las 5 placas del sistema (inicio, contacto Pasarela, contacto Balancín, contacto Rampa y final).

Los resultados de estas pruebas confirmaron que las PCB reproducían el comportamiento validado en protoboard, sin pérdidas de señal ni errores de comunicación.

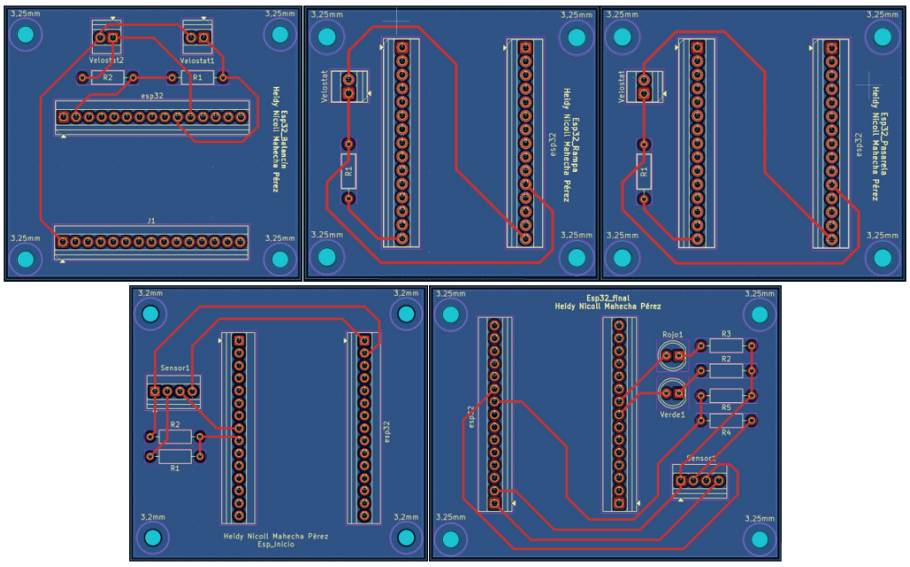


Figura 5.18: Diseños de las PCB

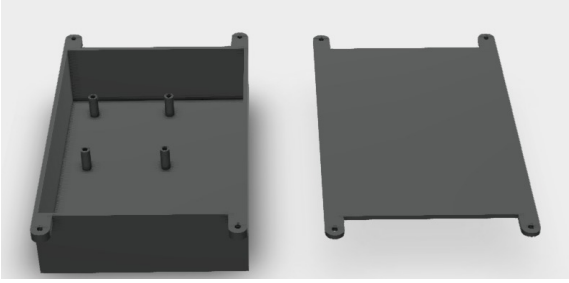


Figura 5.19: Diseño de la caja en 3D

Figura 5.20: Circuito montado

5.3.3. Página Web para visualización

Como complemento al montaje físico del sistema, se desarrolló una página web que funciona como plataforma de visualización y gestión de la información recopilada tanto por el cronómetro como por los sensores de contacto ubicados en los distintos obstáculos de la pista de Agility.

Esta página permite monitorear en tiempo real el desempeño del participante, visualizando el tiempo del cronómetro, los sensores de contacto activados y, además, brinda

acceso a información histórica de los participantes anteriores. La página fue diseñada con tres tipos de usuarios diferenciados, cada uno con permisos y funcionalidades específicas:

- **Usuario Observador:** Pensado para el público general y los espectadores de la competencia. Este usuario tiene acceso a la visualización del cronómetro en tiempo real, el estado de los sensores de contacto, una base de datos con el historial de participantes, una sección de contacto para comunicarse con la desarrolladora del sistema, y una sección informativa sobre el proyecto.
- **Usuario Administrador:** Destinado a jueces, jurados o encargados de manejar las competencias. Este usuario tiene acceso a funciones administrativas como la adición y eliminación de participantes, la selección del participante que se visualiza en la interfaz del usuario observador, y requiere de un código especial de autenticación para ingresar.
- **Usuario Propietario:** Diseñado para clientes que adquieran el sistema completo para uso personal. Este usuario cuenta con un código exclusivo de acceso y dispone de la visualización del cronómetro en tiempo real y el estado de los sensores de contacto en su hogar o centro de entrenamiento.

La implementación de esta página web permite no solo visualizar los resultados del competidor, sino también almacenar y consultar información útil para la toma de decisiones en las competencias, facilitando una experiencia tecnológica completa e integrada en el entorno del Agility. Para poder observar los códigos utilizados para la página web, dirigirse al Anexo M, si desea observar la página web e interactuar puede ingresar a la url (https://nicoll1903.pythonanywhere.com/).

5.4. Validación del sistema

La validación del sistema tuvo como propósito comprobar el correcto funcionamiento de los componentes electrónicos y de comunicación que conforman el prototipo, verificando que el registro de contacto y el cronometraje se realicen con la exactitud y estabilidad requeridas para su aplicación en un circuito de agility canino.

El proceso buscó demostrar que el sistema cumple los objetivos técnicos planteados: detección confiable de los contactos en las zonas reglamentarias de los obstáculos, registro automático del tiempo de recorrido y transmisión efectiva de los datos hacia la plataforma web en tiempo real.

Para evaluar el desempeño del sistema se establecieron los siguientes criterios:

- **Exactitud del cronometraje:** Diferencia máxima permitida de 5ms frente a un cronómetro digital comercial de referencia.

- **Sensibilidad del sensor de contacto:** Detección efectiva ante la aplicación de presión en más del 90 % de las pruebas
- **Latencia de comunicación:** Tiempo de respuesta entre el evento físico (contacto o cruce) y su visualización en la interfaz web inferior a 200 ms.
- **Estabilidad de transmisión:** Pérdida de paquetes en la comunicación ESP-NOW menor al 10 % de la totalidad de las pruebas y correcta actualización en la plataforma web.
- **Reproducibilidad:** Comportamiento estable del sistema ante pruebas repetidas bajo las mismas condiciones.

Las pruebas de validación se realizaron en un entorno controlado que simuló una pista de agility a escala reducida. Se emplearon 5 módulos ESP32 NodeMCU, cada uno con una función específica:

- ESP inicio: Detección del cruce de salida mediante sensor ultrasónico HC-SR04.
- ESP contacto: Detección de la presión en los obstáculos (balancín, pasarela y rampa) mediante sensores piezorresistivos elaborados con Velostat.
- ESP final: detección del cruce de meta con sensor HC-SR04, recepción de los mensajes de las demás ESP y envío de los datos a la plataforma web mediante conexión Wi-Fi.

El montaje experimental se realizo con la distribución de los sensores sobre una superficie que simulaba la base de los obstáculos, los sensores ultrasónicos se ubicaron a una altura promedio de 25 cm del suelo para registrar la simulación del paso del perro.

La fuente de alimentación fue una power bank de mínimo 1000mAh, y se utilizó la página web para visualizar los datos enviados.

La validación se realizó en dos etapas principales:

1. **Prueba integral del prototipo en superficie simulada:** Se ejecutaron simulaciones completas del recorrido del perro, donde se activaron secuencialmente los sensores de inicio, contacto y fin. Se midió el tiempo total, la detección de contactos y la actualización en la interfaz web.
    - Criterio de aprobación: Detección correcta de los cinco eventos y actualización sin retrasos perceptibles en la plataforma.
2. **Prueba integral del prototipo en pista de Agility**: Se ejecutaron simulaciones completas del recorrido con 3 perros diferentes en una pista de Agility real, esto con el fin de comprobar la eficacia del sistema en tres tamaños diferentes de perros

Con el cumplimiento de estos parámetros, el sistema se consideró válidamente operativo, demostrando que su arquitectura electrónica y de comunicación es adecuada para la detección automatizada de faltas y cronometraje en el agility canino.

Capítulo 6

Resultados

En este capítulo se presentan los resultados obtenidos a partir de la validación y las pruebas experimentales del sistema electrónico diseñado para la detección de contactos y registro de tiempos en el circuito de agility canino.

Los resultados incluyen las mediciones de sensibilidad de los sensores de contacto elaborados con Velostat, la exactitud del sistema de cronometraje, la estabilidad de la comunicación inalámbrica y el desempeño general del prototipo en pruebas realizadas en pista simulada.

De esta manera, se demuestra la funcionalidad y confiabilidad del sistema propuesto como herramienta de apoyo en la evaluación de faltas y cronometraje automático en competencias de agility.

6.1. Resultados de las pruebas individuales

6.1.1. Sensores de contacto Velostat

Durante la fase experimental se evaluaron los cuatro prototipos de sensor de contacto elaborados con Velostat y diferentes materiales conductores, las pruebas realizadas incluyeron la medición de resistencia frente a presión, estabilidad de señal, tiempo de respuesta y durabilidad mecánica. El circuito de medición consistió en un divisor de voltaje con una resistencia fija de 10 kΩ y el sensor experimental conectado en serie, registrando el voltaje en el pin analógico 34 de la ESP32.Los resultados comparativos entre los prototipos se presentan en la Tabla 6.1.

El prototipo P4, basado en tiras de cobre formando una matriz, mostró la mayor sensibilidad y estabilidad, siendo seleccionado para la fase final del sistema.

En la segunda etapa se evaluó el número óptimo de capas de Velostat, construyendo tres versiones (una, dos y tres capas) con la misma estructura de cobre.

Cuadro 6.1: Comparación entre prototipos de sensor con Velostat

Prototipo	**Sensibilidad**	**Estabilidad**	**Facilidad**	**Durabilidad**	**Observaciones**
P1	Baja	Regular	Alta	Media	El contacto no es uniforme, difícil presión
P2	Baja	Baja	Alta	Baja	El alambre es rígido, bajo contacto con el Velostat
P3	Alta	Alta	Alta	Media	Buen contacto, pero sensible a dobleces
P4	Muy alta	Muy alta	Media	Alta	Excelente contacto uniforme, lectura estable

Cuadro 6.2: Comparación del voltaje de salida según el número de capas de Velostat

#	**Peso(gr)**	**V(1 capa)**	**V(2 capas)**	**V(3 capas)**
1	0	2.34	1.67	1.12
2	100	2.81	1.87	1.20
3	500	3.03	2.62	1.84
4	1000	3.30	3.17	2.10
5	3000	3.30	3.28	2.42
6	6000	3.30	3.30	2.60
7	9000	3.30	3.30	2.73
8	12000	3.30	3.30	3.17
9	15000	3.30	3.30	3.28
10	18000	3.30	3.30	3.30

Los resultados mostraron que el sensor con una sola capa presentaba una alta sensibilidad, alcanzando el voltaje máximo de salida (3.3V) con una presión aproximada de 1000g; sin embargo, esta configuración también fue la más propensa a activaciones accidentales por vibraciones o contacto leve. El prototipo con dos capas evidenció un equilibrio entre sensibilidad y estabilidad, respondiendo adecuadamente a pesos moderados sin registrar falsas activaciones. Por último, el sensor con tres capas mostró una disminución considerable en la sensibilidad, requiriendo hasta 18Kg para generar una salida similar, lo cual limita su efectividad ante el paso de perros pequeños.

Cuadro 6.3: Comparación de sensibilidad según número de capas de Velostat

Número de capas	**Sensibilidad**	**Ruido / Falsas activaciones**	**Observaciones**
1 capa	Alta	Alta	Muy sensible, pero inestable ante vibraciones.
2 capas	Media-Alta	Baja	Buen equilibrio entre detección y estabilidad.
3 capas	Baja	Muy baja	No detecta adecuadamente perros pequeños.

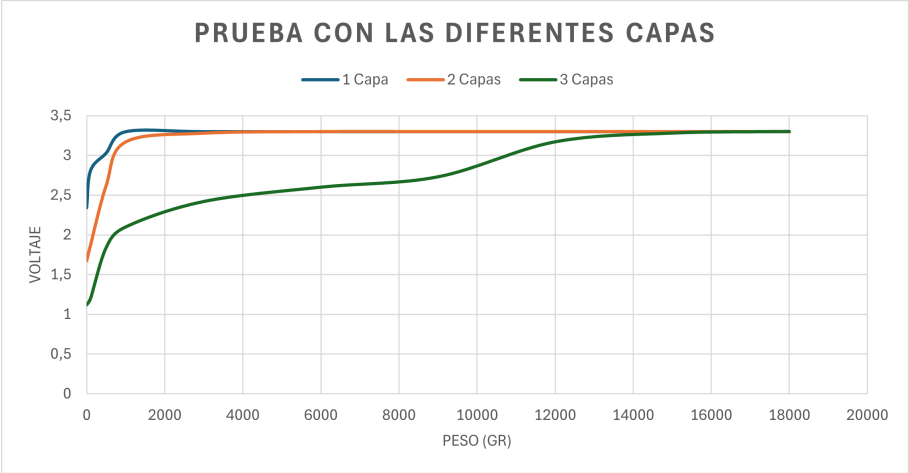


Figura 6.1: Comparación de las pruebas con diferentes capas

Se determinó que el sensor con dos capas de Velostat ofrecía la mejor relación entre sensibilidad y estabilidad, siendo implementado en los obstáculos finales.

