Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Electrónica

Maestría en Ingeniería Electrónica,

opción Instrumentación Electrónica

Tesis para obtener el grado de:

Maestro en Ingeniería Electrónica

Sistema de localización de objetos en tiempo real, utilizando sensores de profundidad y modelos de aprendizaje automático

---

Presenta:

Ing. Javier Martínez Fernández*

Asesores:

M.C. Nicolás Quiroz Hernández

M.C. Ana María Rodríguez Domínguez

Dr. Roberto Olmos Pimentel

*Becario SECIHTI

Puebla, Pue., enero 2026

Agradecimientos

En primer lugar, agradezco al Consejo Nacional de Humanidades, Ciencias y Tecnología (CONAHCYT) por el apoyo económico brindado a lo largo del programa de maestría.

A la Benemérita Universidad Autónoma de Puebla (BUAP), institución en la que realice mis estudios, y a la Facultad de Ciencias de la Electrónica, junto con todos sus profesores, por contribuir de manera significativa a mi formación académica.

Expreso mi más sincero reconocimiento a mis asesores de tesis: el M.C. Nicolás Quiroz Hernández, la M.C. Ana María Rodríguez Domínguez y Dr. Roberto Olmos Pimentel, por su apoyo y paciencia en los momentos complicados, así como la orientación técnica y la confianza recibida durante el desarrollo de este trabajo. También se reconoce el enorme esfuerzo de la M.C. Ana María que gestiono la documentación necesaria para el ingreso a la maestría, muchas gracias.

De igual forma, agradezco a mis sinodales, M.C. Rodrigo Lucio Maya Ramírez, Dr. Aldrin Barreto Flores y M.C. Selene Edith Maya Rueda por sus observaciones que sirvieron para mejorar este trabajo de tesis.

Agradezco también a la universidad KU Leuven y en especial al Dr. Roberto de Lima Hernández, por ayudarme con su valioso conocimiento y experiencia en el ámbito de la calibración de cámaras RGB y sensores LiDAR para mapeo espacial del entorno.

A mis padres y hermano, gracias a su apoyo incondicional, cariño y palabras de aliento que me impulsaron a seguir adelante. Siempre han sido mi motivación para salir adelante, sin ustedes no sería posible llegar hasta este punto.

A mi novia, por su confianza, compañía y respaldo en cada etapa de este camino.

Y finalmente, a mis amigos y compañeros, por su apoyo y motivación constante a lo largo del desarrollo de este trabajo de tesis.

Dedicatoria

Dedico esta tesis a mis padres, Angel y Elvira, así como a mi hermano Miguel, por su constante apoyo para culminar mis estudios, que, a pesar de las dificultades causadas por motivos de salud, hemos podido sobreponernos a las adversidades manteniendo un núcleo familiar unido.

A mi tía Piedad que siempre me animo a seguir adelante, sin embargo, la muerte le aconteció antes de tiempo, siempre la recordaré con mucho cariño.

A toda mi familia por su apoyo que me brindaron para llegar hasta donde estoy hoy en día.

Así mismo, dedico esta tesis a mis asesores, que durante todo el tiempo que estuve en la maestría me acompañaron y apoyaron brindándome su conocimiento y experiencia. A pesar de sus agendas apretadas, encontraban la manera de darme un poco de su tiempo para corregir y mejorar el trabajo de tesis.

A mi novia, que desde el inicio de la maestría me acompaño y me brindo su apoyo para sobreponerme en los momentos complicados.

Y por último, pero no menos importante, a todos mis compañeros de la maestría, especialmente a Abraham y Alberto, gracias a ellos pase momentos muy amenos, incluso en situaciones donde las actividades de las clases se complicaban, mis mejores deseos para ellos.

Índice general

Índice de figuras

Índice de tablas

Glosario

Algoritmo	Procedimiento utilizado para resolver un problema o realizar un cálculo.
Clustering	Es una técnica de aprendizaje no supervisado que agrupa datos en conjuntos basándose en similitudes.
CNH	Histograma Normalizado Compacto (*Compact Normalized Histogram*).
Data lakes	Repositorio de almacenamiento que permite guardar grandes volúmenes de datos en su forma original, sin necesidad de estructurarlos previamente.
DOE	Elemento Óptico Difractivo (*Diffractive Optical Element*).
Edge Computing	Computo en el borde.
FFT	Transformada Rápida de Fourier (FFT) es un algoritmo eficiente para calcular la Transformada Discreta de Fourier (DFT) y su inversa.
FoV	Campo de visión (*Field-of-View*).
Framework	Conjunto de herramientas, bibliotecas y reglas que proporcionan una estructura para desarrollar software de manera más eficiente.
Ground Truth	Es el conjunto de datos etiquetados que se consideran la verdad absoluta frente a la cual se comparan las predicciones de un modelo.
Hiperparámetros	Configuraciones que se establecen antes de entrenar un modelo de aprendizaje automático y afectan su rendimiento.
I2C	Circuito Inter-Integrado (*Inter-Integrated Circuit*).
Kcps/SPAD	Miles de cuentas por segundo por SPAD (Kilo-count per second per SPAD). Unidad utilizada para cuantificar el número de fotones que entran en la matriz SPAD.
MCU	Microcontrolador (*Microcontroller Unit*).
MPI	Interferencia de multipath.
OpenCV	(*Open Source Computer Vision*) es una biblioteca de código abierto especializada en visión artificial y aprendizaje automático.
Pipeline	Secuencia de pasos automatizados que procesan y transforman datos desde su origen hasta su destino final.

Procesos ETL	(*Extract, Transform, Load*) permiten extraer información de múltiples fuentes, transformarla para que sea útil y cargarla en un sistema de almacenamiento o análisis
RAM	Memoria de Acceso Aleatorio (*Random Access Memory*).
Raw data	Datos en bruto, se refiere a la información original que no ha sido procesada, limpiada ni transformada.
SCL	Línea de reloj en serie (*Serial Clock Line*)
SDA	Datos en serie (*Serial Data*).
Sensor ToF monozona	Sensores de tiempo de vuelo que miden la distancia en un solo punto.
Sensor ToF multizona	Sensores de tiempo de vuelo que miden la distancia de múltiples zonas a la vez, dentro de un campo de visión determinado.
SLAM	Localización y mapeo simultáneos (*Simultaneous Localization and Mapping*).
SoC	Sistema en un Chip (*System on a Chip*).
SPAD	Diodo de Avalancha de Fotón Único (*Single Photon Avalanche Diode*).
TinyML	Aprendizaje Automático Integrado.
ToF	Tiempo de Vuelo (*Time-of-Flight*).
Transfer Learning	Técnica de aprendizaje automático que permite reutilizar conocimientos adquiridos en un modelo previamente entrenado para resolver una nueva tarea relacionada.
UAV	Vehículo aéreo no tripulado (*Unmanned Aerial Vehicle*).
ULD	Controlador Ultraligero (*Ultra Light Driver*).
VCSEL	Diodo emisor superficial de cavidad vertical (*Vertical Cavity Surface-Emitting Diode*).
Wearables	Vestibles.
Xtalk	Diafonía (*Crosstalk*)

Resumen

La discapacidad visual representa un problema y a su vez, un desafío para la movilidad autónoma y segura de las personas. Las soluciones tecnológicas actuales suelen depender de hardware costoso, voluminoso o de conectividad a la nube, limitando su accesibilidad y tiempo de respuesta. El presente trabajo de investigación propone el diseño e implementación de un dispositivo de asistencia portable (*wearable*), fundamentado en técnicas de Aprendizaje Automático Integrado (TinyML) y tecnología de sensores de Tiempo de Vuelo (ToF). El sistema integra un módulo de percepción de profundidad MaixSense A010 y una tarjeta de desarrollo Nicla Vision. Se desarrolló una arquitectura de software asimétrica sobre el microcontrolador de doble núcleo STM32H747: el núcleo Cortex-M4 gestiona un bucle de seguridad determinista para la evasión de obstáculos inmediatos mediante retroalimentación vibratoria, mientras que el núcleo Cortex-M7 ejecuta una Red Neuronal Convolucional (CNN) optimizada para la clasificación semántica del entorno (escaleras y personas). Para la implementación del modelo de inteligencia artificial, se construyó un dataset propio de mapas de profundidad, aplicando técnicas de cuantización a enteros de 8 bits (Int8) y compilación en C/C++ (EON Compiler). Los resultados experimentales demostraron un tiempo de inferencia de 16 ms y un F1-Score superior al 96% en la detección de clases objetivo, superando en eficiencia a implementaciones estándar basadas en intérpretes (TensorFlow Lite). Las pruebas en entorno controlado validaron la capacidad del sistema para operar en tiempo real bajo condiciones de iluminación variables, confirmando que la integración de visión artificial embebida de bajo consumo es una solución viable y privada para la asistencia de navegación.

Capítulo 1

1. Generalidades y estado del arte

En este capítulo, se presentan los fundamentos conceptuales y los avances preliminares que sustentan el tema de tesis.

En primer lugar, se establecen los objetivos, tanto el general como los específicos, que delimitan el alcance del proyecto y señalan la dirección que tomara el trabajo, permitiendo diseñar un cronograma de actividades detallado y enfocado a cumplir con estos objetivos.

En las siguientes secciones, se justifica la pertinencia de la investigación a partir de un análisis crítico del estado del arte, en el cual, se consideran iniciativas académicas y proyectos de I+D previos, así como soluciones comerciales actualmente disponibles. Este análisis permite identificar oportunidades de mejora y posibles aportaciones que sustenten la propuesta aquí desarrollada.

Al final del capítulo 1, se incluye una descripción global del sistema planteado y se detalla la estructura del documento, permitiendo establecer una ruta para los capítulos posteriores.

1.1. Introducción

En los últimos años, las técnicas de visión por computadora han evolucionado con rapidez. Los métodos tradicionales, basados en algoritmos programados manualmente para extraer características como: bordes, texturas y formas, han sido desplazados por el aprendizaje automático (ML) y, más recientemente, por las redes neuronales profundas [1]. Hoy, los modelos aprenden representaciones directamente de los datos, sin necesidad de diseñar o extraer a mano las características. Ejemplo de ello son las redes neuronales convolucionales (CNN), que han revolucionado áreas de investigación como el reconocimiento de imágenes y la detección de objetos, haciendo posible identificar automáticamente estructuras complejas a partir de grandes volúmenes de datos.

Estos avances han mejorado la precisión y la eficiencia de los sistemas de visión, ampliando sus aplicaciones en ámbitos como la conducción autónoma, la medicina, la realidad aumentada, entre muchas otras. Al mismo tiempo, han surgido nuevos paradigmas para el desarrollo y despliegue de modelos: uno de los más recientes es el aprendizaje automático integrado (TinyML) [1-3]. Esta nueva forma de aplicar modelos de aprendizaje automático permite ejecutar inferencias directamente en microcontroladores de recursos muy limitados, lo que reduce la dependencia de la nube, que, a su vez, disminuye la latencia y prolonga la autonomía de dispositivos alimentados por baterías.

No obstante, al abordar la visión por computadora mediante modelos de aprendizaje automático un requisito esencial para el entrenamiento de modelos confiables es disponer de datos de buena calidad. Lo que nos lleva a considerar el papel tan importante que tienen los dispositivos de adquisición de datos.

Los sensores actúan como el vínculo entre el entorno físico y la unidad de procesamiento (en este caso, un microcontrolador), proporcionando la información necesaria para la interpretación del contexto [4]. En este sentido, la tecnología de Tiempo de Vuelo (ToF) ofrece una solución interesante para la medición de distancias mediante dispositivos compactos, de bajo consumo energético y con tiempos de respuesta reducidos [5], [6]. La sinergia entre estos sensores y los modelos TinyML abre un amplio espectro de posibilidades en diversos sectores, tales como la

salud, la electrónica de consumo, la robótica industrial y los sistemas de seguridad (véase Figura 1.1).