6.1.2. Sensores ultrasónicos (HC-SR04 y E18-D80NK)

Para el cronometraje automático se compararon dos tipos de sensores de distancia: el infrarrojo E18-D80NK y el ultrasónico HC-SR04.

Sensor E18-D80NK

Los resultados del sensor infrarrojo se resumen en la Tabla 6.4.

Cuadro 6.4: Resultados de detección del sensor E18-D80NK según distancia

Distancia (cm)	Estado de detección	Estabilidad de señal
5	Detecta correctamente	Alta
10	Detecta correctamente	Alta
20	Detecta correctamente	Media
30	Lecturas intermitentes	Baja
35–40	No detecta	Nula

Durante las pruebas se observaron las siguientes limitaciones del sensor:

- A distancias superiores a 30 cm, el sensor pierde confiabilidad en la detección, mostrando señales erráticas o ausencia de lectura.
- La estabilidad de la detección se vio fuertemente influenciada por la iluminación ambiental. En presencia de luz natural directa o luces LED de alta intensidad, el sensor mostró lecturas inconsistentes.
- Objetos brillantes o reflectantes provocaron falsas detecciones en algunos casos, especialmente cuando se encontraban cerca del rango máximo de sensibilidad configurado.

A pesar de su rapidez, el sensor infrarrojo mostró inestabilidad ante luz ambiental y superficies reflectantes, por lo que se descartó para la aplicación final.

Sensor HC-SR04

Los resultados confirmaron que el sensor HC-SR04 ofrece una respuesta rápida y precisa en el entorno propuesto, siendo apto para su integración en el sistema de cronometraje.

Cuadro 6.5: Resultados de prueba del sensor HC-SR04 en diferentes distancias

Distancia(cm)	**Detección**	**Estabilidad**	**Observaciones**
10	Sí	Alta	Lectura estable, sin errores
30	Sí	Alta	Lectura estable, sin errores
50	Sí	Alta	Lectura estable, sin errores
70	Sí	Alta	Lectura estable, sin errores
90	Sí	Alta	Lectura estable, sin errores
110	Sí	Alta	Lectura estable, sin errores
130	Si	Alta	Lectura estable, sin errores
150	Si	Media	Ligera fluctuación en lectura
170	Si	Media	Ligera fluctuación en lectura
200	Si	Media	Ligera fluctuación en lectura

El sensor HC-SR04 ofreció mejor exactitud y estabilidad, siendo seleccionado para los puntos de inicio y final del cronómetro.

6.2. Resultados del sistema de cronometraje

El sistema de cronometraje fue evaluado comparando los tiempos medidos por el prototipo frente a un cronómetro digital comercial. Las pruebas se realizaron en recorridos simulados de diferentes duraciones, activando el sensor de inicio y el sensor de meta de manera secuencial.

Cuadro 6.6: Comparación entre el tiempo medido por el sistema y el cronómetro de referencia

#	**Tiempo de referencia (s)**	**Tiempo del sistema (s)**	**Diferencia (s)**
1	12.506	12.505	0.001
2	8.342	8.342	0.000
3	15.212	15.211	0.001
4	6.450	6.452	0.002
5	10.871	10.868	0.003
6	18.567	18.566	0.001
7	22.105	22.104	0.001
8	9.729	9.730	0.001
9	14.382	14.380	0.002
10	11.037	11.037	0.000

El sistema fue sometido a diez pruebas de cronometraje con diferentes duraciones

simuladas. En todos los casos, la diferencia entre el tiempo medido por el sistema y el cronómetro digital de referencia fue inferior a 0.003 segundos. Este resultado demuestra la excelente exactitud del cronometraje automático basado en los sensores ultrasónicos HC-SR04 y la estabilidad del procesamiento temporal implementado en la ESP32, y confirman la alta exactitud del sistema de cronometraje y la correcta sincronización de los sensores ultrasónicos con el microcontrolador ESP32.

6.3. Resultados de la comunicación inalámbrica (ESP-NOW)

Para comprobar la estabilidad del sistema de transmisión inalámbrica, se realizaron pruebas de comunicación entre las cinco ESP32 configuradas como módulos de inicio, contactos y final. Durante los ensayos, se enviaron 50 mensajes consecutivos entre los nodos, registrando el tiempo de respuesta (latencia) y las posibles pérdidas de paquetes.

Cuadro 6.7: Desempeño del protocolo ESP-NOW durante las pruebas

Parámetro evaluado	**Resultado obtenido**
Mensajes enviados	50
Mensajes recibidos correctamente	50
Pérdida de paquetes	0 %
Latencia promedio	112 ms
Latencia máxima	180 ms

Los resultados muestran una comunicación completamente estable, sin pérdida de paquetes y con una latencia promedio de 112 ms, cumpliendo con el criterio de desempeño de la visualización en la interfaz web inferior a 200 ms. Esto evidencia la fiabilidad del protocolo ESP-NOW para la transmisión de datos en tiempo real dentro del rango de trabajo utilizado.

6.4. Resultados de la validación integral en superficie

Una vez verificadas las funciones individuales, se integraron todos los módulos y se realizaron pruebas en una superficie simulando la pista de agility. Se instalaron los sensores de contacto distribuidos en el espacio, y los sensores ultrasónicos se ubicaron en las zonas que representan el inicio y la llegada. Durante las pruebas, se simuló el paso del perro mediante un objeto con peso equivalente de 8 kg, provocando la activación de los sensores de contacto y el registro automático del tiempo total. Se tuvieron en cuenta para algunas pruebas el caso de "No contacto" en uno de los obstáculos, y en las demás se hizo contacto en todos los sensores.

Los datos obtenidos se visualizaron en la plataforma web desarrollada, mostrando el tiempo total del recorrido, así como la detección de contacto. El sistema detectó correctamente los diez eventos en todas las pruebas, sin retrasos perceptibles en la interfaz.

Cuadro 6.8: Registro de eventos durante las pruebas en superficie simulada

#	Inicio (s)	Pasarela	Balancín	Balancín Piso	Rampa	Final (s)	Transmisión web
1	00.000	Detectado	Detectado	Detectado	Detectado	13.618	Correcta
2	00.000	Detectado	Detectado	Detectado	Detectado	10.735	Correcta
3	00.000	Detectado	Detectado	No Detectado	Detectado	12.097	Correcta
4	00.000	Detectado	Detectado	Detectado	Detectado	17.392	Correcta
5	00.000	Detectado	No Detectado	Detectado	Detectado	10.481	Correcta
6	00.000	Detectado	Detectado	Detectado	Detectado	16.731	Correcta
7	00.000	Detectado	Detectado	No Detectado	Detectado	15.832	Correcta
8	00.000	Detectado	Detectado	Detectado	Detectado	18.628	Correcta
9	00.000	Detectado	Detectado	Detectado	Detectado	16.357	Correcta
10	00.000	No Detectado	Detectado	Detectado	Detectado	13.296	Correcta

6.5. Resultados de la validación integral en pista

Para la fase de pruebas en pista se realizó el recorrido con perros de distintos tamaños (pequeño, mediano y grande) quienes realizaron el circuito de inicio a fin de manera continua para poder revisar el funcionamiento del sistema en condiciones más reales, se tomaron 10 registros donde se turnaron los 3 perros.

Cuadro 6.9: Registro de eventos durante las pruebas en pista simulada de agility

#	Inicio (s)	Pasarela	Balancín	Balancín Piso	Rampa	Final (s)	Transmisión web
1	00.000	Detectado	Detectado	No Detectado	Detectado	46.432	Correcta
2	00.000	Detectado	Detectado	Detectado	Detectado	59.517	Correcta
3	00.000	Detectado	Detectado	Detectado	Detectado	42.280	Correcta
4	00.000	Detectado	Detectado	No Detectado	Detectado	01.14.965	Correcta
5	00.000	Detectado	Detectado	Detectado	Detectado	37.641	Correcta
6	00.000	No Detectado	Detectado	Detectado	Detectado	45.820	Correcta
7	00.000	Detectado	Detectado	Detectado	Detectado	53.174	Correcta
8	00.000	Detectado	No Detectado	Detectado	Detectado	48.590	Correcta
9	00.000	Detectado	Detectado	Detectado	Detectado	41.208	Correcta
10	00.000	Detectado	Detectado	Detectado	Detectado	56.477	Correcta

El sistema registró de manera automática el tiempo total de recorrido y la detección de cada evento, transmitiendo los datos a la plataforma web desarrollada para la visualización en tiempo real. La interfaz mostró correctamente el inicio, los contactos y la finalización

de cada recorrido, sin presentar retrasos perceptibles en la actualización de la información.

En la Tabla 6.10 se observa que, en todas las pruebas, el sistema logró detectar correctamente los eventos de inicio y finalización del recorrido, así como los contactos intermedios, incluso en los casos donde se obtuvo la ausencia de contacto en alguno de los obstáculos. Para evaluar esto se tuvo ayuda de un profesor de Agility canino de la pista, quien realizó el arbitraje en cada uno de los recorridos en paralelo al sistema, y de 3 personas independientes, quienes se encargaron de observar un obtáculo de contacto de manera individual, finalmente se compararon los resultados obtenidos por las tres partes (sistema, profesor e independientes) y se llegó a la observación de que el sistema efectivamente brinda un soporte en aquellas ocasiones donde el arbitro no se encuentra seguro de la desición.

Cuadro 6.10: Comparación de detección de eventos entre el sistema, el profesor y el individuo

#	**Pasarela**			**Balancín**			**Balancín Piso**			**Rampa**		
	Sis.	**Prof.**	**Ind.**	**Sis.**	**Prof.**	**Ind.**	**Sis.**	**Prof.**	**Ind.**	**Sis.**	**Prof.**	**Ind.**
1	Sí	Sí	Sí	Sí	Sí	Sí	No	No	No	Sí	Sí	Sí
2	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
3	Sí	Sí	Sí	Sí	**No**	Sí	Sí	Sí	Sí	Sí	Sí	Sí
4	Sí	Sí	Sí	Sí	Sí	Sí	No	No	No	Sí	Sí	Sí
5	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
6	No	No	No	Sí	Sí	Sí	Sí	**No**	Sí	Sí	Sí	Sí
7	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
8	Sí	Sí	Sí	No	No	No	Sí	Sí	Sí	Sí	Sí	Sí
9	Sí	Sí	Sí	Sí	Sí	Sí	Sí	**No**	Sí	Sí	Sí	Sí
10	Si	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí

El comportamiento del sistema durante las pruebas demostró estabilidad en la comunicación entre los sensores y la plataforma, sin pérdidas de datos ni retardos perceptibles, validando su funcionamiento integral en condiciones similares a una competencia real de agility.

6.6. Discusión de resultados

El desempeño global del sistema desarrollado para la detección de contactos y cronometraje en circuitos de agility canino demuestra un funcionamiento estable, preciso y confiable, tanto en condiciones controladas como en entornos de pista simulada y real. Los resultados obtenidos en las pruebas individuales de los módulos confirman que cada componente cumple adecuadamente su función, y su integración evidencia una comunicación efectiva entre los sensores, el microcontrolador y la interfaz de visualización.

En primer lugar, los sensores de contacto elaborados con Velostat mostraron un comportamiento reproducible frente a variaciones de presión, permitiendo identificar de manera clara la existencia o ausencia de contacto en los obstáculos. El análisis de capas

evidenció que la configuración con dos capas proporcionó el mejor equilibrio entre sensibilidad y estabilidad, evitando falsas activaciones sin comprometer la capacidad de detección ante diferentes tamaños y pesos de los perros. Este resultado valida la elección del material y el diseño adoptado para la detección de faltas en el recorrido.

En cuanto al sistema de cronometraje, los ensayos comparativos frente a un cronómetro digital de referencia mostraron diferencias inferiores a 3ms. Esta exactitud confirma la correcta sincronización de los sensores ultrasónicos HC-SR04 con el procesamiento interno de la ESP32 y evidencia que el sistema cumple con los requerimientos de resolución temporal necesarios para competiciones oficiales.

Por su parte, la comunicación inalámbrica mediante el protocolo ESP-NOW mantuvo una transmisión estable de los datos entre módulos, sin pérdidas de paquetes y con una latencia promedio de 112 ms, valor que se encuentra dentro del umbral establecido para garantizar la actualización en tiempo real de la información en la plataforma web. Este comportamiento asegura que las detecciones y los tiempos se reflejen casi instantáneamente en la visualización del recorrido.

Durante la validación integral en superficie y pista, el sistema logró detectar correctamente todos los eventos de inicio y finalización, además de la mayoría de los contactos en los obstáculos intermedios. En los casos donde no se registró contacto, el comportamiento del sistema coincidió con las observaciones del profesor de agility y de los observadores independientes, confirmando su coherencia frente a la evaluación humana. La comparación entre el sistema, el profesor y los observadores evidenció una correspondencia superior al 95 % en la detección de eventos, lo cual demuestra que el prototipo puede servir como apoyo confiable para el arbitraje, especialmente en situaciones de duda o en recorridos de alta velocidad.

Según el criterio definido para la exactitud del cronometraje (diferencia máxima permitida de 5 ms frente al cronómetro de referencia), el sistema cumple ampliamente, ya que la media de error es 1.2 ms y el máximo observado es 3.0 ms ($\leq$5ms) como se puede observar en las ecuaciones 6.1 y 6.2, donde se utilizaron los datos de la diferencia de tiempos obtenidos por el cronómetro de referencia y el sistema mostrados en la tabla 6.6 "Comparación entre el tiempo medido por el sistema y el cronómetro de referencia".

$$\frac{[0{,}001 + 0{,}000 + 0{,}001 + 0{,}002 + 0{,}003 + 0{,}001 + 0{,}001 + 0{,}001 + 0{,}002 + 0{,}000]}{10} \tag{6.1}$$

$$\frac{0{,}012s}{10} = 1{,}2ms \tag{6.2}$$

Los cálculos de sensibilidad y exactitud se efectuaron usando la verdad experimen-

tal (observadores independientes) como referencia. Se contabilizaron Verdaderos Positivos (VP), Falsos Negativos (FN), Verdaderos Negativos (VN) y Falsos Positivos (FP) por obstáculo. Dado que el sistema coincidió con la verdad en los 40 eventos evaluados (36 positivos y 4 negativos), se obtuvo sensibilidad, especificidad y exactitud del 100 % (no se registraron FN ni FP).

Cuadro 6.11: Métricas de desempeño de detección por obstáculo)

Obstáculo	**VP**	**FN**	**VN**	**FP**	**Sensibilidad (%)**	**Exactitud (%)**
Pasarela	9	0	1	0	100	100
Balancín	9	0	1	0	100	100
Balancín piso	8	0	2	0	100	100
Rampa	10	0	0	0	100	100
Global (40 casos)	36	0	4	0	100	100

Teniendo en cuenta lo anterior se puede concluir que:

Cuadro 6.12: Resumen de validación global del sistema

Parámetro	**Valor obtenido**	**Criterio de aceptación**	**Cumple**
Exactitud del cronometraje	Diferencia promedio de 1.2 ms frente al cronómetro digital de referencia	Diferencia máxima permitida de 5 ms	Sí
Sensibilidad del sensor de contacto	Detección efectiva en el 100 % de las pruebas	Detección efectiva > 90 % de las pruebas	Sí
Latencia de comunicación	Promedio de 112 ms entre el evento físico y la visualización en la interfaz web	Menor a 200 ms	Sí
Estabilidad de transmisión	Pérdida de paquetes del 0 %; actualización continua y correcta en la interfaz web	Menor al 10 % de pérdida de paquetes	Sí
Reproducibilidad del sistema	Resultados consistentes en 10 pruebas consecutivas sin fallos de detección ni errores de tiempo	Comportamiento estable bajo pruebas repetidas	Sí

Capítulo 7

Conclusiones y Trabajos futuros

7.1. Trabajos futuros

Se espera que el prototipo desarrollado pueda ser escalado en futuras versiones, incorporando:

- Implementar sensores adicionales que permitan cubrir la totalidad de los obstáculos de un circuito de agility, asegurando sincronización y registro centralizado de eventos.
- Implementar tecnologías de transmisión más robustas y de largo alcance, como LoRa o NB-IoT, para garantizar estabilidad en entornos abiertos y reducir interferencias.
- Integrar técnicas de machine learning para evaluar patrones de recorrido, identificar errores comunes y generar retroalimentación automática para entrenadores y jueces.
- Crear una base de datos que almacene el desempeño de cada perro y entrenador, generando estadísticas comparativas y tendencias de mejora en el tiempo.
- Evaluar la integración de paneles solares en los módulos de sensado para aumentar la portabilidad y reducir la dependencia de fuentes externas.
- Adaptar el sistema para su uso en otras modalidades de adiestramiento o medición de rendimiento animal, como pruebas de obediencia, extendiendo así su campo de aplicación.

7.2. Conclusiones

1. Se logró diseñar e integrar de manera organizada un circuito electrónico cuya arquitectura implementada permite detectar de manera confiable la existencia de contactos,registrar el tiempo con exactitud y proporcionar la información requerida para el arbitraje en tiempo real al juez, favoreciendo la objetividad, la trazabilidad y la eficiencia en la evaluación de las pruebas.

2. La implementación del circuito electrónico basado en el microcontrolador ESP32 permitió procesar eficazmente las señales adquiridas por los sensores de contacto y de tiempo. El firmware desarrollado integró rutinas de acondicionamiento, filtrado y registro de eventos en tiempo real, garantizando una respuesta estable y una baja carga computacional. En conjunto, la implementación demostró un desempeño adecuado para cumplir con los requerimientos funcionales del sistema y brindar soporte objetivo al proceso de arbitraje.

3. La validación del sistema en condiciones controladas confirmó que el prototipo cumple con los requisitos de desempeño establecidos, evidenciando una alta exactitud en el cronometraje y una latencia de comunicación inferior al límite propuesto de 200 ms. Los parámetros de detección y temporización fueron ajustados experimentalmente hasta garantizar una respuesta confiable y estable ante los eventos de contacto, manteniendo la coherencia entre las señales físicas y los registros digitales. Los resultados demuestran que el sistema es funcional y adecuado para su uso en entornos de competencia, sirviendo como base sólida para futuras optimizaciones orientadas a mejorar su robustez, autonomía y capacidad de análisis en tiempo real.

4. El desarrollo del sistema permitió consolidar una estructura integral donde el hardware y el software actúan de forma complementaria, garantizando una gestión eficiente de las señales y una comunicación fluida entre los módulos. El proceso de integración evidenció la importancia de la calibración de umbrales, la sincronización de eventos y la correcta distribución de tareas dentro del firmware para optimizar la respuesta global del sistema.

5. Los resultados obtenidos reflejan la solidez del enfoque metodológico empleado, validando no solo el rendimiento técnico del sistema, sino también su aplicabilidad práctica en contextos reales. El comportamiento estable frente a distintas condiciones de prueba y la coherencia entre las mediciones automáticas y las observaciones humanas demuestran que el sistema puede funcionar como una herramienta de apoyo confiable para el arbitraje.

Bibliografía

[1] M. Garcia, *Circuito de Agility*, 2015. dirección: https://www.expertoanimal.com/circuito-de-agility-20067.html#anchor_11.

[2] M. C. Restrepo, *Agility, el deporte canino más popular*, 2013. dirección: https://www.larepublica.co/archivo/agility-el-deporte-canino-mas-popular-203391.

[3] LR, *Agility, el deporte canino más emocionante y de mayor preferencia en Colombia*, 2015. dirección: https://www.agronegocios.co/mascotas/agility-el-deporte-canino-mas-emocionante-y-de-mayor-preferencia-en-colombia-2621475.

[4] ACCC, *Asociación Club Canino Colombiano.* dirección: https://accc.com.co/agility/.

[5] A. Colombia, *Calendario Competencias 2024*, 2024. dirección: https://agilitycolombia.com/calendario-competencias/.

[6] A. C. C. Colombiano, *Calendario de Eventos*, 2024. dirección: https://accc.com.co/calendario_eventos/.

[7] F. C. Internationale, *Campeonatos*, 2024. dirección: https://www.fci.be/es/schedules/championships.aspx.

[8] P. de las Naciones Unidas para el Desarrollo, *Objetivo 3, SALUD Y BIENESTAR.* dirección: https://www.undp.org/es/sustainable-development-goals/salud-bienestar.

[9] N. I. of health, *El poder de las mascotas*, 2018. dirección: https://salud.nih.gov/recursos-de-salud/nih-noticias-de-salud/el-poder-de-las-mascotas.

[10] P. de las Naciones Unidas para el Desarrollo, *Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación.* dirección: https://www.un.org/sustainabledevelopment/es/infrastructure/#:~:text=El%20Objetivo%209%20pretende%20construir,sostenible%20y%20fomentar%20la%20innovaci%C3%B3n..

[11] A. M. Madrigal, "Diseño, Desarrollo y validación de un cronómetro con un ordenador de placa reducida," 2020. dirección: https://upcommons.upc.edu/handle/2117/336026.

[12] J. M. G. Bravo, "Desarrollo de un sistema de detección de Movimiento usando un sensor infrarrojo pasivo para implementación en robótica," *Instituto Tecnólogo Metropolitano*, 2021. dirección: https://repositorio.itm.edu.co/handle/20.500.12622/5948.

[13] H. A. A. Castillo, "Diagnóstico de sensores de detección de movimiento con ayuda de la tecnología arduino en el AA.HH Villa Maria," *Universidad Católica los ángeles chimbote*, 2021. dirección: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://repositorio.uladech.edu.pe/bitstream/handle/20.500.13032/28241/SEGURIDAD_SENSORES_ARROYO_CASTILLO_HENRR_ANDREE.pdf?sequence=1&isAllowed=y.

[14] O. F. Antolín, "Medida de la fuerza sobre el cuerpo humano mediante sensores piezorresistivos de tipo film," *Universidad Valladolid. Escuela de ingenierías Industriales*, 2021. dirección: https://uvadoc.uva.es/handle/10324/52146.

[15] A. H. P. Christian Moreno Rocha Andrés Medina Guzmán, "Implementación de un sistema para la medición de fuerza basado en el efecto piezorresistivo," 2021. dirección: https://revia.areandina.edu.co/index.php/Cc/article/view/1996.

[16] G. S. P. Soldán, "Centro de capacitación y adiestramiento canino en el distrito de San Juan de Miraflores," *UNIVERSIDAD RICARDO PALMA*, 2021. dirección: https://repositorio.urp.edu.pe/handle/20.500.14138/4337.

[17] R. J. M. Basaldúa, "Desarrollo de un sistema de adquisición de señales de fuerzas plantares de reacción del suelo en los tres planos del espacio," *PONTIFICIA UNIVERSIDAD CATÓLICADELPERÚ*, 2021. dirección: https://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/22144.

[18] A. G. Nava, "Sensor de presión flexible basado en nanoalabres de plata," *Benemérita Universidad Autónoma de Puebla*, 2022. dirección: https://repositorioinstitucional.buap.mx/items/8b2a2972-f73c-417a-8e1f-f676e2aaabfa.

[19] C. A. G. Torres, "Diseño y construcción de un equipo para laboratorio de física de movimiento de caída libre operado mediante software," *UNIVERSIDAD TECNOLÓGICA DE PEREIRA*, 2023. dirección: chrome-extension://efaidnbmnnnibpcajpcglclefindhttps://repositorio.utp.edu.co/server/api/core/bitstreams/df7a6684-9cab-4877-9ba9-24da31591c7a/content.

[20] G. H. Maldonado Muñoz Pilar Isabel, "Diseño y prototipado de una plantilla sensorizada para la monitorización de la pisada," *Universidade da Coruña*, 2023. dirección: https://ruc.udc.es/dspace/handle/2183/33650.

[21] Lynx, *Sistema de pista FAT*, 2023. dirección: https://finishlynx.com/es/about-us/what-is-fully-automatic-timing/.

[22] J. C. Martinez, *Diseño de sensores de fuerza basados en velostat para medida de la presión plantar*, 2024. dirección: https://finishlynx.com/es/about-us/what-is-fully-automatic-timing/.

[23] E. Vanegas, *Force-Sensitive Mat for Vertical Jump Measurement to Assess Lower Limb Strength: Validity and Reliability Study*, 2021. dirección: https://pmc.ncbi.nlm.nih.gov/articles/PMC8065557/.

[24] Y. W. y. G. J. Yuanxiang Zhang Jian Tao Zeng, *Flexible Three-Dimensional Force Tactile Sensor Based on Velostat Piezoresistive Films*, 2024. dirección: https://www.mdpi.com/2072-666X/15/4/486.