Figura 1.1. Ejemplos de aplicaciones que integran sensores ToF y TinyML.

Tomando en cuenta las características anteriores. Este trabajo propone el desarrollo de un sistema portátil, compacto y de bajo consumo de energía, implementando un módulo ToF MaixSense A-010 y un modelo TinyML que harán posible detectar la posición y distancia de objetos en tiempo real.

Respondiendo a los Objetivos de Desarrollo Sostenible de la Organización de las Naciones Unidas, particularmente al ODS 3 (Salud y Bienestar) [7], este sistema está pensado como una primera parte, que posteriormente se integrará en un proyecto aún más grande, el cual busca tener un impacto social positivo, dirigido a personas con discapacidad visual.

1.2. Objetivos

1.2.1. Objetivo general

- Desarrollar un sistema de localización de objetos en tiempo real, empleando sensores de profundidad y modelos de aprendizaje automático, con el propósito de facilitar la inclusión de personas con limitaciones visuales.

1.2.2. Objetivos específicos

1. Analizar y seleccionar el hardware adecuado para el sistema de localización, considerando los siguientes requerimientos: bajo consumo de energía, portabilidad y respuesta en tiempo real.

2. Generar y preprocesar un conjunto de datos para el entrenamiento de modelos de identificación de objetos.

3. Adaptar y entrenar un modelo de identificación de objetos.

4. Generar la visualización del mapa de profundidad en tiempo real.

5. Integrar el sistema y realizar pruebas de localización de objetos en un ambiente controlado.

1.3. Justificación

El desarrollo del presente trabajo responde a una problemática de salud pública latente y de escala global. De acuerdo con información de la Organización Mundial de la Salud, se reporta que existen aproximadamente 2,200 millones de personas con limitaciones visuales [8]. En el contexto nacional, el panorama no es muy diferente; según el Censo de Población y Vivienda 2020 del INEGI, la discapacidad visual es la limitación más frecuente en México, afectando al 43.5% de la población con discapacidad, lo que representa a más de 2.6 millones de personas [9]. Estas estadísticas se acentúan en el estado de Puebla, donde la incidencia de limitaciones visuales asciende al 46.3% de las discapacidades registradas [10]. Estas cifras ponen en evidencia la necesidad de implementar soluciones tecnológicas enfocadas a resolver y mitigar las barreras de inclusión y movilidad que tiene este sector de la población.

En este sentido, la implementación de tecnología destinada a personas con discapacidad visual no solo representa un reto técnico, sino un compromiso social alineado con los Objetivos de Desarrollo Sostenible (ODS), específicamente con el ODS 3 (Salud y bienestar) y el ODS 10 (Reducción de las desigualdades) [7].

Desde una perspectiva técnica, el sistema que se propone en este trabajo aborda los puntos débiles de los métodos tradicionales. Si bien la detección de obstáculos ha sido explorada en otros trabajos, estas soluciones presentan limitaciones significativas como: elevada carga computacional, latencia alta, consumo energético ineficiente y un factor de forma poco ergonómico. Así mismo, en el contexto del aprendizaje automático, los enfoques basados en la nube comprometen la privacidad del usuario al requerir la transmisión y procesamiento externo de datos sensibles del entorno.

Para solventar estos desafíos, el presente trabajo propone un enfoque fundamentado en la unión de dos tecnologías emergentes: los sensores de Tiempo de Vuelo (ToF) y el paradigma de Aprendizaje Automático Integrado (TinyML). La investigación preliminar permitió identificar que el procesamiento de datos de profundidad ToF representa una alternativa prometedora, pues demanda una carga computacional significativamente menor que las imágenes RGB de alta resolución, y al mismo tiempo, proporciona información de la distancia a los objetivos. Finalmente, al implementarse un modelo TinyML, se protege la privacidad del usuario y de terceros, ya que no se almacenan rasgos faciales y los datos se procesan en el mismo dispositivo, sin necesidad de una comunicación externa

1.4. Estado del arte

Para establecer la propuesta técnica de esta tesis, se ha realizado una revisión minuciosa de la literatura enfocada en sistemas de detección mediante sensores ToF y su integración con modelos TinyML. El análisis parte de la comprensión fundamental de la tecnología de tiempo de vuelo, avanzando hacia implementaciones complejas en sistemas embebidos y dispositivos de asistencia.

Tabla 1-1. Tabla comparativa de trabajos previos para la evaluación y caracterización de sensores ToF monozona.

Título del artículo	Descripción	Sensor utilizado	Características clave	Desafíos abordados
Aplicación del sensor ToF VL53L0X de bajo costo para la detección del entorno de robots [11].	Evaluación del sensor VL53L0X para la detección de proximidad en robótica, incluyendo su desempeño en varios materiales y condiciones de iluminación.	VL53L0X	• Bajo costo. • Tamaño pequeño. • Confiable en detección de proximidad. • Adecuado para robótica.	• Impacto de la reflectividad de la superficie. • Iluminación ambiental en la precisión de las mediciones.
Caracterización de un sensor de profundidad IR miniaturizado con una región de interés programable que permite aplicaciones de cartografía de riesgos [12].	Caracterización del sensor ToF VL53L1X para aplicaciones portátiles de mapeo de riesgos, evaluando su resolución espacial e integración con IMU para mapeo de profundidad.	VL53L1X	• Región de interés programable. • Bajo consumo. • Preciso hasta 4m.	• Interferencia de luz ambiental. • Miniaturización para aplicaciones móviles.
Caracterización de un sensor LiDAR portátil para distanciamiento social [13].	Estudio del desempeño de un sensor LiDAR portátil para mantener el distanciamiento social, con enfoque en los impactos ambientales.	VL53L1X	• Diseño compacto. • Alcance adecuado para distanciamiento social. • Aplicación portátil.	• Impacto de la intensidad de la luz. • El color del objeto. • Las condiciones ambientales en el desempeño.
Calibración geométrica de sensores de distancia de un solo píxel [14].	Desarrollo de un procedimiento de calibración para sensores de distancia de un solo píxel para mejorar su integración con sistemas robóticos.	VL6180X, VL53L3CX	• La calibración mejora la precisión. • Permite mapeo 3D.	• Estimación de la posición del sensor. • Alineación geométrica para mediciones precisas.
Medición láser de alta precisión basada en microcontrolador STM32 [15].	Desarrollo de un sistema portátil de medición láser con STM32 y VL53L0X, con modos de alta precisión y largo alcance.	VL53L0X	• Doble modo para alta precisión. • Alcance extendido, integración con STM32.	• Equilibrar precisión. • Rendimiento de alcance en diferentes entornos.

En primera instancia, es importante analizar el comportamiento físico de los sensores ToF monozona (series VL53L0X, VL53L1X y VL53L3CX), los cuales sirven como base tecnológica para sistemas más avanzados. Como se resume en la Tabla 1-1, diversos trabajos [11], [12], [13] se han centrado en la caracterización y evaluación de estos dispositivos, sometiéndolos a pruebas bajo distintas condiciones de iluminación ambiental y frente a objetos de diversos materiales y reflectividades. Estos estudios permiten identificar las limitaciones inherentes de la tecnología óptica ante interferencias externas. Por otro lado, investigaciones como las presentadas en [14] y [15] profundizan en la calibración geométrica y la mejora de la precisión milimétrica, demostrando que, si bien estos sensores son robustos para mediciones puntuales de distancia, carecen de la resolución espacial necesaria para interpretar la geometría de un entorno complejo. Esta revisión establece el punto de partida técnico para justificar la transición hacia sensores ToF multizona.

Para superar las limitaciones que presentan los sensores monozona, una segunda vertiente de investigación se enfoca en la utilización de sensores ToF multizona integrados en sistemas de cómputo en el borde (*Edge AI*). La Tabla 1-2 resume una serie de trabajos recientes que utilizan esta tecnología para implementar aprendizaje automático en sistemas empotrados. En esta categoría, se destacan proyectos de alto rendimiento como la presentada en [16], donde se emplea una FPGA para ejecutar redes neuronales convolucionales (CNN) con datos de un sensor VL53L8CX, o el proyecto NanoSLAM [17], que utiliza un SoC RISC-V para mapeo y localización simultánea en drones.

Así mismo, se ha explorado la viabilidad de estas técnicas en hardware con menores recursos de procesamiento. Investigaciones como [18], [19], [20] y [21] demuestran que es posible ejecutar modelos TinyML en microcontroladores de propósito general (como el RP2040 o la serie STM32 Cortex-M) para clasificar objetos simples o detectar obstáculos. Sin embargo, un aspecto común y una limitante en todos estos trabajos es la resolución del sensor: al depender de matrices de máximo 8x8 zonas (64 píxeles en total), la capacidad de los modelos para identificar formas geométricas complejas se ve muy limitada. Si bien estos sistemas se destacan por su bajo consumo de energía y costo, la escasa cantidad de información espacial impide una clasificación detallada del entorno, relegándolos a tareas de detección básica de proximidad.

Tabla 1-2. Tabla comparativa de trabajos que implementan sensores ToF multizona (8x8) compactos y modelos TinyML.

Título del artículo	Descripción	Sensores/Dispositivos	Técnicas aplicadas	Resultados
Clasificación multiclase de objetos mediante sensores de tiempo de vuelo de resolución ultrabaja [16].	Clasificación multiclase usando sensores ToF con redes optimizadas para procesamiento en hardware embebido.	• VL53L8CX (8x8 píxeles) • FPGA Artix-7	• CNN compacta optimizada para FPGA	• Precisión del 90.21% en 10 clases. • Consumo de 65.6 nJ/inferencia.
NanoSLAM: SLAM a bordo para robots diminutos [17].	SLAM optimizado para drones nano-UAV con sensores ToF.	• VL53L5CX • SoC RISC-V GAP9 • UAV Crazyflie 2.1	• SLAM basado en gráficos.	• Precisión de 4.5 cm en mapeo. • Consumo de 87.9 mW.
Detección de robots móviles en miniatura mediante un sensor de tiempo de vuelo de resolución ultrabaja [18].	Detección de robots móviles en escenarios reales usando un sensor ToF de baja resolución.	• VL53L5CX (8x8) • MCU RP2040	• Clasificación binaria con CNN	• Precisión del 91.8%. • Consumo energético de 3.685 mJ por inferencia.
Detección de la profundidad de ratones con redes neuronales diminutas desplegables en jaulas pequeñas [19].	Detección de formas de ratones en jaulas pequeñas usando redes neuronales pequeñas implementadas en microcontroladores.	• VL53L8CX • NUCLEO-STM32U5	• Redes tipo U-Net, Micro-net optimizadas para MCUs	• Precisión de 86%. • Inferencias en 0.27 ms con uso de 0.5 kB RAM y 1.3 kB FLASH.
Clasificación de objetos mediante un sensor de tiempo de vuelo de resolución ultrabaja y una minúscula red neuronal convolucional [20].	Clasificación de objetos con sensor ToF de baja resolución y red neuronal convolucional.	• VL53L8CX • STM32 Cortex-M33 • STM32 Cortex-M4	• CNN con cuantización a 8 bits	• Precisión del 92.23%. • Inferencias en 1 ms con 32 µJ de energía. • Red ocupando 2.65 kB de memoria.
Pequeño modelo de aprendizaje automático (TinyML) para la detección de obstáculos con sensores de tiempo de vuelo multizona [21].	Detección de obstáculos con sensores ToF multi-zona usando TinyML optimizado para MCU con bajo uso de memoria.	• Sensor ToF multizona • STM32F746NG	• Modelo CNN	• Precisión >90% para 6 clases. • Uso de memoria menor a 100 kB.