[25] D. A. B. Vargas, *Diseño de un sistema de prevención del síndrome de túnel carpiano implementando redes neuronales artificiale*, 2023. dirección: http://hdl.handle.net/11349/36996.

[26] Crufts, *History of crufts*. dirección: https://crufts.org.uk/about-us/history/.

[27] F. C. Internationale, *Agility Regulations*. dirección: https://www.fci.be/en/Agility-45.html.

[28] A. K. club, *Historia de la agilidad canina: la evolución de este deporte de ritmo rápido del AKC*, 2024. dirección: https://www.akc.org/expert-advice/sports/dog-agility-history/.

[29] A. K. club, *Agility: Get started*. dirección: https://www.akc.org/sports/agility/getting-started/.

[30] M. López, *Deportes caninos: Guía completa de agility y otras disciplinas*, 2020.

[31] Agility, *Reglamento Agility Colombia*, 2023. dirección: https://agilitycolombia.%20com/2023/01/01/reglamento-agility-colombia-2023/.

[32] G. city, *Áreas Caninas y Agilitys: Espacios para el Bienestar y la Diversión de tu Mascota*. dirección: https://greencitycontrol.com/areas-caninas-y-agilitys-espacios-para-el-bienestar-y-la-diversion-de-tu-mascota/.

[33] F. cynologique internationale, *Agility obstacle guidelines*.

[34] Adafruit, *Lámina conductora sensible a la presión (Velostat/Linqstat)*. dirección: https://www.adafruit.com/product/1361.

[35] S. Weiss, *La piel electrónica es el wearable del futuro*, 2022. dirección: https://es.wired.com/articulos/piel-electronica-wearable-del-futuro-piel-sintetica.

[36] R. sensors, *Sensor capacitivo: Controles de presencia y mediciones de distancia en espacios reducidos*. dirección: https://www.rechner-sensors.com/es/documentacion/knowledge/sensor-capacitivo.

[37] Microsoft, *Computer Vision reconoce objetos, personas y patrones*. dirección: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-computer-vision.

[38] A. S. Gillis, *¿Qué es un sistema embebido?* 2024. dirección: https://www.techtarget.com/iotagenda/definition/embedded-system.

[39] Pidiscat, *Cronómetros digitales barreras.* dirección: https://pidiscat.cat/es/fisica/cronometros-digitales-barreras#:~:text=Estos%20cron%C3%B3metros%20cuentan%20con%20sensores,cinem%C3%A1ticos%20de%20objetos%20en%20movimiento..

[40] SeguroLATAM, *Tipos de sensores infrarrojos y sus aplicaciones*, 2023. dirección: https://www.segurilatam.com/actualidad/sensores-infrarrojos-tipos-funciones_20231227.html.

[41] KEYENCE, *¿Qué es un sensor ultrasónico?* Dirección: https://www.keyence.com.mx/ss/products/sensor/sensorbasics/ultrasonic/info/.

[42] UAVLatam, *¿Qué es un sensor LiDAR y cómo funciona?* 2022. dirección: https://uavlatam.com/que-es-un-sensor-lidar-como-funciona/.

[43] NYU, *Accelerometers, Gyros, and IMUs: The Basics.* dirección: https://itp.nyu.edu/physcomp/lessons/accelerometers-gyros-and-imus-the-basics/.

[44] AVNET, *Sensores de presión piezorresistivos.* dirección: https://my.avnet.com/abacus/solutions/technologies/sensors/pressure-sensors/core-technologies/piezoresistive-strain-gauge/.

[45] A. International, *¿Cómo funcionan los sensores piezoeléctricos?* 2024. dirección: https://www.americanpiezo.com/blog/how-piezoelectric-sensors-work/.

[46] Wenglor, *Tecnología de los sensores de fibra óptica.* dirección: https://www.wenglor.com/es/Tecnologia-de-los-sensores-de-fibra-optica/s/Technologie+der+Faseroptischen+Sensoren?FaseroptischeSensorenBranchen=%3A1.

[47] SPI, *Comprensión de los sensores de presión matriciales.* dirección: https://www.sensorprod.com/glossary/matrix-pressure-sensor/.

[48] N. mechatronics, *Sensor de proximidad fotoeléctrico Infrarrojo E18-D80NK.* dirección: https://naylampmechatronics.com/sensores-proximidad/236-sensor-de-proximidad-fotoelectrico-infrarrojo-e18-d80nk-npn-no.html#:~:text=Su%20funcionamiento%20es%20el%20siguiente,%C3%BAnicamente%20muestra%20estados%20on%2Foff..

Anexo A

Cronograma

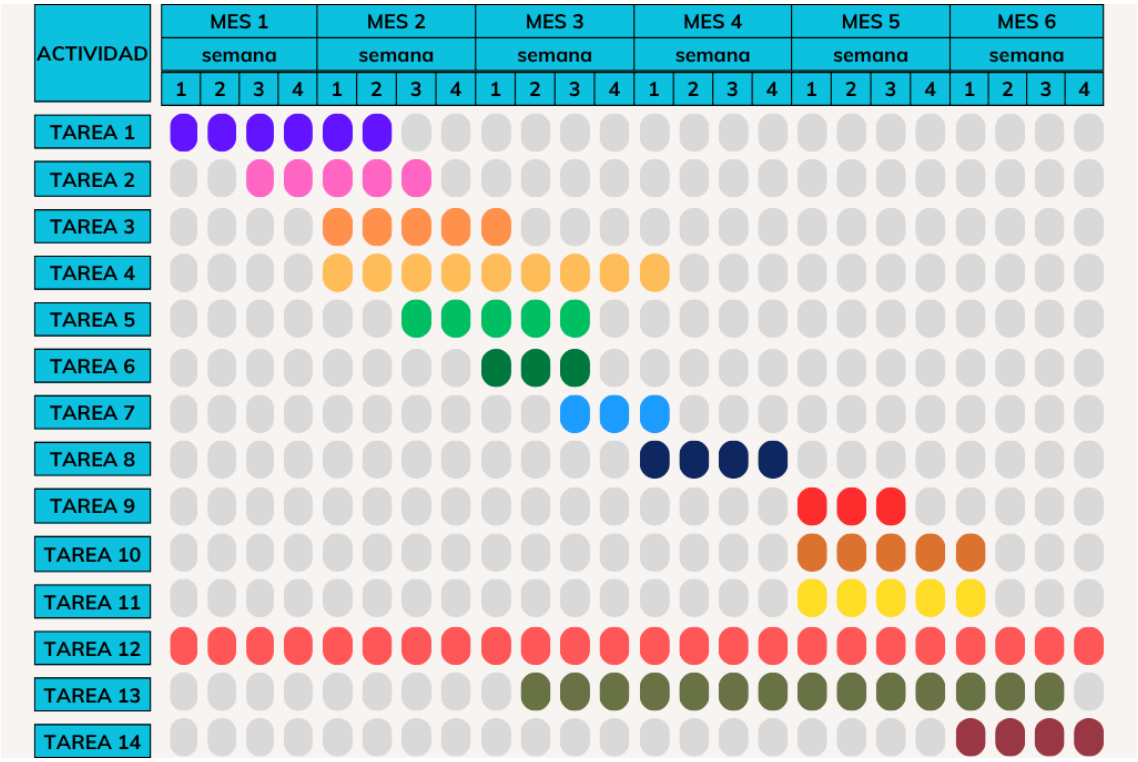


Figura A.1: Cronograma

Anexo B

Diagrama de flujo para la selección de dispositivos

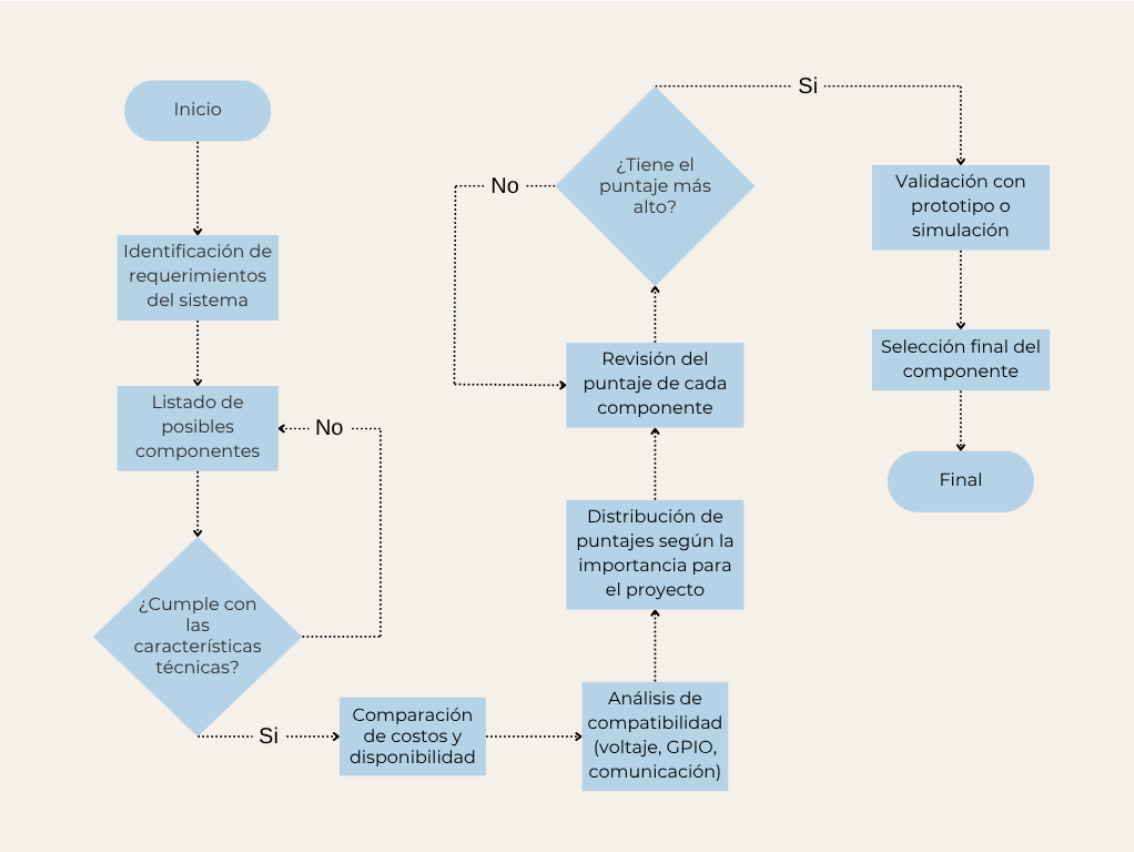


Figura B.1: Diagrama de flujo utilizado para la selección de dispositivos

Anexo C

Comparativa de los Sensores para el Cronómetro Digital

		6			7			5		
#	Sensor	Tiempo de Respuesta			Precio (COP)			Alcance		
1	E18-D80NK	2 ms	0,996	**5,976**	$ 160.000	0,55555556	**3,88888889**	3 - 80 cm	0,066667	**0,333333**
2	SHARP GP2Y0A21YK	38 ms	0,924	**5,544**	$ 140.000	0,61111111	**4,27777778**	10 - 80 cm	0,066667	**0,333333**
3	TF Mini Lidar	10 ms	0,98	**5,88**	$ 360.000	0	**0**	30 - 1200 cm	1	**5**
4	TCRT5000	2 ms	0,996	**5,976**	$ 64.736	0,82017778	**5,74124444**	0.2 - 2.5 cm	0,002083	**0,010417**
5	OSRAM SFH 9201	5 ms	0,99	**5,94**	$ 180.000	0,5	**3,5**	5 - 50 cm	0,041667	**0,208333**
6	KY-033	3 ms	0,994	**5,964**	$ 80.000	0,77777778	**5,44444444**	2 - 30 cm	0,025	**0,125**
7	GP2Y0A02YK	50 ms	0,9	**5,4**	$ 160.000	0,55555556	**3,88888889**	20 - 150 cm	0,125	**0,625**
8	IS471F	1 ms	0,998	**5,988**	$ 120.000	0,66666667	**4,66666667**	0.2 - 12 cm	0,01	**0,05**
9	PIR HC-SR501	500 ms	0	**0**	$ 28.000	0,92222222	**6,45555556**	300 - 700 cm	0,583333	**2,916667**
10	HC-SR04	50ms	0,9	**5,4**	$ 10.000	0,97222222	**6,80555556**	2 - 400cm	0,333333	**1,666667**

Figura C.1: Comparativa de sensores de cronómetro Parte 1

4			1			2			3			
Sensibilidad			**Clasificación IP (polvo)**			**Clasificación IP (agua)**			**Voltaje Operativo**			Total
Alta (1)	1	**4**	IP5	0,83333	**0,83333**	IP4	0,8	**1,6**	5V	0	**0**	16,6316
Media (0.6)	0,6	**2,4**	IP4	0,66667	**0,66667**	IP0	0	**0**	4.5V - 5.5V(5V)	0	**0**	13,2218
Alta (1)	1	**4**	IP6	1	**1**	IP5	1	**2**	4.5V - 6V (5V)	0	**0**	17,88
Alta (1)	1	**4**	IP4	0,66667	**0,66667**	IP0	0	**0**	5V	0	**0**	16,3943
Media (0.6)	0,6	**2,4**	IP4	0,66667	**0,66667**	IP0	0	**0**	3.3V - 5V (4V)	0,2	**0,6**	13,315
Media (0.6)	0,6	**2,4**	IP4	0,66667	**0,66667**	IP0	0	**0**	3.3V - 5V (4V)	0,2	**0,6**	15,2001
Media (0.6)	0,6	**2,4**	IP4	0,66667	**0,66667**	IP0	0	**0**	4.5V - 5.5V (5V)	0	**0**	12,9806
Alta (1)	1	**4**	IP4	0,66667	**0,66667**	IP0	0	**0**	2.7V - 5.5V(4,5V)	0,1	**0,3**	15,6713
Media (0.6)	0,6	**2,4**	IP4	0,66667	**0,66667**	IP4	0,8	**1,6**	3.3V - 5V (4V)	0,2	**0,6**	14,6389
Alta (1)	1	**4**	IP4	0,66667	**0,66667**	IP4	0,8	**1,6**	5V	0	**0**	20,1389

Figura C.2: Comparativa de sensores de cronómetro Parte 2

Anexo D

Comparativa de los Sensores para la detección de contactos

		7			6			4			3		
#	Sensor	Facilidad de adaptabilidad al proyecto			Sensibilidad			Tiempo de Respuesta (ms)			Voltaje (V)		
1	FlexiForce A201	Alta	1	7	Alta(0,8)	0,8	4,8	10	0,3333	1,3333	3.3–5	0,2	0,6
2	FSR402	Alta	1	7	Alta(0,8)	0,8	4,8	10	0,3333	1,3333	3.3–5	0,2	0,6
3	KY-036	Baja	0	0	Alta(0,8)	0,8	4,8	5	0,6667	2,6667	3.3–5	0,2	0,6
4	Piezoeléctrico	Alta	1	7	Alta(0,8)	0,8	4,8	2	0,8667	3,4667	3.3–5	0,2	0,6
5	D6T-44L	Media	0,5	3,5	Muy Alta(1)	1	6	15	0	0	5	0	0
6	Velostat	Alta	1	7	Muy alta (1)	1	6	5	0,6667	2,6667	3.3	1	3
7	MS60L	Baja	0	0	Media (0,4)	0,4	2,4	10	0,3333	1,3333	N/A	1	3
8	Switch Mecánico	Baja	0	0	Media (0,4)	0,4	2,4	5	0,6667	2,6667	N/A	1	3

Figura D.1: Comparativa de sensores de contacto Parte 1

5				1			2		
Precio (COP)	Cantidad necesaria (aprox)			IP Polvo			IP Agua		Total
$ 95.000	18	0,05	**0,25**	4	0,666667	**0,66667**	0	**0**	**14,65**
$ 60.000	30	0	**0**	4	0,666667	**0,66667**	0	**0**	**14,4**
$ 15.000	8	0,93333	**4,6667**	4	0,666667	**0,66667**	0	**0**	**13,4**
$ 6.300	40	0,86	**4,3**	4	0,666667	**0,66667**	0	**0**	**20,83333333**
$ 250.000	3	0,58333	**2,9167**	4	0,666667	**0,66667**	0	**0**	**13,08333333**
$ 35.700	7	0,86117	**4,3058**	4	0,666667	**0,66667**	0	**0**	**23,63916667**
$ 27.000	15	0,775	**3,875**	6	1	**1**	7	**1**	**12,60833333**
$ 5.000	20	0,94444	**4,7222**	6	1	**1**	7	**1**	**14,78888889**

Figura D.2: Comparativa de sensores de contacto Parte 2

Anexo E

Comparativa de los microcontroladores

		4			5			1		
#	Modelo	RAM			(Vout)			GPIO		
1	Arduino Nano	2 KB	3,8147E-06	1,5259E-05	5	1	5	14	0,21538462	0,21538462
2	ESP32	520 KB	0,00099182	0,00396729	5 - 3,3	1	5	34	0,52307692	0,52307692
3	ESP-01S	80 KB	0,00015259	0,00061035	3,3	0,66	3,3	2	0,03076923	0,03076923
4	Raspberry Pi 4	1GB	1	4	5 - 3,3	1	5	40	0,61538462	0,61538462
5	STM32F103C8T6	20 KB	3,8147E-05	0,00015259	5 - 3,3	1	5	37	0,56923077	0,56923077
6	Teensy 4.0	512 KB	0,00097656	0,00390625	5 - 3,3	1	5	40	0,61538462	0,61538462
7	Arduino Mega	8 KB	1,5259E-05	6,1035E-05	5	1	5	54	0,83076923	0,83076923
8	Raspberry Pi Pico	264 KB	0,00050354	0,00201416	5 - 3,3	1	5	26	0,4	0,4
9	NodeMCU ESP8266	80 KB	0,00015259	0,00061035	3,3	0,66	3,3	17	0,26153846	0,26153846
10	BeagleBone Black	512 MB	0,00097656	0,00390625	5 - 3,3	1	5	65	1	1

Figura E.1: Comparativa de microcontroladores Parte 1

7			6			3			2	
Interfaces de comun.			Precio			Vel. de reloj			Mem. Flash	
UART, SPI, I2C	0	0	$ 38.900	0,82334242	4,9400545	16MHz	0,010666667	0,032	32KB	10,1874704
WiFi, BT	1	7	$ 24.400	0,88919164	5,33514986	240MHz	0,16	0,48	4MB	18,3441941
WiFi	0,5	3,5	$ 8.000	0,96366939	5,78201635	80 - 160 MHz	0,106666667	0,32	1MB	12,9338959
WiFi, Ethernet	0,5	3,5	$ 220.200	0	0	1,5GHz	1	3	0	16,1153846
UART, SPI, I2C	0	0	$ 30.000	0,86376022	5,18256131	72MHz	0,048	0,144	64KB	10,8959767
USB, I2C, SPI	0	0	$ 120.000	0,45504087	2,73024523	600MHz	0,4	1,2	2MB	9,5505361
UART, SPI, I2C	0	0	$ 80.000	0,63669391	3,82016349	16MHZ	0,010666667	0,032	256KB	9,68312175
UART, SPI, I2C	0	0	$ 18.000	0,91825613	5,50953678	130MHz	0,086666667	0,26	2MB	11,1725509
WiFi	0,5	3,5	$ 15.000	0,93188011	5,59128065	80 - 160MHz	0,106666667	0,32	4MB	12,9754295
Ethernet, USB, UART	0	0	$ 190.000	0,13714805	0,82288828	1GHz	0,666666667	2	4GB	10,8267945