Finalmente, el análisis contempla los sistemas de asistencia integral, abarcando tanto prototipos académicos como productos comerciales. La Tabla 1-3 presenta desarrollos recientes que buscan una navegación autónoma completa. En este ámbito, la tendencia predominante es el uso de hardware de procesamiento robusto; por ejemplo, el sistema DeepEye [22] y otras propuestas [23], [24], [25] dependen de plataformas de alto rendimiento como NVIDIA Jetson o computadoras portátiles (laptops) para procesar flujos de video RGB y mapas de profundidad.

Tabla 1-3. Tabla comparativa de trabajos académicos enfocados a la asistencia guiada de personas con discapacidad visual.

Título del artículo	Descripción	Dispositivos utilizados	Técnicas aplicadas	Resultados obtenidos
DeepEye: Diseño e implementación de un sistema embebido para percepción del entorno y navegación de personas con discapacidad visual[22].	Sistema que detecta obstáculos mediante segmentación de instancias en tiempo real y proporciona indicaciones por voz para una navegación segura en exteriores.	• Jetson AGX Orin • Cámara RGB • Módulo de audio Bluetooth • Hardware embebido portátil	• *Deep learning* con YolactEdge para segmentación de instancias. • Estimación de distancia por cuadrícula. • Síntesis de voz (TTS).	• Precisión media promedio = 44 % y 40,5 FPS. • Demostraciones exitosas de guiado y reducción de colisiones.
Integración de la háptica portátil y la evitación de obstáculos en la navegación en interiores para personas con discapacidad visual: Un enfoque centrado en el usuario [23].	Sistema de navegación en interiores que combina cámara RGB-D y una banda háptica (CUFF) para guiar al usuario mediante fuerzas normales y tangenciales, diseñado con participación de personas con discapacidad visual.	• Cámara RGB-D Asus Xtion Pro • Ordenador portátil • Dispositivo háptico CUFF • Bastón blanco opcional	• Detección de esquinas y cálculo de profundidad para evitación de obstáculos. • Retroalimentación háptica con fuerzas normal y tangencial.	• 97 % de aciertos en reconocimiento de estímulos. • Tiempos de reacción ≈ 1 s. • Usuarios reportan menor fatiga y buena usabilidad.
Guía de caminar al aire libre para personas con discapacidad visual basada en la segmentación semántica y el mapa de profundidad [24].	Dispositivo portátil para exteriores que estima la transitabilidad del terreno combinando segmentación semántica y mapas de profundidad, generando indicaciones según la confianza de caminabilidad.	• Cámara y sensor de profundidad integrados en un módulo *wearable*.	• Segmentación semántica basada en redes profundas. • Cálculo de '*walkability confidence*' usando mapa de profundidad. • Algoritmo de guiado multimodal.	• Prototipo validado en distintas superficies. • Sin métricas cuantitativas publicadas.
Sistema de navegación portátil para invidentes y personas con discapacidad visual [25].	Plataforma portátil que crea mapas con SLAM y guía al usuario mediante audio por conducción ósea, integrando sensores láser e IMU para entornos desconocidos en interior y exterior.	• Escáner láser de corto alcance. • IMU montada en el pie. • Computador *wearable.* • Auriculares de conducción ósea.	• SLAM fusionado con *Pedestrian Dead Reckoning*. • Planificación de rutas y fusión multisensorial.	• Concepto verificado con datos simulados. • Pruebas con usuarios planificadas.

A continuación, en la Figura 1.2 se observan tres propuestas de dispositivos con aplicaciones similares a las que se proponen en este trabajo de tesis.

a) En el área comercial y de transferencia tecnológica, se encuentra el dispositivo denominado SmartOjo [26], que comenzó como un proyecto desarrollado por ingenieros del Centro de Investigación y de Estudios Avanzados (CINVESTAV) en 2014. En el cual, se implementaban

sensores de sonido estéreo y tecnología GPS para guiar al usuario con discapacidad visual a un punto específico y evitar chocar con obstáculos estáticos o en movimiento. Posteriormente, se agregaron más funciones, incorporando técnicas modernas como el aprendizaje profundo (*Deep Learning).* Sin embargo, tras su transición al sector privado en 2020, no se han publicado actualización del estado del proyecto, lo que pone en evidencia los desafíos de sostenibilidad y continuidad que muchas veces enfrentan estos desarrollos tecnológicos en áreas de investigación.

Figura 1.2. Gafas inteligentes para personas con discapacidad visual [25-27].

b) SuperBrain 1 [27] es un dispositivo desarrollado por la startup 7Sense con sede en Estonia, el cual integra cámaras 3D y cámaras de video (RGB y térmica), estos componentes transmiten varios megabytes de datos por segundo a la unidad de procesamiento principal que está en la parte posterior del dispositivo, que combina todos estos datos en un solo paquete, después, se realiza un análisis para hacer el formato lo más compacto posible y posteriormente se envía a los actuadores apticos ubicados en la frente del usuario. Estos actuadores están distribuidos en forma de matrices físicas que, mediante pequeños pulsos y variando la frecuencia de vibración en función de la distancia del objetivo, dibujan el objeto que la persona tiene frente a ella. Actualmente se pueden adquirir por un costo de 9000 euros, asiéndolos inaccesibles para muchos sectores de la población.

c) La startup .LUMEN [28] con sede en Rumania, presenta un prototipo con características y especificaciones similares a SuperBrain 1. Su diseño tiene forma de diadema para ser colocado alrededor de la cabeza del usuario. El dispositivo integra sensores y cámaras de profundidad para la detección de objetos, así como cámaras RGB para complementar la información. Actualmente aún no se comercializa, pero se puede hacer una solicitud para probar el dispositivo, sólo para usuarios que están dentro de Estados Unidos o Rumania.

Los dos últimos dispositivos [27] y [28] no mencionan explícitamente el sistema que realiza el procesamiento de las imágenes, sin embargo, en ambos casos se menciona que tienen una alianza con la empresa norteamericana NVIDIA, la cual tiene gran relevancia en el campo de la IA, considerando lo anterior, lo más probable es que todos los procesos de cómputo los realice una tarjeta de la serie Jetson, enfocadas a robótica y *Edge AI*.

Observaciones importantes

El análisis de la literatura revela una brecha tecnológica clara. Actualmente, las soluciones se pueden ubicar en dos categorías:

- **Sistemas con recursos limitados:** Basados en sensores ToF de muy baja resolución (8x8 zonas) o ultrasonido, que son eficientes y privados pero incapaces de clasificar la geometría del entorno.
- **Sistemas con recursos avanzados:** Basados en visión artificial compleja (SmartOjo, Lumen, DeepEye), que son precisos pero costosos, demandan mayor energía y podrían resultar intrusivos para la privacidad del usuario y de terceros.

Hasta el momento de realizar este análisis, no se identificaron trabajos que se enfoquen en un punto medio: el uso de sensores ToF de buena resolución (como 50x50 o 100x100 puntos) procesados exclusivamente mediante modelos TinyML en microcontroladores de bajo consumo. Esta es el área de oportunidad que el presente trabajo de tesis pretende abordar.

1.5. Descripción del sistema

El sistema propuesto consiste en un dispositivo electrónico portátil diseñado para la asistencia en la movilidad de personas con discapacidad visual. Su arquitectura se fundamenta en el procesamiento en el borde, eliminando la dependencia de conectividad externa para mejorar la respuesta en tiempo real y la privacidad del usuario.

Como se ilustra en el diagrama de bloques general de la Figura 1.3, el sistema se compone de cuatro bloques funcionales principales: adquisición de datos, procesamiento central, gestión de energía e interfaz de usuario.

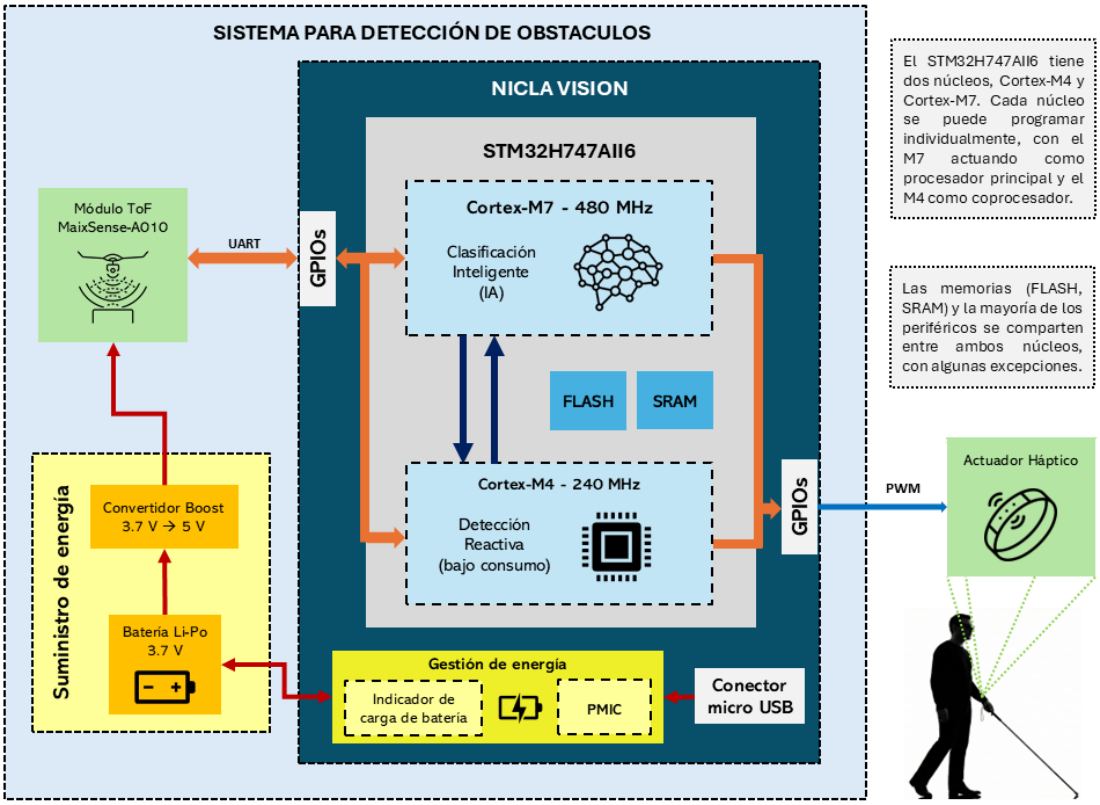

Figura 1.3. Diagrama de bloques del sistema propuesto.

El bloque de adquisición está constituido por el sensor de Tiempo de Vuelo (ToF) matricial MaixSense A010. Este dispositivo es el encargado de percibir el entorno, generando mapas de profundidad con una resolución de 50x50 puntos. Este sensor opera a una frecuencia de 940 nm, permitiendo la detección de obstáculos en condiciones de oscuridad. Los datos de distancia

adquiridos son transmitidos mediante una interfaz UART de alta velocidad hacia la unidad de procesamiento central.

El núcleo del sistema es la tarjeta de desarrollo Arduino Nicla Vision, que integra un microcontrolador STM32H747XI de arquitectura Arm dual-core. Esta unidad central gestiona el flujo de datos y la ejecución de los algoritmos mediante una comunicación inter-núcleo. El funcionamiento utiliza una arquitectura de software híbrida diseñada para optimizar los recursos:

- Modo de Detección Reactiva (Bajo Consumo): Ejecutado en el núcleo Cortex-M4. Procesa los datos de profundidad para detectar la proximidad inmediata de obstáculos.
- Modo de Clasificación Inteligente (IA): Ejecutado en el núcleo Cortex-M7. Cuando se requiere, activa una Red Neuronal Convolucional (CNN) ligera (TinyML) para clasificar la naturaleza del obstáculo basándose en la geometría del mapa de profundidad.

El bloque de gestión de energía asegura la autonomía del dispositivo. El sistema es alimentado por una batería Li-Po, cuya carga y estado son monitoreados por el circuito integrado MAX17262 a través de I2C. La regulación y distribución de los diferentes niveles de voltaje requeridos por los componentes están a cargo del Circuito Integrado de Gestión de Energía (PMIC) MC34PF1550 y el convertidor boost MT3608, permitiendo una operación estable y eficiente.

Nota: Los circuitos integrados MAX17262 y MC34PF1550 están implementados dentro de la tarjeta Nicla Vision, lo que permite aprovechar sus capacidades para gestionar la carga y descarga de la batería Li-Po. Mediante el conector micro USB de la tarjeta se puede depurar el código y, al mismo tiempo, cargar la batería del sistema.

Finalmente, la interfaz de usuario (actuador háptico) se integra con fines demostrativos para validar las salidas del sistema de procesamiento. Aunque la arquitectura permite la expansión a matrices complejas, el alcance de este trabajo se limita a la implementación de un arreglo lineal de tres motores de vibración controlados por señales PWM. Este prototipo simplificado es suficiente para proporcionar indicaciones direccionales básicas (izquierda, centro, derecha) y alertas de clasificación durante las pruebas experimentales, dejando el desarrollo de una interfaz háptica de alta resolución, así como el estudio de su ergonomía, para una línea de trabajo futuro.

1.6. Metodología de trabajo

El plan de trabajo para desarrollar el sistema propuesto se compone de varias etapas, clasificadas en cuatro fases:

- Fase I: Investigación y requerimientos.
- Fase II: Ingeniería de datos y caracterización.
- Fase III: Desarrollo de IA y algoritmia.
- Fase IV: Integración de sistemas y validación.

Este plan se divide en tareas específicas que permiten un seguimiento detallado del progreso del proyecto. En ciertas circunstancias, es posible trabajar simultáneamente en más de una etapa, sobre todo durante el entrenamiento y evaluación del modelo ML.

En la Figura 1.4 se presenta el diagrama metodológico que guía el progreso del proyecto.

Figura 1.4. Diagrama metodológico de actividades.

Fase I: Investigación y requerimientos

En la etapa inicial se define el problema que se busca resolver y se consideran las posibles soluciones. Aunque pueda parecer sencillo, esta es probablemente la parte más importante al

comenzar un proyecto. Una vez que se tiene claro este apartado, es momento de pasar al estado del arte, esta etapa se enfoca en la investigación y análisis de los trabajos previos, así como en el panorama actual relacionado al tema de tesis propuesto. Esto es fundamental para entender cómo otros han abordado el problema y qué oportunidades de mejora existen. Después de revisar esta información, se podrá determinar si el trabajo que se pretende realizar tiene una contribución relevante o de lo contrario, se debe replantear la idea inicial. Finalmente, se establecen los requerimientos del sistema, basándose en la investigación previa, se identificarán los requisitos o elementos necesarios para alcanzar los objetivos propuestos.

Fase II: Ingeniería de datos y caracterización

En esta fase se realiza el estudio técnico de los componentes físicos. Para lo cual, se realiza una nueva investigación, enfocada en seleccionar los componentes adecuados según las características previamente definidas. Una vez seleccionados estos componentes, se procederá a realizar pruebas individuales para comprender su funcionamiento y rendimiento en diferentes situaciones. Debido a que no existen bases de datos de acceso público con imágenes de profundidad ToF, será necesario diseñar e implementar un prototipo de recolección de datos. Este sistema permite la construcción de un dataset propio, capturando y etiquetando mapas de profundidad en entornos reales para entrenar las clases de objetos de interés.

Fase III: Desarrollo de IA y algoritmia

El núcleo de inteligencia del sistema (modo IA) se desarrolla en esta fase siguiendo el flujo de trabajo de TinyML. Primero, se realiza la adaptación de una arquitectura de modelo ML (como una CNN ligera) capaz de procesar imágenes de profundidad bajo las restricciones de memoria de un microcontrolador. Posteriormente, se procede al entrenamiento y evaluación del modelo, ajustando hiperparámetros para maximizar la precisión. Finalmente, se realiza la optimización y conversión a TinyML, aplicando técnicas de cuantización y compiladores especializados para reducir el tamaño del modelo y asegurar su compatibilidad con la arquitectura del procesador.

Fase IV: Integración de sistemas y validación

La fase final comprende el ensamblaje y la puesta en marcha del prototipo funcional. El proceso inicia con el despliegue del modelo en la Nicla Vision, cargándolo específicamente en el núcleo Cortex-M7. Simultáneamente, se desarrolla la lógica de traducción háptica, definiendo los patrones de vibración para el usuario. Un paso importante es el diseño del software del sistema, donde se implementa la comunicación RPC entre ambos núcleos para una operación dual-core. Posteriormente, se realiza la integración del hardware final del prototipo. El proyecto concluye con la implementación de pruebas bajo condiciones controladas para medir el rendimiento y un análisis de resultados detallado que identifique oportunidades de mejora a futuro.

1.7. Organización del documento

El presente trabajo de tesis se encuentra estructurado en cinco capítulos principales, organizados secuencialmente para guiar al lector desde los fundamentos teóricos hasta la validación experimental del prototipo desarrollado. A continuación, se describe brevemente el contenido de cada sección:

El Capítulo 2: Marco Teórico, establece las bases conceptuales necesarias para la comprensión del proyecto. Se profundiza en el principio de funcionamiento de los sensores de tiempo de vuelo (ToF), distinguiendo entre las tecnologías directa e indirecta, así como el análisis de errores inherentes a esta tecnología. Asimismo, se abordan los fundamentos del Aprendizaje Automático Integrado (TinyML), detallando las arquitecturas de redes neuronales convolucionales (CNN) y las estrategias de optimización para su despliegue en sistemas empotrados.

En el Capítulo 3: Desarrollo e implementación del sistema, se describe la metodología de diseño y la ingeniería del proyecto. Esta sección abarca la caracterización y selección de los componentes de hardware (tarjeta Nicla Vision y sensores ToF), el diseño del protocolo para la recolección y limpieza de datos, y la construcción del dataset. De igual forma, se detalla la arquitectura de la red neuronal propuesta, el entorno de entrenamiento y la configuración inicial de los entornos de desarrollo y protocolos de comunicación necesarios para la integración del sistema.

El Capítulo 4: Resultados, presenta la evidencia experimental y el análisis del desempeño del prototipo. Se exponen los resultados de la caracterización de los sensores VL53L8CH y MaixSense A010, así como las métricas de entrenamiento y validación del modelo de clasificación, incluyendo una comparativa de rendimiento entre distintas técnicas de cuantización y compilación. En este capítulo se detalla también la arquitectura final del *firmware* implementado bajo un esquema de procesamiento asimétrico (núcleos Cortex-M4 y Cortex-M7), la integración física de los elementos y los resultados de las pruebas funcionales realizadas en ambiente controlado.

Finalmente, el Capítulo 5: Conclusiones, sintetiza los hallazgos más relevantes de la investigación, validando el cumplimiento de los objetivos planteados. Se discuten las implicaciones de los resultados obtenidos y se proponen oportunidades de mejora y líneas de trabajo a futuro para la evolución del prototipo hacia un dispositivo comercial.

Capítulo 2

2. Marco Teórico

El presente capítulo establece los fundamentos teóricos sobre la tecnología de tiempo de vuelo (ToF), definiéndola como un método de detección activa que emplea señales infrarrojas para obtener información precisa de profundidad. Se analizan las diferencias operativas entre los métodos de medición directos e indirectos, así como la arquitectura interna de los módulos compactos que integran emisores VCSEL y detectores SPAD para generar mapas espaciales. Asimismo, se examinan las fuentes de error sistemáticas y no sistemáticas que afectan la precisión del sensor, subrayando la importancia de implementar técnicas de mitigación frente al ruido y las interferencias ambientales para mejorar la calidad de los datos recolectados.

Posteriormente, se aborda el paradigma del aprendizaje automático integrado (TinyML), el cual permite la ejecución de modelos de inteligencia artificial directamente en el borde utilizando microcontroladores de ultra baja potencia. Se describen las arquitecturas de redes neuronales convolucionales (CNN) diseñadas para procesar datos con estructura de rejilla de forma eficiente, optimizando el uso de memoria RAM y FLASH mediante operaciones de convolución y capas de agrupamiento. Finalmente, se detalla el ciclo de vida de los datos y los procesos de optimización esenciales, como la cuantización de 8 bits, que transforman modelos complejos en soluciones viables para sistemas embebidos que funcionan en tiempo real.

2.1. Sensores de tiempo de vuelo

La tecnología de Tiempo de Vuelo (Time-of-Flight, ToF) se define como un método de detección activa para la obtención de información de profundidad, basado en la medición del tiempo que tarda una señal luminosa, usualmente en el espectro infrarrojo (IR), en viajar desde un emisor hasta un objeto y regresar a un detector [5], [6]. A diferencia de los sistemas de visión estereoscópica que dependen de complejos algoritmos de correspondencia de píxeles, los sensores ToF calculan la distancia de forma directa o indirecta mediante la velocidad de la luz, lo que reduce significativamente la carga computacional en sistemas embebidos.

Dependiendo de la técnica de modulación y detección empleada, los sistemas ToF se clasifican principalmente en dos categorías: tiempo de vuelo indirecto y tiempo de vuelo directo.

A continuación, se profundizará más a detalle en qué consisten estos métodos.

2.1.1. Tiempo de vuelo indirecto (iToF)

El método iToF no mide el tiempo de viaje de los fotones de forma explícita. En su lugar, emite una onda de luz continúa modulada (frecuentemente sinusoidal o cuadrada) y mide el desplazamiento de fase entre la señal emitida y la reflejada. Aunque esta técnica permite alcanzar resoluciones espaciales de nivel VGA, presenta desventajas críticas para aplicaciones en exteriores, como la susceptibilidad a la interferencia por luz ambiental y un mayor consumo energético debido a la necesidad de una emisión constante de luz [6]. En la Figura 2.1 se observa el principio de funcionamiento de este método.

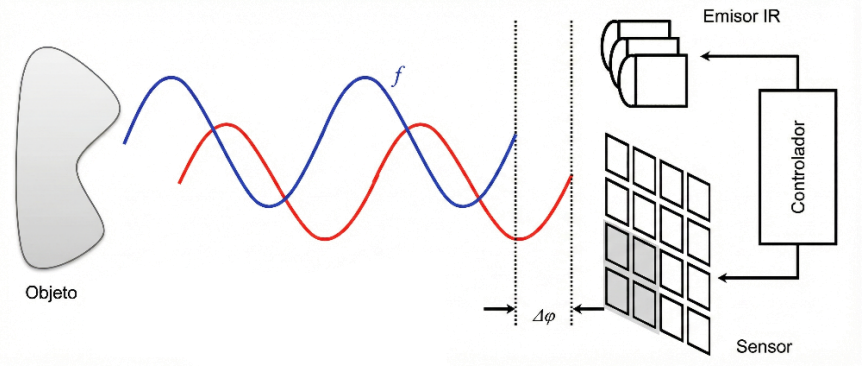

Figura 2.1. Principio de funcionamiento del sensor iToF [6].

La relación entre la diferencia de fase $\Delta\phi$ y la distancia se obtiene al aplicar la Ecuación 1.

$$d = \frac{c \cdot \Delta\phi}{4\pi f} \qquad \textit{Ecuación 1.}$$

donde f es la frecuencia de modulación.

2.1.2. Tiempo de vuelo directo (dToF)

El método dToF, utilizado en sensores como el MaixSense A010 y la serie VL53L8CX de STMicroelectronics, se basa en la emisión de pulsos láser muy cortos y la medición directa del tiempo transcurrido hasta su recepción mediante cronómetros digitales de alta precisión denominados Convertidores de Tiempo a Digital (*Time-to-Digital Converters, TDC*).