Figura E.2: Comparativa de microcontroladores Parte 2

Anexo F

Diagrama de Bloques

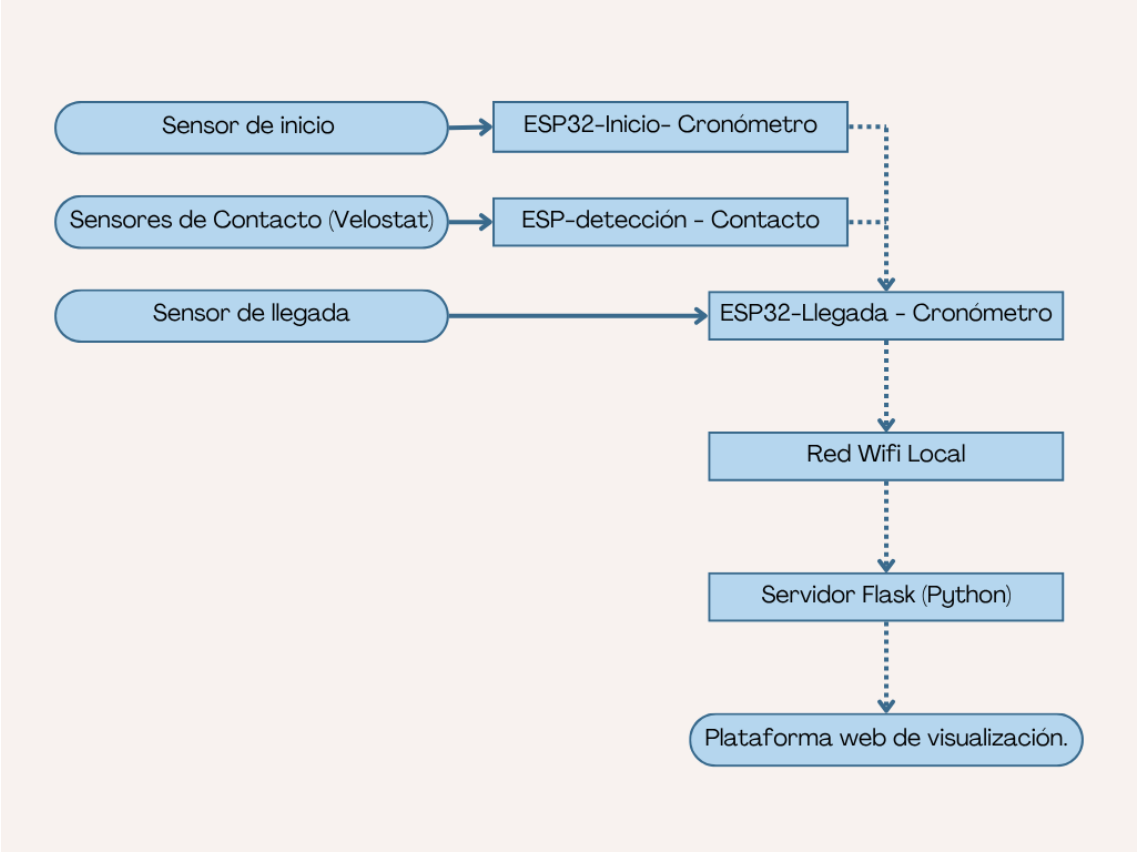


Figura F.1: Diagrama de bloques general

Anexo G

Diagrama de Flujo

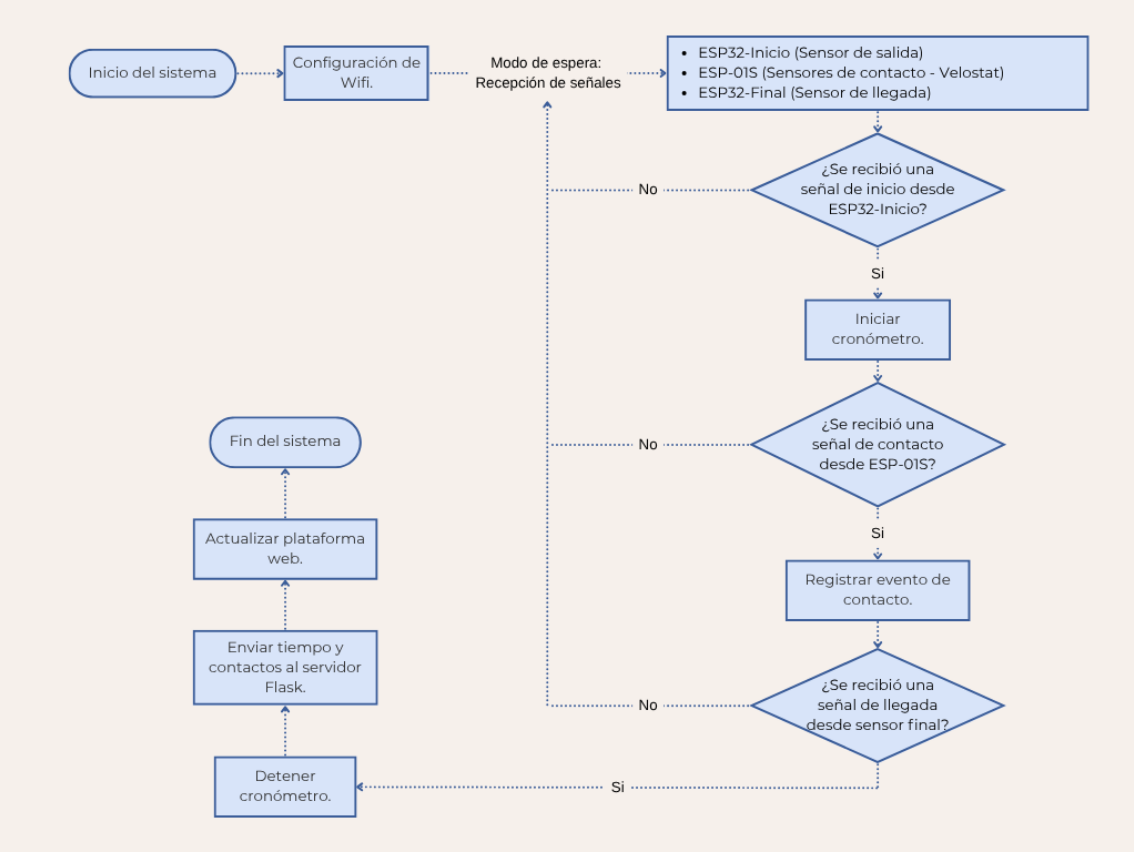


Figura G.1: Diagrama de flujo de operación

Anexo H

Código de Inicio

```
#include <esp_now.h>
#include <WiFi.h>

const char* ssid = "FAMILIA.MMP";
const char* password = "Maniva86";

// Dirección MAC del receptor (ESP32-Final)
uint8_t macFinal[] = {0x3C, 0x8A, 0x1F, 0xAB, 0xF3, 0x34};

#define TRIG_PIN 5
#define ECHO_PIN 18

// Umbral de detección (en cm)
#define UMBRAL 20

// Estructura del mensaje
typedef struct struct_message {
  char evento[10];  // "inicio"
} struct_message;

struct_message mensaje;

void conectarWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Conectando al WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("\nConectado al WiFi");
}

// Callback cuando se envía un mensaje
void OnSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("Mensaje enviado: ");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Éxito" : "Fallo");
}

// Medir distancia con HC-SR04
long medirDistancia(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duracion = pulseIn(echoPin, HIGH, 30000);  // Máximo 30ms = ~5 metros
  long distancia = duracion * 0.034 / 2;  // Convertir a cm
  return distancia;
}

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Configurar pines del sensor
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  conectarWiFi();

  // Cambiar modo WiFi a estación (solo para ESP-NOW)
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();  // Asegura que no esté conectado a internet
  delay(100);

  Serial.print("MAC de esta ESP32 (Inicio): ");
  Serial.println(WiFi.macAddress());
```

```
  // Inicializar ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error inicializando ESP-NOW");
    return;
  }

  esp_now_register_send_cb(OnSent);

  // Registrar peer
  esp_now_peer_info_t peerInfo = {};
  memcpy(peerInfo.peer_addr, macFinal, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  if (!esp_now_is_peer_exist(macFinal)) {
    if (esp_now_add_peer(&peerInfo) != ESP_OK) {
      Serial.println("Error añadiendo el peer");
      return;
    } else {
      Serial.println("Peer añadido correctamente");
    }
  }

  Serial.println("ESP-Inicio listo");
}

void loop() {
  long distancia = medirDistancia(TRIG_PIN, ECHO_PIN);
  Serial.print("Distancia: ");
  Serial.print(distancia);
  Serial.println(" cm");

  if (distancia > 0 && distancia < UMBRAL) {
    Serial.println("¡Ha pasado el perro!");

    strcpy(mensaje.evento, "inicio");

    esp_err_t result = esp_now_send(macFinal, (uint8_t *)&mensaje,
    sizeof(mensaje));
    if (result == ESP_OK) {
      Serial.println("Mensaje enviado correctamente");
```

```
    } else {
      Serial.print("Error al enviar mensaje: ");
      Serial.println(result);
    }

    delay(1000);  // Evitar múltiples disparos
  }

  delay(100);  // Espera breve para el próximo ciclo
}
```

Anexo I

Código de Rampa

```
#include <WiFi.h>
#include <esp_now.h>
#include <esp_wifi.h>  // Necesario para fijar canal

// --- CONFIGURACIÓN ---
const int pinVelostat = 34;
const float umbralVoltaje = 2.5;
bool enviado = false;

// Dirección MAC del receptor (ESP32-Final)
uint8_t macFinal[] = {0x3C, 0x8A, 0x1F, 0xAB, 0xF3, 0x34};

// Estructura del mensaje
typedef struct struct_message {
  char evento[10];  // "Rampa"
} struct_message;

struct_message mensaje;

// --- CALLBACK de envío ---
void OnSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("Mensaje enviado: ");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Éxito" : "Fallo");
}

void setup() {
  Serial.begin(115200);
  delay(1000);
```

```
  pinMode(pinVelostat, INPUT);

  // Configurar WiFi en modo estación y fijar canal manual
  WiFi.mode(WIFI_STA);
  esp_wifi_set_channel(8, WIFI_SECOND_CHAN_NONE);  // Fijar canal wifi

  Serial.print("MAC de esta ESP32 (rampa): ");
  Serial.println(WiFi.macAddress());

  // Inicializar ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error al iniciar ESP-NOW");
    return;
  }

  esp_now_register_send_cb(OnSent);

  // Registrar el peer receptor
  esp_now_peer_info_t peerFinal = {};
  memcpy(peerFinal.peer_addr, macFinal, 6);
  peerFinal.channel = 8; //canal wifi
  peerFinal.encrypt = false;

  if (!esp_now_is_peer_exist(macFinal)) {
    if (esp_now_add_peer(&peerFinal) != ESP_OK) {
      Serial.println("Error añadiendo peer Final");
      return;
    } else {
      Serial.println("Peer Final añadido correctamente");
    }
  }

  Serial.println("ESP-rampa listo");
}

void loop() {
  int lecturaADC = analogRead(pinVelostat);
  float voltaje = lecturaADC * 3.3 / 4095.0;

  Serial.print("Voltaje: ");
  Serial.print(voltaje);
```

```
  Serial.print(" V --> ");

  if (voltaje > umbralVoltaje && !enviado) {
    Serial.println("Perro detectado en Rampa");

    strcpy(mensaje.evento, "rampa");
    esp_err_t result = esp_now_send(macFinal, (uint8_t *)&mensaje,
    sizeof(mensaje));

    if (result == ESP_OK) {
      Serial.println("Mensaje enviado correctamente");
      enviado = true;
    } else {
      Serial.print("Error al enviar mensaje: ");
      Serial.println(result);
    }
  } else if (voltaje <= umbralVoltaje) {
    Serial.println("Sin detección.");
    enviado = false;
  }

  delay(200);
}
```

Anexo J

Código de Balancín

```
#include <WiFi.h>
#include <esp_now.h>
#include <esp_wifi.h>

// --- CONFIGURACIÓN ---
const int pinVelostat1 = 32;   // Sensor Balancín
const int pinVelostat2 = 35;   // Sensor Balancín Piso
const float umbralVoltaje = 2.5;

bool enviado1 = false;
bool enviado2 = false;

// Dirección MAC del receptor (ESP32-Final)
uint8_t macFinal[] = {0x3C, 0x8A, 0x1F, 0xAB, 0xF3, 0x34};

// Estructura del mensaje
typedef struct struct_message {
  char evento[20];  // Puede ser "balancin" o "balancin_piso"
} struct_message;

struct_message mensaje;

// --- CALLBACK de envío ---
void OnSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("Mensaje enviado: ");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Éxito" : "Fallo");
}

void setup() {
```

```
  Serial.begin(115200);
  delay(1000);

  pinMode(pinVelostat1, INPUT);
  pinMode(pinVelostat2, INPUT);

  // Configurar WiFi en modo estación y fijar canal manual
  WiFi.mode(WIFI_STA);
  esp_wifi_set_channel(8, WIFI_SECOND_CHAN_NONE);  // Fijar canal según wifi

  Serial.print("MAC de esta ESP32 (Balancín): ");
  Serial.println(WiFi.macAddress());

  // Inicializar ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error al iniciar ESP-NOW");
    return;
  }

  esp_now_register_send_cb(OnSent);

  // Registrar el peer receptor
  esp_now_peer_info_t peerFinal = {};
  memcpy(peerFinal.peer_addr, macFinal, 6);
  peerFinal.channel = 8; //← También canal de wifi
  peerFinal.encrypt = false;

  if (!esp_now_is_peer_exist(macFinal)) {
    if (esp_now_add_peer(&peerFinal) != ESP_OK) {
      Serial.println("Error añadiendo peer Final");
      return;
    } else {
      Serial.println("Peer Final añadido correctamente");
    }
  }

  Serial.println("ESP-Balancín listo con 2 sensores");
}

void loop() {
  // --- Sensor 1: Balancín ---
```

```
  int lecturaADC1 = analogRead(pinVelostat1);
  float voltaje1 = lecturaADC1 * 3.3 / 4095.0;

  Serial.print("Voltaje Balancín: ");
  Serial.print(voltaje1);
  Serial.print(" V --> ");

  if (voltaje1 > umbralVoltaje && !enviado1) {
    Serial.println("Perro detectado en balancín");

    strcpy(mensaje.evento, "balancin");
    esp_err_t result = esp_now_send(macFinal, (uint8_t *)&mensaje,
    sizeof(mensaje));

    if (result == ESP_OK) {
      Serial.println("Mensaje 'balancin' enviado correctamente");
      enviado1 = true;
    } else {
      Serial.print("Error al enviar mensaje balancín: ");
      Serial.println(result);
    }
  } else if (voltaje1 <= umbralVoltaje) {
    Serial.println("Balancín sin detección.");
    enviado1 = false;
  }

  // --- Sensor 2: Balancín Piso ---
  int lecturaADC2 = analogRead(pinVelostat2);
  float voltaje2 = lecturaADC2 * 3.3 / 4095.0;

  Serial.print("Voltaje Balancín Piso: ");
  Serial.print(voltaje2);
  Serial.print(" V --> ");

Serial.print("Lectura ADC2 cruda: ");
Serial.println(lecturaADC2);

  if (voltaje2 > umbralVoltaje && !enviado2) {
    Serial.println("Perro detectado en balancín piso");

    strcpy(mensaje.evento, "balancin_piso");
```

```
    esp_err_t result = esp_now_send(macFinal, (uint8_t *)&mensaje,
    sizeof(mensaje));

    if (result == ESP_OK) {
      Serial.println("Mensaje 'balancin_piso' enviado correctamente");
      enviado2 = true;
    } else {
      Serial.print("Error al enviar mensaje balancín piso: ");
      Serial.println(result);
    }
  } else if (voltaje2 <= umbralVoltaje) {
    Serial.println("Balancín piso sin detección.");
    enviado2 = false;
  }

  delay(200);
}
```

Anexo K

Código de Pasarela

```
#include <WiFi.h>
#include <esp_now.h>
#include <esp_wifi.h>

// --- CONFIGURACIÓN ---
const int pinVelostat = 34;
const float umbralVoltaje = 2.5;
bool enviado = false;

// Dirección MAC del receptor (ESP32-Final)
uint8_t macFinal[] = {0x3C, 0x8A, 0x1F, 0xAB, 0xF3, 0x34};

// Estructura del mensaje
typedef struct struct_message {
  char evento[10];  // "pasarela"
} struct_message;

struct_message mensaje;

// --- CALLBACK de envío ---
void OnSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("Mensaje enviado: ");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Éxito" : "Fallo");
}

void setup() {
  Serial.begin(115200);
  delay(1000);
```

```
  pinMode(pinVelostat, INPUT);

  // Configurar WiFi en modo estación y fijar canal manual
  WiFi.mode(WIFI_STA);
  esp_wifi_set_channel(8, WIFI_SECOND_CHAN_NONE);  // Fijar canal de wifi

  Serial.print("MAC de esta ESP32 (Pasarela): ");
  Serial.println(WiFi.macAddress());

  // Inicializar ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error al iniciar ESP-NOW");
    return;
  }

  esp_now_register_send_cb(OnSent);

  // Registrar el peer receptor
  esp_now_peer_info_t peerFinal = {};
  memcpy(peerFinal.peer_addr, macFinal, 6);
  peerFinal.channel = 8; // canal wifi
  peerFinal.encrypt = false;

  if (!esp_now_is_peer_exist(macFinal)) {
    if (esp_now_add_peer(&peerFinal) != ESP_OK) {
      Serial.println("Error añadiendo peer Final");
      return;
    } else {
      Serial.println("Peer Final añadido correctamente");
    }
  }

  Serial.println("ESP-pasarela listo");
}

void loop() {
  int lecturaADC = analogRead(pinVelostat);
  float voltaje = lecturaADC * 3.3 / 4095.0;

  Serial.print("Voltaje: ");
  Serial.print(voltaje);
```

```
  Serial.print(" V --> ");

  if (voltaje > umbralVoltaje && !enviado) {
    Serial.println("Perro detectado en pasarela");

    strcpy(mensaje.evento, "pasarela");
    esp_err_t result = esp_now_send(macFinal, (uint8_t *)&mensaje,
    sizeof(mensaje));

    if (result == ESP_OK) {
      Serial.println("Mensaje enviado correctamente");
      enviado = true;
    } else {
      Serial.print("Error al enviar mensaje: ");
      Serial.println(result);
    }
  } else if (voltaje <= umbralVoltaje) {
    Serial.println("Sin detección.");
    enviado = false;
  }

  delay(200);
}
```

Anexo L

Código Final

```
#include <WiFi.h>
#include <esp_now.h>
#include <HTTPClient.h>

// Pines HC-SR04
#define TRIG_PIN  32
#define ECHO_PIN  33

// LEDs
#define LED_VERDE 5
#define LED_ROJO  18

// Dirección MAC de la ESP32-Inicio y de sensores contacto
uint8_t macIniciar[] = {0xE8, 0x6B, 0xEA, 0xD0, 0x9F, 0xC8};
uint8_t macBalancin[] = {0x8C, 0xAA, 0xB5, 0x80, 0xF2, 0x28};
uint8_t macPasarela[]  = {0x80, 0xF3, 0xDA, 0x41, 0x27, 0x60};
uint8_t macRampa[]     = {0x08, 0xA6, 0xF7, 0x47, 0xD5, 0x0C};

// WiFi
const char* ssid = "FAMILIA.MMP";
const char* password = "Maniva86";

// URL del servidor Flask
const char* servidorTiempo   =
"http://nicoll1903.pythonanywhere.com/recibir_tiempo";
const char* servidorEvento =
"http://nicoll1903.pythonanywhere.com/evento_sensor";

// Estructura del mensaje recibido
```

```
typedef struct struct_message {
  char evento[20];
  // "inicio", "balancin", "pasarela", "rampa", "balancin_piso"
} struct_message;

struct_message mensajeRecibido;

// Variables del cronómetro
unsigned long tiempoInicio = 0;
unsigned long tiempoFinal = 0;
bool cronometroEnMarcha = false;
bool sensorActivado = false;
unsigned long tiempoDebounceLlegada = 0;
const unsigned long debounceDelay = 200;
const int DISTANCIA_UMBRAL = 20;

// Banderas para eventos
bool enviarEventoBalancin = false;
bool enviarEventoBalancinPiso = false;
bool enviarEventoPasarela = false;
bool enviarEventoRampa    = false;

void conectarWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Conectando a WiFi...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConectado al WiFi");
}

void agregarPeer(uint8_t *mac) {
  esp_now_peer_info_t peer = {};
  memcpy(peer.peer_addr, mac, 6);
  peer.channel = 0;
  peer.encrypt = false;

  if (!esp_now_is_peer_exist(mac)) {
    if (esp_now_add_peer(&peer) != ESP_OK) {
```

```
      Serial.println("Error al agregar peer");
    } else {
      Serial.print("Peer añadido: ");
      for (int i = 0; i < 6; i++) {
        Serial.printf("%02X", mac[i]);
        if (i < 5) Serial.print(":");
      }
      Serial.println();
    }
  }
}

void onDataRecv(const esp_now_recv_info_t *info, const uint8_t
*incomingData, int len) {
  memcpy(&mensajeRecibido, incomingData, sizeof(mensajeRecibido));
  Serial.print("Mensaje recibido: ");
  Serial.println(mensajeRecibido.evento);

  if (strcmp(mensajeRecibido.evento, "inicio") == 0) {
    if (!cronometroEnMarcha) {
      tiempoInicio = millis();
      cronometroEnMarcha = true;
      sensorActivado = false;
      Serial.println("Cronómetro iniciado");

      digitalWrite(LED_VERDE, HIGH);
      digitalWrite(LED_ROJO, LOW);
    }
  } else if (strcmp(mensajeRecibido.evento, "balancin") == 0) {
    Serial.println("Perro detectado en el balancín");
    enviarEventoBalancin = true;
  } else if (strcmp(mensajeRecibido.evento, "balancin_piso") == 0) {
    Serial.println("Perro detectado en el balancín piso");
    enviarEventoBalancinPiso = true;
  } else if (strcmp(mensajeRecibido.evento, "pasarela") == 0) {
    Serial.println("Perro detectado en la pasarela");
    enviarEventoPasarela = true;
  } else if (strcmp(mensajeRecibido.evento, "rampa") == 0) {
    Serial.println("Perro detectado en la rampa");
    enviarEventoRampa = true;
  }
```

```
}

void setup() {
  Serial.begin(115200);
  delay(1000);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_VERDE, OUTPUT);
  pinMode(LED_ROJO, OUTPUT);
  digitalWrite(LED_VERDE, LOW);
  digitalWrite(LED_ROJO, HIGH);

  conectarWiFi();

  Serial.print("MAC de esta ESP32 (Final): ");
  Serial.println(WiFi.macAddress());

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error al iniciar ESP-NOW");
    return;
  }

// Agregar peers
  agregarPeer(macIniciar);
  agregarPeer(macBalancin);
  agregarPeer(macPasarela);
  agregarPeer(macRampa);

  esp_now_register_recv_cb(onDataRecv);
  Serial.println("ESP-Final lista y esperando señal...");
}

long medirDistanciaCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duracion = pulseIn(ECHO_PIN, HIGH, 30000);
```

```
  long distancia = duracion * 0.034 / 2;
  return distancia;
}

void enviarEvento(const char* servidor, const char* nombre) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(servidor);
    http.addHeader("Content-Type", "application/json");

    String jsonPayload = "{\"evento\": \"" + String(nombre) + "\"}";
    int httpResponseCode = http.POST(jsonPayload);

    if (httpResponseCode > 0) {
      Serial.printf("Evento '%s' enviado. Código HTTP: %d\n", nombre,
      httpResponseCode);
    } else {
      Serial.printf("Error al enviar evento '%s'\n", nombre);
    }

    http.end();
  }
}

void loop() {
  long distancia = medirDistanciaCM();

  if (cronometroEnMarcha && distancia < DISTANCIA_UMBRAL && !sensorActivado
  && (millis() - tiempoDebounceLlegada > debounceDelay)) {
    tiempoFinal = millis();
    cronometroEnMarcha = false;
    sensorActivado = true;

    unsigned long duracion = tiempoFinal - tiempoInicio;
    unsigned int minutos = duracion / 60000;
    unsigned int segundos = (duracion % 60000) / 1000;
    unsigned int milisegundos = duracion % 1000;

    char tiempoFormateado[12];
    sprintf(tiempoFormateado, "%02u.%02u.%03u", minutos, segundos,
    milisegundos);
```

```
  Serial.print("Cronómetro detenido. Tiempo: ");
  Serial.println(tiempoFormateado);

  digitalWrite(LED_VERDE, LOW);
  digitalWrite(LED_ROJO, HIGH);

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(servidorTiempo);
    http.addHeader("Content-Type", "application/json");

    String jsonPayload = "{\"tiempo\": \"" + String(tiempoFormateado)
    + "\"}";
    int httpResponseCode = http.POST(jsonPayload);

    if (httpResponseCode > 0) {
      Serial.print("Tiempo enviado. Código HTTP: ");
      Serial.println(httpResponseCode);
    } else {
      Serial.print("Error al enviar tiempo: ");
      Serial.println(httpResponseCode);
    }

    http.end();
  }

  tiempoDebounceLlegada = millis();
}

// Enviar eventos pendientes
if (enviarEventoBalancin) {
  enviarEvento(servidorEvento, "balancin");
  enviarEventoBalancin = false;
}
if (enviarEventoBalancinPiso) {
  enviarEvento(servidorEvento, "balancin_piso");
  enviarEventoBalancinPiso = false;
}
if (enviarEventoPasarela) {
  enviarEvento(servidorEvento, "pasarela");
```

```
    enviarEventoPasarela = false;
  }
  if (enviarEventoRampa) {
    enviarEvento(servidorEvento, "rampa");
    enviarEventoRampa = false;
  }

  // Reactivar sensor si ya no hay objeto cerca
  if (distancia > DISTANCIA_UMBRAL + 5) {
    sensorActivado = false;
  }

  delay(100);
}
```