Este proceso se apoya en detectores de alta sensibilidad llamados Diodos de Avalancha de Fotón Único (*Single-Photon Avalanche Diodes, SPAD*), los cuales permiten detectar el arribo de un único fotón y generar una respuesta de temporización precisa [5], [29], [30].

La Figura 2.2 muestra este principio de funcionamiento.

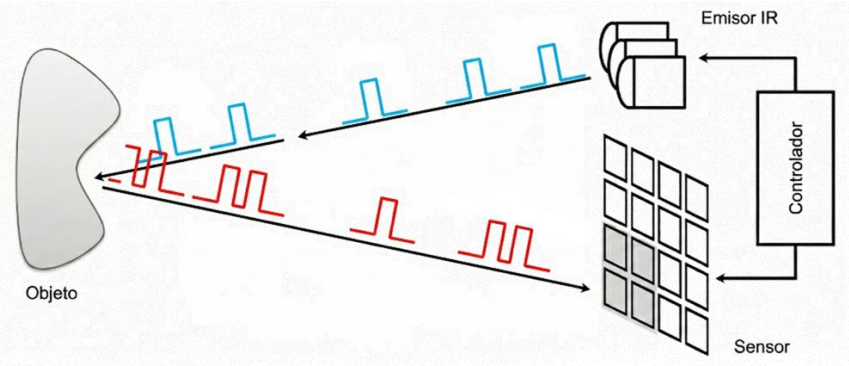

Figura 2.2. Principio de funcionamiento del sensor dToF.

Al medir el tiempo transcurrido entre la emisión y la recepción de la señal, es posible calcular la distancia al objeto utilizando la Ecuación 2.

$$d = \frac{c \cdot t}{2} \qquad \textit{Ecuación 2.}$$

donde:

- d es la distancia al objeto.
- c es la velocidad de la luz en el medio (aproximadamente $3 \times 10^8\ m/s$ en el aire).
- t es el tiempo de vuelo medido.

El factor $\frac{1}{2}$ se incluye porque el tiempo medido corresponde al recorrido de ida y vuelta de la señal.

2.1.3. Arquitectura multizona y representación de la información de distancia

La evolución de los sensores ToF ha permitido pasar de mediciones puntuales a la generación de mapas espaciales complejos, fundamentales para la navegación autónoma y la asistencia sensorial. Estos dispositivos se colocan en módulos compactos (todo en uno) que integran un emisor láser de cavidad vertical y emisión superficial (VCSEL), una etapa óptica especializada, un microcontrolador embebido y arreglos de detectores de fotón único (SPAD) [29], [31]. Esta arquitectura interna del módulo se puede observar en la Figura 2.3.

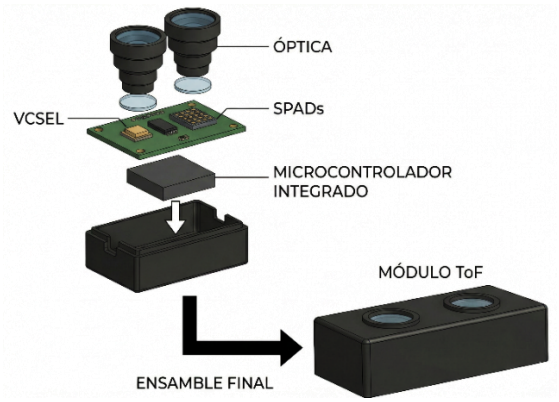

Figura 2.3. Arquitectura interna de un sensor ToF compacto.

En las siguientes subsecciones se examinan los conceptos fundamentales necesarios para comprender el principio de operación de los dispositivos ToF comerciales. El análisis se enfoca en el método de tiempo de vuelo directo (dToF), por ser la técnica predominante en sensores de distancia compactos debido a su capacidad para realizar mediciones de alta velocidad con un consumo energético reducido.

2.1.3.1. Sensores monozona frente a sistemas multizona

Los sensores de una sola zona (monozona) están diseñados para medir una distancia única (o múltiples objetivos promediados) dentro de su campo de visión, pero carecen de la capacidad de proporcionar las coordenadas espaciales de los objetos detectados. Por el contrario, los sensores multizona dividen su arreglo de detectores SPAD en múltiples regiones independientes, permitiendo el mapeo de profundidad. En estos sistemas, cada zona actúa como un píxel que reporta su propia distancia, generando una matriz de información geométrica de la escena [29].

En la Figura 2.4 se coloca de forma gráfica una comparativa del funcionamiento del sensor ToF monozona y el multizona.

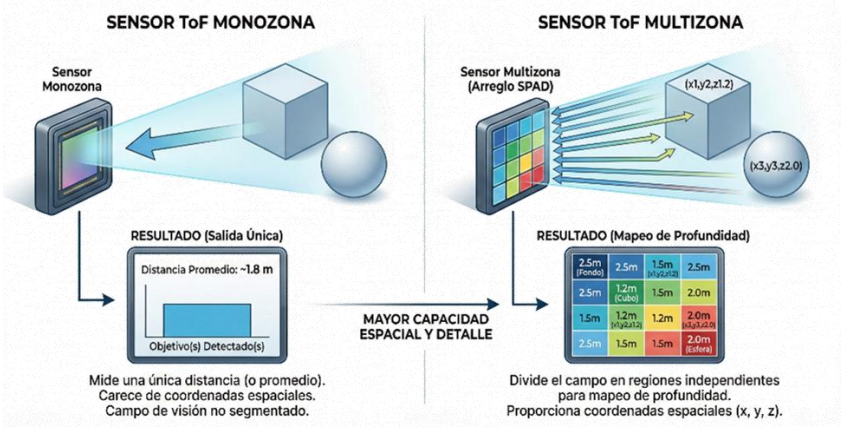

Figura 2.4. Comparativa en el funcionamiento de sensores ToF monozona y multizona.

Los sensores más recientes, como el VL53L8CH, permiten resoluciones de 8×8 zonas (64 zonas totales), o el módulo MaixSense A010, que llega hasta 100×100 zonas (10 000 zonas totales) en configuraciones nativas [32].

2.1.3.2. Interpretación de histogramas ToF

La documentación técnica describe la salida de estos sensores no solo como valores numéricos, sino como Histogramas Normalizados Compactos (*Compact Normalized Histogram, CNH*), como se muestra en la Figura 2.5. Un histograma ToF registra la intensidad de la señal de retorno (eje Y) frente al rango de distancia o tiempo (eje X) [29].

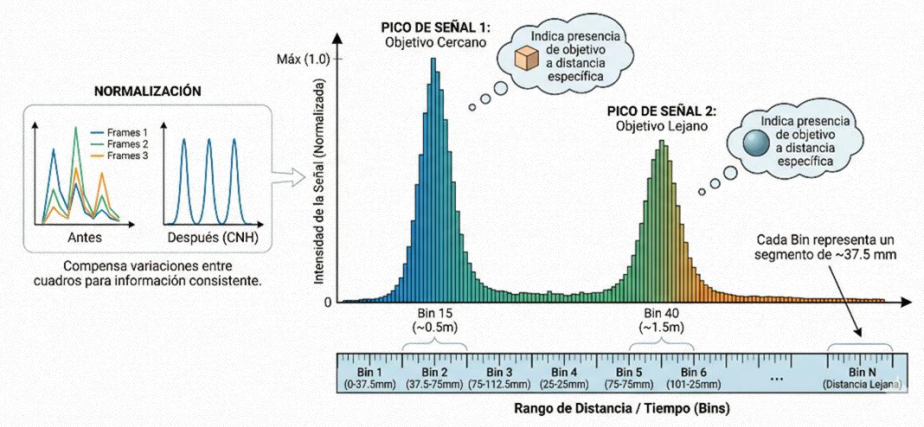

Figura 2.5. Interpretación de histogramas ToF.

Un pico en el histograma indica la presencia de un objetivo a una distancia específica. Los datos se ajustan para compensar variaciones entre cuadros, lo que permite que el modelo de ML reciba información consistente independientemente de los ajustes automáticos del sensor. El rango de distancia se divide en contenedores (bins); por ejemplo, en ciertas configuraciones del sensor, cada bin representa un segmento de aproximadamente 37.5 mm [30].

- **Número de contenedores (bins).** Son los pequeños contendores del histograma que produce el sensor ToF, cada bin acumula los fotones que llegan desde una franja de distancia fija. Cuantos más bins se habiliten, mayor será el alcance en profundidad de la escena.

2.1.3.3. Representación y normalización de imágenes

Para que un modelo de aprendizaje automático procese correctamente estos datos, la información de profundidad debe ser normalizada y representada visualmente.

Mientras que sensores como el MaixSense A010 suelen entregar mapas de profundidad en formatos de 8 bits, dispositivos de alta precisión como el VL53L8CH operan con una profundidad de 16 bits para registrar distancias en milímetros con mayor rango dinámico [29], [32].

Los valores de distancia pueden mapearse a una escala de grises, donde los tonos más oscuros representen proximidad y los claros lejanía, véase la Figura 2.6.

También es posible combinar datos de múltiples zonas físicas en estructuras de datos agregadas para optimizar el ancho de banda sin perder la coherencia espacial necesaria para aplicaciones de IA.

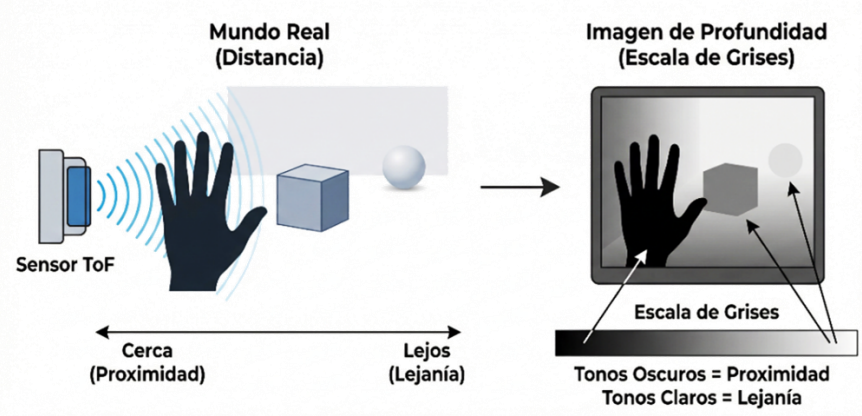

Figura 2.6. Mapeo de distancia a escala de grises.

2.1.4. Análisis de errores en sensores ToF y técnicas de mitigación

Las imágenes de profundidad producidas por los sensores ToF presentan problemas característicos que se dividen en errores sistemáticos y no sistemáticos. Los errores sistemáticos, como el ruido y la ambigüedad, están directamente relacionados con el sensor, mientras que los errores no sistemáticos, como el esparcimiento y el desenfoque por movimiento, están más relacionados con el contenido de la escena [6], [33].

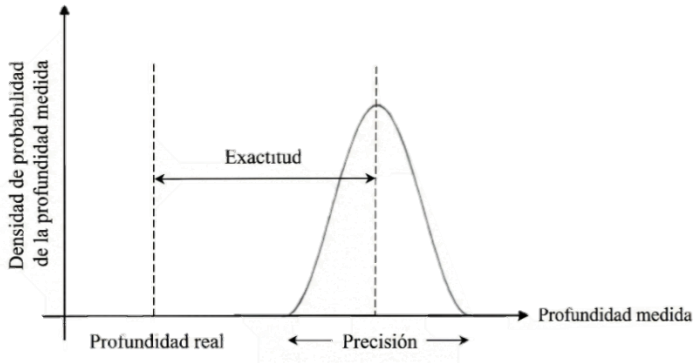

Figura 2.7. Ilustración de los términos exactitud y precisión de la profundidad medida [6].

Estas fuentes de error degradan la precisión y exactitud de las mediciones de rango. La exactitud se refiere a la proximidad de las mediciones al valor real, mientras que la precisión se refiere a la consistencia de las mediciones entre sí. En la Figura 2.7 se muestra de forma gráfica un ejemplo de estos dos conceptos.

Por otro lado, el ruido se manifiesta como desviaciones en magnitud (ΔA), mientras que las fluctuaciones se manifiestan como variaciones en el tiempo (Δt). Ambas afectan la calidad de las mediciones [6].

En la Figura 2.8 se muestra un ejemplo de esta señal de ruido comparada con una señal ideal.

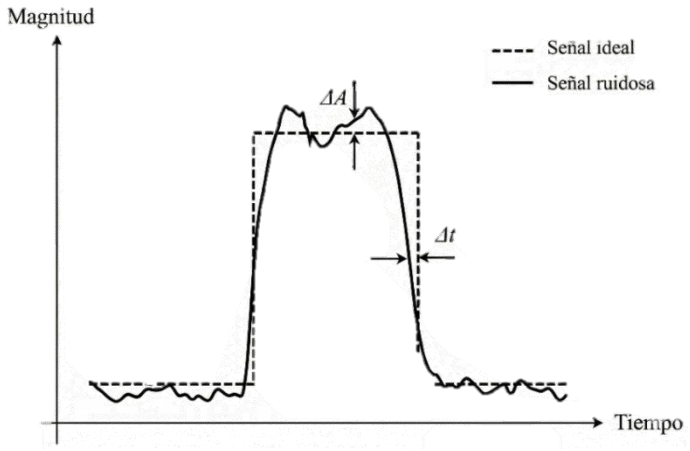

Figura 2.8. Comparativa entre una señal ruidosa y su señal ideal [6].

A continuación, se revisan algunos de los problemas más comunes en este tipo de sensores ToF.

2.1.4.1. Tiempo de integración y magnitud de la señal IR

El tiempo de integración más largo generalmente mejora la precisión de la medición de profundidad, pero también limita la velocidad de fotogramas. Además, la variación en la cantidad de la señal IR reflejada introduce ruido en el cálculo de profundidad.

En la Figura 2.9 se presentan ejemplos de este ruido y error sistemáticos: a) Representa el error de tiempo de integración: un tiempo de integración más largo muestra una mayor precisión de la profundidad (derecha) que un tiempo de integración más corto (izquierda). b) Muestra el error de magnitud IR, los puntos 3D de la misma profundidad (tablero de ajedrez a la izquierda) muestran diferentes magnitudes IR (tablero de ajedrez a la derecha) según el color del objeto [6], [33].

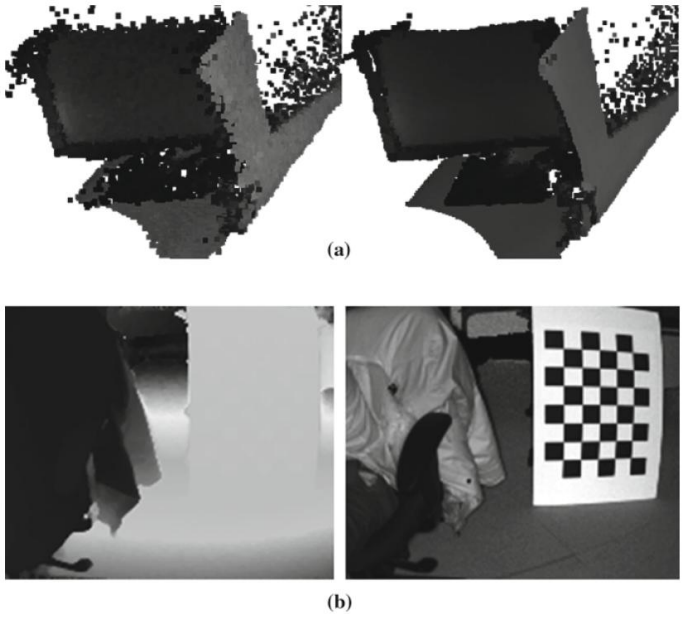

Figura 2.9. Comportamiento del tiempo de integración y la magnitud de la señal IR [6].

2.1.4.2. Material y geometría de los objetos

La precisión de las mediciones de profundidad varía significativamente según las propiedades del material y la geometría de los objetos.

En la Figura 2.10 se observa una tabla de evaluación basada en mapas de profundidad y sus imágenes de referencia obtenidas con un escáner 3D comercial (Kinect Depth).

En escenarios con objetos opacos y de superficies mayoritariamente difusas o lambertianas, como se observa en los casos A-5 y A-10, ambas tecnologías presentan una correlación aceptable con la verdad de campo (*Ground Truth*). No obstante, se observa que incluso en estos casos favorables, el sensor Kinect tiende a generar mapas de profundidad con una mayor suavización de bordes y una pérdida sutil de detalle en geometrías complejas en comparación con la referencia ideal [6].

La principal caída del rendimiento ocurre ante la presencia de superficies con altas propiedades especulares o metálicas, representadas en las filas B-3 a B-10. Para objetos con acabados reflectantes, tanto el sensor ToF como el Kinect presentan diferencias significativas; en el caso del ToF, se perciben distorsiones morfológicas y errores en la estimación de la distancia debido a la interferencia por trayectorias múltiples de la señal infrarroja. Por su parte, el sensor Kinect presenta zonas de oclusión o vacío de datos (áreas negras), donde la reflexión especular impide que el patrón

de luz regrese de forma coherente al receptor, imposibilitando la triangulación precisa de la profundidad.

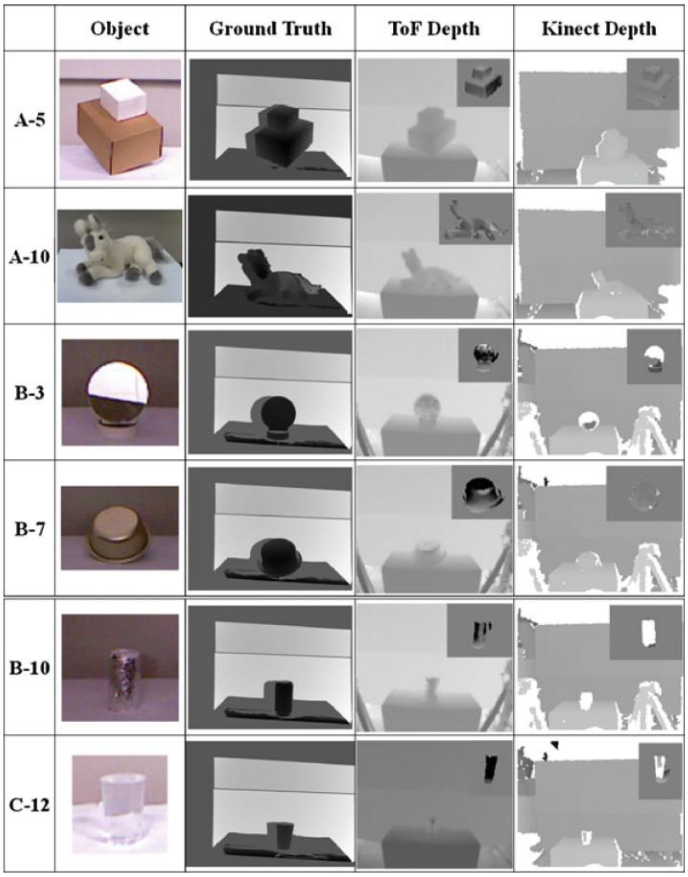

Figura 2.10. Imágenes de profundidad de muestra y conjunto de imágenes ToF de prueba [6].

Finalmente, el desafío más importante para estos sistemas de visión se presenta en la detección de objetos transparentes, como se ilustra en la muestra C-12. Los resultados demuestran una incapacidad casi total para reconstruir la geometría del objeto, ya que la mayor parte de la emisión lumínica atraviesa el material o se refracta de manera impredecible. Mientras que el *Ground Truth* logra definir la silueta del contenedor, los mapas de ToF y Kinect muestran una representación fragmentada o inexistente.

- En este contexto, el término ***Ground Truth*** se refiere a la información real y verificada que se utiliza como referencia para evaluar la precisión de modelos y sistemas, en este caso representa las dimensiones geométricas del objeto real.

- Una **superficie lambertiana** es un modelo físico ideal que describe una superficie perfectamente difusa. Su característica principal es que, al ser iluminada, refleja la energía lumínica de manera uniforme en todas las direcciones del hemisferio superior, lo que provoca que su brillo o radiancia sea constante para un observador, independientemente del ángulo desde el cual se mire.

- Las **propiedades especulares** definen la capacidad de una superficie para reflejar la luz en una sola dirección predominante, actuando de forma similar a un espejo. A diferencia de las superficies lambertianas (difusas), donde la luz se esparce en todas direcciones, en una superficie especular el rayo de luz rebota de manera ordenada y predecible.

2.1.4.3. Otras fuentes de ruido en sensores ToF

La distorsión de profundidad en los sensores ToF se refiere a cualquier discrepancia entre la distancia real de un objeto y la medida capturada por el sensor. Entre otros factores complejos, se encuentran: la interferencia de multipath y el desenfoque de movimiento [6], [33].

La interferencia de multipath ocurre cuando las múltiples reflexiones de luz causan errores significativos en el rango de medición. Mientras que el desenfoque de movimiento se presenta cuando la escena o la cámara se mueven durante la captura de la imagen de profundidad [6], [30].

La temperatura también desempeña un papel importante en el comportamiento de los sensores ToF, las variaciones internas y externas en este tipo de sensores generan inestabilidades en las mediciones hasta que se alcanza una temperatura estable. Además, el diseño del dispositivo de medición también introduce ruido relacionado con los píxeles (zonas), debido a variaciones en los materiales de los sensores, e incluso por la reflectancia causada por las cubiertas de protección, este ruido afecta la uniformidad de la profundidad entre píxeles adyacentes [6], [33].

Finalmente, la relación señal-ruido es esencial para obtener mediciones confiables, pues comparar el nivel de la señal con el ruido permite evaluar la calidad de los datos. Controlar y mitigar estos factores es un aspecto importante en el diseño y aplicación de sensores ToF [30].

2.1.4.4. Técnicas de mitigación del ruido

Para mejorar la precisión y reducir el ruido en sensores ToF, se han desarrollado diversas técnicas de mitigación que abordan los distintos tipos de ruido en profundidad. La temperatura, por ejemplo, es una fuente común de inestabilidad en las mediciones; algunas cámaras y sensores incluyen

sistemas de enfriamiento que ayudan a mantener una temperatura interna estable, reduciendo así la fluctuación de los datos de profundidad [5], [6], [33].

Para el ruido relacionado con los píxeles, se emplean matrices de Ruido de Patrón Fijo (FPN) y técnicas de filtrado que permiten suavizar las variaciones en las lecturas de los píxeles adyacentes, mejorando la uniformidad en las mediciones de profundidad. En cuanto a la dispersión de luz, se aplican modelos de compensación basados en datos crudos y análisis analíticos que ayudan a reducir los errores causados por la dispersión en la superficie de la escena [6].

La distorsión de profundidad puede ser mitigada promediando valores de medición o usando referencias de distancia. Para tratar los efectos de la interferencia de multipath y las no linealidades en la señal, se emplean técnicas avanzadas como la distribución de Cauchy, algoritmos de separación de señales y redes neuronales convolucionales. Estas herramientas son efectivas para distinguir la señal principal de las reflexiones no deseadas [6], [30].

Finalmente, el desenfoque de movimiento es un problema común en escenas dinámicas o cuando la cámara se mueve. Técnicas de exposición codificada y el uso de filtros de Kalman ayudan a estabilizar las mediciones y minimizar los efectos de movimiento, permitiendo obtener una estimación de la profundidad más precisa en condiciones complejas [6].