Anexo M

Código de la página web

M.1. App.py

```
from flask import Flask, render_template, request, redirect, url_for, session,
flash, jsonify
import sqlite3
import os
import json
from werkzeug.security import generate_password_hash, check_password_hash
from datetime import datetime

app = Flask(__name__)
app.secret_key = 'tu_clave_secreta'

tiempo_ultimo = "00.00.000"

# Conexión a la base de datos SQLite
def obtener_conexion():
    conn = sqlite3.connect('database.db')
    conn.row_factory = sqlite3.Row
    return conn

# --- LOGIN ---
@app.route('/', methods=['GET', 'POST'])
def login():
    session.clear()
    if request.method == 'POST':
        usuario = request.form['usuario'].strip()
        contrasena = request.form['contrasena'].strip()
```

```
        conn = obtener_conexion()
        cursor = conn.cursor()
        cursor.execute("SELECT usuario, contrasena, tipo FROM usuarios WHERE
        usuario=?", (usuario,))
        user = cursor.fetchone()
        conn.close()

        if user:
            user_usuario = user[0]
            user_contrasena = user[1]
            user_tipo = user[2].lower()

            if check_password_hash(user_contrasena, contrasena):
                session['usuario'] = user_usuario
                session['tipo'] = user_tipo

                if user_tipo == 'administrativo':
                    return redirect(url_for('admin'))
                elif user_tipo == 'propietario':
                    return redirect(url_for('propietario'))
                else:
                    return redirect(url_for('index'))

        flash("Usuario o contraseña incorrectos", "error")

    return render_template('login.html')

# --- REGISTRO ---
@app.route('/registro', methods=['GET', 'POST'])
def registro():
    if request.method == 'POST':
        usuario = request.form['usuario'].strip()
        contrasena = request.form['contrasena'].strip()
        tipo = request.form['tipo'].strip().lower()
        clave_especial = request.form.get('clave_especial', '').strip()

        if not usuario:
            flash("El nombre de usuario no puede estar vacío", "error")
            return redirect(url_for('registro'))

        conn = obtener_conexion()
```

```
        cursor = conn.cursor()

        if tipo == "administrativo" and clave_especial != "12345":
            flash("Clave especial incorrecta para Administrador", "error")
            conn.close()
            return redirect(url_for('registro'))

        if tipo == "propietario":
            cursor.execute("SELECT id FROM claves_propietarios WHERE clave=?
            AND usada=0", (clave_especial,))
            clave_valida = cursor.fetchone()
            if not clave_valida:
                flash("Clave especial incorrecta o ya utilizada", "error")
                conn.close()
                return redirect(url_for('registro'))

        contrasena_hash = generate_password_hash(contrasena)

        try:
            cursor.execute("INSERT INTO usuarios (usuario, contrasena, tipo) VALUES
            (?, ?, ?)",
                           (usuario, contrasena_hash, tipo))
            conn.commit()

            if tipo == "propietario":
                cursor.execute("UPDATE claves_propietarios SET usada = 1 WHERE clave
                = ?", (clave_especial,))
                conn.commit()

            flash("Cuenta creada con éxito, inicia sesión", "success")
        except sqlite3.IntegrityError:
            flash("El usuario ya existe", "error")

        conn.close()
        return redirect(url_for('login'))

    return render_template('registro.html')

# --- PÁGINAS ---
@app.route('/index')
def index():
```

```
    if not session.get('usuario'):
        return redirect(url_for('login'))

    if session.get('tipo') == 'administrativo':
        return redirect(url_for('admin'))

    return render_template('index.html')

@app.route('/admin')
def admin():
    if not session.get('usuario') or session.get('tipo') != 'administrativo':
        flash("Acceso denegado: Solo administradores pueden acceder", "error")
        return redirect(url_for('index'))

    conn = obtener_conexion()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM participantes")
    participantes = cursor.fetchall()
    conn.close()

    return render_template('admin.html', participantes=participantes)

@app.route('/registrar_participante', methods=['POST'])
def registrar_participante():
    if not session.get('usuario') or session.get('tipo') != 'administrativo':
        flash("Acceso denegado: Solo administradores pueden realizar esta acción",
        "error")
        return redirect(url_for('index'))

    nombre = request.form['nombre']
    fecha = request.form['fecha']
    lugar = request.form['lugar']

    conn = obtener_conexion()
    cursor = conn.cursor()
    cursor.execute("INSERT INTO participantes (nombre, fecha, lugar) VALUES (?, ?, ?)",
                   (nombre, fecha, lugar))
    conn.commit()
    conn.close()

    flash("Participante registrado correctamente", "success")
```

```
    return redirect(url_for('admin'))

@app.route('/actualizar_participante', methods=['POST'])
def actualizar_participante():
    if not session.get('usuario') or session.get('tipo') != 'administrativo':
        flash("Acceso denegado: Solo administradores pueden realizar esta acción",
        "error")
        return redirect(url_for('index'))

    participante_id = request.form['id']
    nombre = request.form['nombre']
    fecha = request.form['fecha']
    lugar = request.form['lugar']

    conn = obtener_conexion()
    cursor = conn.cursor()
    cursor.execute("UPDATE participantes SET nombre=?, fecha=?, lugar=? WHERE id=?",
                   (nombre, fecha, lugar, participante_id))
    conn.commit()
    conn.close()

    flash("Participante actualizado correctamente", "success")
    return redirect(url_for('admin'))

@app.route('/seleccionar_participante', methods=['POST'])
def seleccionar_participante():
    if not session.get('usuario') or session.get('tipo') != 'administrativo':
        return jsonify({"error": "Acceso denegado"}), 403

    data = request.get_json()
    participante_id = data.get('id')

    if not participante_id:
        return jsonify({"error": "ID de participante no proporcionado"}), 400

    conn = obtener_conexion()
    cursor = conn.cursor()

    cursor.execute("DELETE FROM tiempo_real")
    cursor.execute("INSERT INTO tiempo_real (participante_id) VALUES (?)",
    (participante_id,))
```

```
    conn.commit()
    conn.close()

    session['participante_id'] = participante_id
    return jsonify({"status": "ok"})

# --- EVENTOS DESDE ESP32 (UNIFICADO) ---
@app.route('/evento_sensor', methods=['POST'])
def evento_sensor():
    try:
        data = request.get_json()
        print(f"Mensaje recibido desde ESP32: {data}")

        evento = data.get("evento", "").lower()

        columnas = {
            "balancin": "contactoBalancin",
            "balancin_piso": "contactoBalancinPiso",
            "pasarela": "contactoPasarela",
            "rampa": "contactoRampa"
        }

        if evento not in columnas:
            return jsonify({'status': 'error', 'mensaje': 'Evento no válido'}), 400

        columna = columnas[evento]

        conn = obtener_conexion()
        cursor = conn.cursor()
        cursor.execute(f"""
            UPDATE tiempo_real
            SET {columna} = 'Detectado'
            WHERE ROWID = (SELECT ROWID FROM tiempo_real ORDER BY ROWID DESC LIMIT 1)
        """)
        conn.commit()
        conn.close()

        print(f"Evento '{evento}' registrado en {columna}")
        return jsonify({'status': 'ok', 'mensaje': f'{evento} detectado'}), 200
```

```
    except Exception as e:
        print(f"Error procesando evento: {e}")
        return jsonify({'error': str(e)}), 500

# --- GUARDAR PARTICIPANTE ACTUAL ---
@app.route('/guardar_participante_actual', methods=['POST'])
def guardar_participante_actual():
    if not session.get('usuario') or session.get('tipo') != 'administrativo':
        return jsonify({"error": "Acceso no autorizado"}), 403

    conn = obtener_conexion()
    cursor = conn.cursor()

    # Traemos cronómetro y contactos desde tiempo_real
    cursor.execute("""
        SELECT participante_id, cronometro, contactoPasarela, contactoBalancin,
        contactoBalancinPiso, contactoRampa
        FROM tiempo_real
        ORDER BY ROWID DESC LIMIT 1
    """)
    resultado = cursor.fetchone()

    if resultado is not None:
        participante_id = resultado['participante_id']
        tiempo = resultado['cronometro'] if resultado['cronometro'] else '00.00.000'

        contacto_pasarela = resultado['contactoPasarela'] if resultado
        ['contactoPasarela'] else 'No detectado'
        contacto_balancin = resultado['contactoBalancin'] if resultado
        ['contactoBalancin'] else 'No detectado'
        contacto_balancin_piso = resultado['contactoBalancinPiso'] if resultado
        ['contactoBalancinPiso'] else 'No detectado'
        contacto_rampa = resultado['contactoRampa'] if resultado['contactoRampa']
        else 'No detectado'

        cursor.execute("SELECT * FROM participantes WHERE id=?", (participante_id,))
        participante = cursor.fetchone()

        if participante:
            fecha_actual = participante['fecha']
```

```
            cursor.execute('''
                INSERT INTO registros (fecha, nombre, lugar, cronometro,
                contactoPasarela, contactoBalancin, contactoBalancinPiso, contactoRampa)
                VALUES (?, ?, ?, ?, ?, ?, ?, ?)
            ''', (
                fecha_actual,
                participante['nombre'],
                participante['lugar'],
                tiempo,
                contacto_pasarela,
                contacto_balancin,
                contacto_balancin_piso,
                contacto_rampa
            ))
            conn.commit()
            conn.close()
            flash("Participante guardado correctamente", "success")
            return jsonify({'success': True})

    conn.close()
    return jsonify({'error': 'No hay participante activo'}), 404

@app.route('/registros_fechas')
def registros_fechas():
    conn = obtener_conexion()
    cursor = conn.cursor()

    cursor.execute("SELECT DISTINCT fecha FROM registros ORDER BY fecha DESC")
    fechas = [fila['fecha'] for fila in cursor.fetchall()]

    conn.close()
    return jsonify(fechas=fechas)

@app.route('/registros_fechas/<fecha>')
def registros_fecha_individual(fecha):
    conn = obtener_conexion()
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM registros WHERE fecha = ?", (fecha,))
    participantes = [dict(row) for row in cursor.fetchall()]
```

```
    conn.close()
    return jsonify({'participantes': participantes})

@app.route('/participante_actual', methods=['GET'])
def participante_actual():
    conn = obtener_conexion()
    cursor = conn.cursor()

    # Traemos cronómetro y contactos desde tiempo_real
    cursor.execute("""
        SELECT participante_id, cronometro, contactoPasarela, contactoBalancin,
        contactoBalancinPiso, contactoRampa
        FROM tiempo_real
        ORDER BY ROWID DESC LIMIT 1
    """)
    resultado = cursor.fetchone()

    if resultado is None:
        conn.close()
        return jsonify({})  # No hay datos, devolvemos JSON vacío

    participante_id = resultado['participante_id']
    cronometro = resultado['cronometro'] if resultado['cronometro'] else '00.00.000'
    contacto_pasarela = resultado['contactoPasarela'] if resultado['contactoPasarela']
    else 'No detectado'
    contacto_balancin = resultado['contactoBalancin'] if resultado['contactoBalancin']
    else 'No detectado'
    contacto_balancin_piso = resultado['contactoBalancinPiso'] if resultado
    ['contactoBalancinPiso'] else 'No detectado'
    contacto_rampa = resultado['contactoRampa'] if resultado['contactoRampa']
    else 'No detectado'

    cursor.execute("SELECT * FROM participantes WHERE id = ?", (participante_id,))
    participante = cursor.fetchone()
    conn.close()

    if participante:
        return jsonify({
            'nombre': participante['nombre'],
            'fecha': participante['fecha'],
            'lugar': participante['lugar'],
```

```
            'cronometro': cronometro,
            'contactoPasarela': contacto_pasarela,
            'contactoBalancin': contacto_balancin,
            'contactoBalancinPiso': contacto_balancin_piso,
            'contactoRampa': contacto_rampa
        })

    return jsonify({})

@app.route('/vista_observador')
def vista_observador():
    return render_template('index.html')  # o el archivo HTML correspondiente

@app.route('/propietario')
def propietario():
    if not session.get('usuario') or session.get('tipo') != 'propietario':
        flash("Acceso denegado: Solo Propietarios pueden acceder", "error")
        return redirect(url_for('index'))

    return render_template('propietario.html')

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('login'))

@app.route('/recibir_tiempo', methods=['POST'])
def recibir_tiempo():
    global tiempo_ultimo
    data = request.get_json()

    tiempo_ultimo = data.get('tiempo', '00.00.000')
    print(f"Tiempo recibido desde ESP32: {tiempo_ultimo}")

    try:
        conn = obtener_conexion()
        cursor = conn.cursor()

        # Actualizar SOLO la última fila en tiempo_real
        cursor.execute("""
            UPDATE tiempo_real
```

```
            SET cronometro=?
            WHERE ROWID = (SELECT ROWID FROM tiempo_real ORDER BY ROWID DESC LIMIT 1)
        """, (tiempo_ultimo,))

        if cursor.rowcount == 0:
            print("No se actualizó ninguna fila. ¿Hay participante seleccionado?")
        else:
            print("Tiempo guardado correctamente en la última fila.")

        conn.commit()
        conn.close()

        return jsonify({'status': 'ok', 'tiempo': tiempo_ultimo}), 200

    except Exception as e:
        print(f"Error al guardar el tiempo v2: {e}")
        return jsonify({'error': f'Error al guardar el tiempo v2: {e}'}), 500

@app.route('/lista_participantes', methods=['GET'])
def lista_participantes():
    try:
        conn = obtener_conexion()
        cursor = conn.cursor()
        cursor.execute('SELECT id, nombre FROM participantes')
        participantes = cursor.fetchall()
        conn.close()

        participantes_list = [{'id': p[0], 'nombre': p[1]} for p in participantes]

        return jsonify(participantes_list), 200

    except Exception as e:
        print(f"Error al obtener la lista de participantes: {e}")
        return jsonify({'error': 'No se pudieron obtener los participantes'}), 500

@app.route('/obtener_tiempo', methods=['GET'])
def obtener_tiempo():
    global tiempo_ultimo
    if tiempo_ultimo:
        return jsonify({'tiempo': tiempo_ultimo}), 200
    else:
```

```
        return jsonify({'tiempo': '00.00.000'}), 200

@app.route('/tiempo_actual', methods=['GET'])
def tiempo_actual():
    return jsonify({'tiempo': tiempo_ultimo})

@app.route('/consultar_cronometro', methods=['GET'])
def consultar_cronometro():
    conn = obtener_conexion()
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM tiempo_real")
    tiempo_real = [dict(row) for row in cursor.fetchall()]

    conn.close()
    return jsonify({'tiempo_real': tiempo_real})


@app.route('/guardar_actual', methods=['POST'])
def guardar_actual():
    try:
        conn = obtener_conexion()
        cursor = conn.cursor()

        # Traemos también los contactos desde tiempo_real
        cursor.execute("""
            SELECT
                t.participante_id,
                t.cronometro,
                t.contactoPasarela,
                t.contactoBalancin,
                t.contactoBalancinPiso,
                t.contactoRampa,
                p.fecha,
                p.nombre,
                p.lugar
            FROM tiempo_real t
            INNER JOIN participantes p ON t.participante_id = p.id
            ORDER BY t.ROWID DESC LIMIT 1
        """)
        participante = cursor.fetchone()
```

```
if participante:

    contacto_pasarela = participante['contactoPasarela'] if participante
    ['contactoPasarela']
    else 'No detectado'
    contacto_balancin = participante['contactoBalancin'] if participante
    ['contactoBalancin'] else 'No detectado'
    contacto_balancin_piso = participante['contactoBalancinPiso'] if
    participante['contactoBalancinPiso'] else 'No detectado'
    contacto_rampa = participante['contactoRampa'] if participante
    ['contactoRampa'] else 'No detectado'

    cursor.execute("""
        INSERT INTO registros (fecha, nombre, lugar, cronometro,
        contactoPasarela, contactoBalancin, contactoBalancinPiso, contactoRampa)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)
    """, (
        participante['fecha'],
        participante['nombre'],
        participante['lugar'],
        participante['cronometro'],
        contacto_pasarela,
        contacto_balancin,
        contacto_balancin_piso,
        contacto_rampa
    ))

    conn.commit()
    conn.close()

    participante_dict = {
        'nombre': participante['nombre'],
        'fecha': participante['fecha'],
        'lugar': participante['lugar'],
        'cronometro': participante['cronometro'],
        'contactoPasarela': contacto_pasarela,
        'contactoBalancin': contacto_balancin,
        'contactoBalancinPiso': contacto_balancin_piso,
        'contactoRampa': contacto_rampa
    }
```

```
            return jsonify({'participante actualizado': participante_dict})

    except Exception as e:
        print(f"Error al guardar el participante actual en la base de datos: {e}")
        return jsonify({'error': f'Error al guardar el tiempo {e}'}), 500

    return jsonify({})
```