2.2. Aprendizaje Automático Integrado (TinyML)

A diferencia de lo que comúnmente cree la mayoría de las personas, la Inteligencia Artificial (IA) no fue un desarrollo que inicio hace pocos años, sus cimientos matemáticos se remontan a finales del siglo XVIII con trabajos como la regresión lineal de Gauss. No obstante, la evolución de la IA ha atravesado ciclos de expansión y retroceso, conocidos como inviernos de la IA, donde las expectativas tecnológicas superaron las capacidades reales de la época. Desde la acuñación del término en la conferencia de Dartmouth en 1956 y el desarrollo del perceptrón, hasta el auge de los sistemas expertos y las primeras redes neuronales convolucionales (CNN) en los años 80 para el reconocimiento de caracteres, la disciplina ha buscado emular procesos cognitivos con éxito variable [1].

El periodo actual, iniciado a principios de la década de 2010, es considerado un renacimiento tecnológico impulsado por la revolución del aprendizaje profundo (*Deep Learning*) [1]. Como se observa en la Figura 2.11, este auge no es accidental, sino el resultado de cuatro factores clave que han redefinido el panorama tecnológico [3], [34], [35].

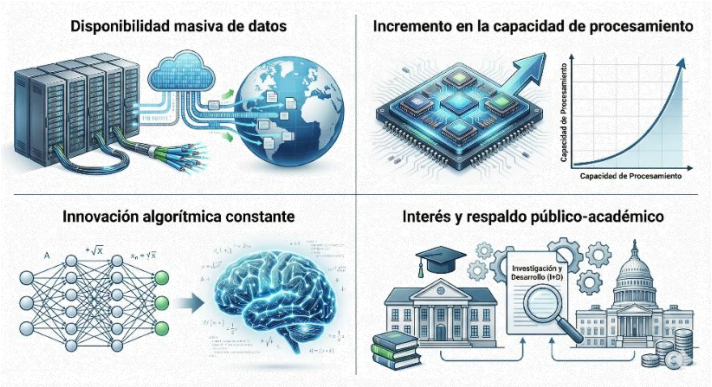

Figura 2.11. Factores clave que impulsaron el crecimiento de la IA.

1. Disponibilidad masiva de datos: La generación de información a escala global ha crecido de manera exponencial, proporcionando el combustible necesario para entrenar algoritmos complejos; se estima que la humanidad produce actualmente en dos días más datos que en toda su historia registrada hasta el año 2003.
2. Incremento en la capacidad de procesamiento: Siguiendo trayectorias como la Ley de Moore, la potencia de las unidades centrales de procesamiento (CPU) se ha duplicado

aproximadamente cada 18 meses, permitiendo el manejo de volúmenes masivos de información.

3. Innovación algorítmica constante: El desarrollo de nuevos métodos y capacidades en el espacio del aprendizaje automático ocurre con una frecuencia semanal, tanto en el ámbito de la investigación académica como en aplicaciones industriales.
4. Interés y respaldo público-académico: Existe una inversión sin precedentes en programas de investigación y desarrollo (I+D) por parte de gobiernos y universidades, consolidando la ciencia de datos y el aprendizaje automático como pilares educativos y económicos globales.

2.2.1. Fundamentos de aprendizaje automático en sistemas embebidos

El Aprendizaje Automático (*Machine Learning, ML*) representa un cambio de paradigma en la ingeniería de software [1]. Mientras que la programación tradicional se basa en algoritmos deterministas donde un desarrollador escribe reglas explícitas para procesar datos y obtener resultados, el ML invierte este flujo: el sistema procesa grandes volúmenes de datos y resultados conocidos para aprender o, mejor dicho, inferir las reglas matemáticas subyacentes que los relacionan. Dependiendo de la naturaleza de los datos y la forma en que el algoritmo interactúa con ellos, el ML se divide en tres principales categorías: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo [1], [35-37], véase la Figura 2.12.

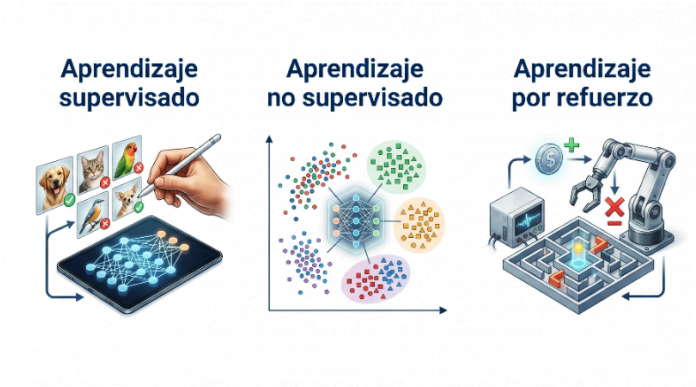

Figura 2.12. Principales categorías del aprendizaje automático (ML).

- El **aprendizaje supervisado** es el enfoque más utilizado en aplicaciones de visión artificial y TinyML. En este modelo, el algoritmo se entrena con un conjunto de datos previamente

etiquetados (entradas y salidas deseadas), permitiéndole aprender una función de mapeo para clasificar nueva información.

- A diferencia del anterior, el sistema de **aprendizaje no supervisado** trabaja con datos sin etiquetas y busca encontrar estructuras o agrupaciones intrínsecas (*clustering*) dentro de la información. Es útil para descubrir patrones ocultos, aunque su implementación en sistemas de tiempo real con recursos limitados suele ser más compleja.

- El **aprendizaje por refuerzo** se basa en un ciclo de retroalimentación donde un agente aprende a tomar decisiones mediante un sistema de recompensas y penalizaciones al interactuar con un entorno. Aunque es común en robótica avanzada, su alta demanda computacional suele superar las capacidades de los microcontroladores estándar de TinyML.

El concepto de TinyML surge como una rama especializada del aprendizaje automático que busca ejecutar estos modelos probabilísticos directamente en el borde, específicamente en microcontroladores [2], [36], [37]. Esta integración local es posible gracias a los facilitadores tecnológicos mencionados anteriormente, en especial la optimización de algoritmos y el incremento en la eficiencia de las arquitecturas de procesamiento como el núcleo Cortex-M7.

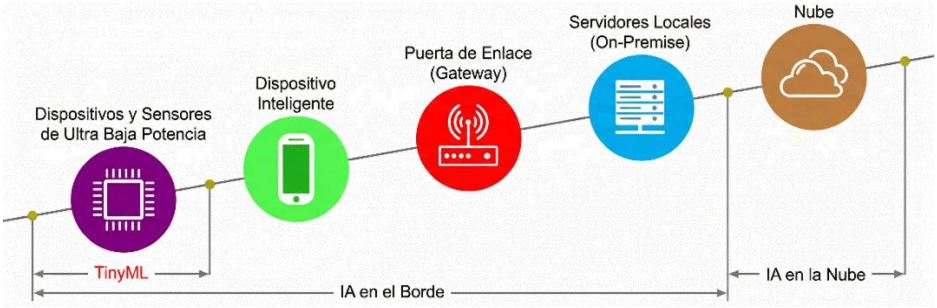

Figura 2.13. Espectro de inteligencia distribuida [1].

A diferencia del aprendizaje automático convencional ejecutado en servidores o computadoras personales, véase la Figura 2.13, el diseño de sistemas TinyML está condicionado por tres principales restricciones de hardware; 1) Los dispositivos operan con una memoria RAM limitada (frecuentemente en el rango de los kilobytes), lo que restringe el tamaño de las activaciones de la

red neuronal durante la inferencia. 2) La memoria FLASH disponible debe albergar tanto el código de control del sistema como los pesos del modelo entrenado, obligando al uso de técnicas de compresión. 3) Para garantizar la portabilidad y autonomía de un dispositivo de asistencia, el procesamiento de la IA debe realizarse con valores de potencia extremadamente bajos, idealmente inferiores a un milivoltio [1], [4].

2.2.2. Redes neuronales y arquitecturas convolucionales (CNN)

Las Redes Neuronales Artificiales (ANN, por sus siglas en inglés) constituyen la infraestructura fundamental del aprendizaje profundo. Estas estructuras se inspiran en la organización biológica del cerebro, disponiéndose en capas de neuronas interconectadas que procesan información mediante pesos sinápticos ajustables y funciones de activación no lineales [1], [34].

No obstante, el uso de redes densamente conectadas para procesar matrices de datos espaciales, como los mapas de profundidad de 50×50 puntos, resulta ineficiente en entornos de TinyML debido a la explosión en el número de parámetros, lo que sobrepasa rápidamente las capacidades de memoria de un microcontrolador [36].

Para abordar este desafío, se emplean las Redes Neuronales Convolucionales (CNN), las cuales están diseñadas específicamente para procesar datos con estructura de rejilla. La arquitectura de una CNN permite extraer características jerárquicas de la escena mediante dos propiedades que reducen drásticamente la carga computacional.

A diferencia de las redes densas, cada neurona en una capa convolucional solo se conecta a una pequeña región de la entrada, permitiendo identificar patrones locales como bordes o cambios bruscos de profundidad.

Se aplica el mismo conjunto de parámetros (filtro o kernel) en toda la matriz de entrada [37] (véase Figura 2.14). Esto no solo disminuye el uso de memoria RAM y FLASH, sino que otorga al sistema la capacidad de detectar una característica (por ejemplo, el borde de un escalón) sin importar su ubicación en el campo de visión del sensor ToF.

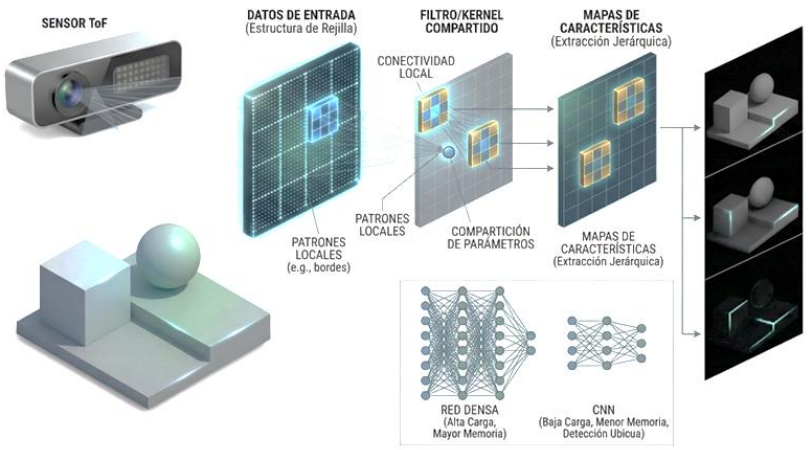

Figura 2.14. Red densa VS. CNN con TinyML y sensores ToF.

El funcionamiento interno de estas redes se basa principalmente en una secuencia de capas operativas que transforman los datos de entrada en representaciones abstractas de alto nivel:

En la Figura 2.15 se puede observar gráficamente un ejemplo de la estructura que se utiliza en una Red Neuronal Convolucional para TinyML.

2.2.2.1. Capa de convolución y filtros espaciales

Es el componente fundamental donde se realiza la extracción de características. En esta capa, un conjunto de filtros o kernels (matrices de pesos de tamaño reducido, por ejemplo 3×3) se desplaza a través de la matriz de profundidad del sensor [1], [32]. En cada posición, se calcula un producto escalar entre los pesos del filtro y los valores de los píxeles de la zona correspondiente.

Matemáticamente, para una entrada I y un núcleo K, el valor de un píxel en el mapa de características resultante (S) se define mediante la operación [1], [34]:

$$S(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \qquad \textit{Ecuación 3.}$$

Esta operación permite resaltar patrones específicos como bordes, esquinas o cambios de gradiente en la distancia, los cuales son almacenados en mapas de características (*feature maps*).

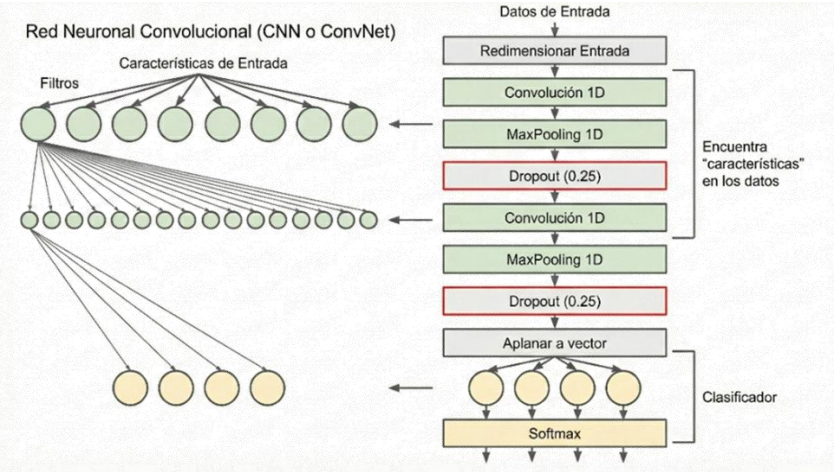

Figura 2.15. Estructura general de una Red Neuronal Convolucional.

2.2.2.2. Función de activación (ReLU)

Tras la convolución, se aplica una función de activación no lineal para permitir que la red aprenda relaciones complejas. En arquitecturas TinyML, la función más utilizada es la Unidad Lineal Rectificada (ReLU), debido a su baja demanda computacional. Su función es descartar los valores negativos resultantes de la convolución (sustituyéndolos por cero) y mantener los positivos, lo que ayuda a la red a concentrarse únicamente en las características relevantes detectadas [1], [34], [36].

2.2.2.3. Capa de agrupamiento (*Pooling)*

Para reducir la sensibilidad del modelo a pequeñas variaciones en la posición de los objetos y aligerar la carga de memoria, se emplean capas de *Pooling*. La técnica más común es el Max-Pooling, que selecciona el valor máximo dentro de una ventana específica (por ejemplo, de 2×2 píxeles).

Esta operación reduce la resolución espacial del mapa de características, conservando las señales de mayor intensidad y disminuyendo significativamente el número de parámetros que el microcontrolador debe procesar en las etapas siguientes [1], [3].

2.2.2.4. Capas totalmente conectadas

Al final de la arquitectura, los mapas de características tridimensionales se aplanan en un vector unidimensional. Este vector alimenta a una o más capas densas donde cada neurona está conectada

con todas las de la capa anterior. Aquí es donde se realiza la clasificación final, asignando una probabilidad a cada categoría de obstáculo definida [1], [3].

2.2.3. Ciclo de vida de los datos: recolección y aumento

En el desarrollo de sistemas basados en aprendizaje automático, la calidad del modelo resultante está intrínsecamente ligada a la representatividad y limpieza de los datos utilizados durante su entrenamiento. Este principio, a menudo resumido en la premisa; basura entra, basura sale, es especialmente importante en el aprendizaje supervisado, donde el algoritmo intenta encontrar una función que mapee las entradas con sus etiquetas correspondientes [1], [35-37].

2.2.3.1. Recolección de datos y representatividad

La fase de recolección consiste en la adquisición de muestras que capturen la variabilidad real del entorno donde operará el sistema [1], [35]. Para que un modelo sea robusto, el *dataset* debe cumplir con criterios de:

- **Diversidad**: Incluir una amplia gama de escenarios, ángulos y distancias.
- **Balance de clases**: Asegurar que el número de muestras para cada categoría sea equitativo, evitando que el modelo desarrolle un sesgo hacia la clase mayoritaria.
- **Etiquetado**: El proceso de asignar una verdad fundamental (*ground truth*) a cada muestra.

2.2.3.2. Aumento de Datos (*Data Augmentation*)

En proyectos de ingeniería donde la recolección manual de datos es costosa o el *dataset* inicial es limitado, se recurre al aumento de datos. Esta técnica consiste en generar nuevas muestras de entrenamiento de forma sintética a partir de las existentes, aplicando transformaciones matemáticas que mantienen la validez de la etiqueta original [1].

Las técnicas de aumento más relevantes para datos espaciales y mapas de profundidad incluyen:

- **Transformaciones geométricas**: Rotaciones leves, traslaciones y espejado horizontal (*flipping*). Estas operaciones ayudan a que el modelo sea invariante a la posición o al ángulo en el que el usuario porta el dispositivo.

- **Inyección de ruido**: Agregar variaciones aleatorias (ruido gaussiano) a los valores de profundidad para simular las imperfecciones de lectura propias de los sensores ToF en condiciones adversas.
- **Escalamiento y recorte**: Permite que la red neuronal aprenda a reconocer obstáculos a diferentes escalas, mejorando la detección tanto a corta como a larga distancia.

El uso de estas estrategias permite prevenir el sobreajuste (*overfitting*), un fenómeno donde el modelo memoriza las muestras de entrenamiento, pero pierde la capacidad de generalizar ante datos nuevos [35].

2.2.4. Evaluación y optimización del modelo

Una vez que el modelo ha sido entrenado, es necesario realizar una evaluación detallada de su desempeño antes de proceder a su despliegue en el microcontrolador. En sistemas de asistencia para personas con discapacidad visual, la confiabilidad es muy importante, ya que un error de clasificación puede comprometer la seguridad física del usuario.

2.2.4.1. Matriz de confusión y métricas de clasificación

La herramienta fundamental para el análisis de desempeño es la matriz de confusión. Esta tabla permite visualizar el desempeño del algoritmo comparando las predicciones del modelo contra el *ground truth* del *dataset*. De esta matriz se derivan cuatro valores base: Verdaderos Positivos (VP), Verdaderos Negativos (VN), Falsos Positivos (FP) y Falsos Negativos (FN) [1], [34], [36], [38].

A partir de estos valores, se calculan las métricas estadísticas que permiten cuantificar la eficacia del sistema:

Exactitud (*Accuracy*): Representa el porcentaje total de predicciones correctas frente al total de casos evaluados.

$$Exactitud = \frac{VP+VN}{VP+VN+FP+FN} \qquad \textit{Ecuación 4.}$$

Precisión o Valor Predictivo Positivo (PPV): Indica la capacidad del modelo para no clasificar como positiva una muestra que es negativa. Es fundamental para reducir falsas alarmas.

Sensibilidad o Tasa de Verdaderos Positivos (TPR): También conocida como *Recall*, mide la

$$Precisión\ (PPV) = \frac{VP}{VP+FP} \qquad \textit{Ecuación 5.}$$

$$Sensibilidad\ (TPR) = \frac{VP}{VP+FN} \qquad \textit{Ecuación 6.}$$

proporción de casos positivos reales que fueron identificados correctamente.

Especificidad o Tasa de Verdaderos Negativos (TNR): Mide la capacidad del sistema para identificar correctamente las muestras negativas.

Tasa de Falsos Positivos (FPR): Representa la probabilidad de que el sistema detecte un obstáculo inexistente.

$$Especificidad\ (TNR) = \frac{VN}{VN+FP} \qquad \textit{Ecuación 7.}$$

$$FPR = \frac{FP}{VN+FP} \qquad \textit{Ecuación 8.}$$

$$FNR = \frac{FN}{VP+FN} \qquad \textit{Ecuación 9.}$$

Tasa de Falsos Negativos (FNR): Ocurre cuando el sistema no detecta un peligro real.

F1-Score: Es la media armónica entre la precisión y la sensibilidad. Proporciona un equilibrio entre ambas métricas, siendo especialmente útil cuando las clases en el *dataset* están desbalanceadas.

$$\text{Puntuación F1} = 2 \times \frac{Precisión \times Sensibilidad}{Precisión + Sensibilidad} \qquad \textit{Ecuación 10.}$$

2.2.4.2. Optimización para TinyML: Cuantización

Debido a que el entrenamiento suele realizarse en computadoras con procesadores de alto rendimiento utilizando precisión de punto flotante de 32 bits (float32), el modelo resultante suele ser demasiado pesado para un microcontrolador. Para solventar esto, se aplica la cuantización.

Este proceso consiste en reducir la precisión de los pesos y las activaciones de la red neuronal, generalmente convirtiéndolos a enteros de 8 bits (int8) [1], [36], [38].

La cuantización permite:

- Reducir el tamaño del modelo disminuye el uso de memoria FLASH y RAM en aproximadamente un 75%.
- Acelerar la inferencia, ya que los microcontroladores procesan operaciones con enteros de manera mucho más eficiente y rápida que con punto flotante.

Capítulo 3

3. Desarrollo e implementación del sistema

El presente capítulo describe el desarrollo del sistema de asistencia, abarcando desde la selección de la arquitectura de hardware hasta la implementación final del firmware de control. El objetivo central de esta fase es traducir los fundamentos teóricos en un prototipo funcional capaz de operar en tiempo real bajo las restricciones de un sistema empotrado (TinyML).

El desarrollo se estructura en tres etapas. En primer lugar, se detalla la integración electrónica del sistema, fundamentada en el uso de la tarjeta de desarrollo Arduino Nicla Vision y el sensor de tiempo de vuelo MaixSense A010, definiendo los protocolos de comunicación necesarios para la adquisición de mapas de profundidad.

Posteriormente, se aborda el diseño y entrenamiento del modelo de red neuronal convolucional. Se pone especial énfasis en las métricas de evaluación y en las estrategias de optimización, comparando el desempeño de modelos de precisión simple (Float32) frente a versiones cuantizadas (Int8), justificando la selección del compilador EON para maximizar la eficiencia de los recursos de memoria y tiempo de inferencia.

Finalmente, se presenta la arquitectura del firmware, diseñada bajo un esquema de procesamiento asimétrico de doble núcleo (STM32H7). Se explica cómo se distribuyen las cargas de trabajo entre el núcleo maestro (Cortex-M7), dedicado a la inteligencia artificial, y el coprocesador (Cortex-M4), encargado de la seguridad determinista y el control de actuadores.

3.1. Caracterización y análisis de los componentes del sistema

En lugar de seleccionar los componentes basándose únicamente en las especificaciones teóricas de las hojas de datos, se desarrollaron prototipos y pruebas preliminares para caracterizar el desempeño real de los dispositivos en condiciones controladas. Este proceso permitió evaluar métricas importantes como la linealidad de la medición, el alcance mínimo y máximo de los sensores y la facilidad de su integración junto a un sistema embebido, Así mismo, se evaluaron las capacidades de las unidades de procesamiento MCUs para ejecutar algoritmos de visión artificial y modelos de aprendizaje automático integrado.

A continuación, se detalla el análisis comparativo y las pruebas experimentales que condujeron a la selección del módulo MaixSense A010 y la plataforma de desarrollo Nicla Vision como el núcleo del sistema, descartando otras alternativas tras un análisis detallado de su rendimiento.

3.1.1. Análisis comparativo de los sensores ToF

Para determinar el sensor más adecuado para el sistema de asistencia propuesto, se evaluaron diversas soluciones comerciales de tecnología ToF. Los criterios de selección se centraron en cuatro ejes fundamentales: resolución espacial (capacidad para definir formas), interfaz de comunicación (facilidad de integración con microcontroladores), rango de operación y factor de forma.

La Tabla 3-1 resume las especificaciones técnicas de los dispositivos analizados, contrastando sensores de grado industrial, módulos de desarrollo y cámaras de profundidad completas.

Tras el análisis de las especificaciones, se descartaron opciones como la RealSense R200 y la Arducam ToF debido a su alta demanda computacional (requieren interfaces USB 3.0 o MIPI CSI-2) y su elevado consumo de energía, lo cual es incompatible con la autonomía requerida el sistema que se propone en este trabajo.

La decisión final se debatió principalmente entre el VL53L8CH de STMicroelectronics y el MaixSense A010 de Sipeed.

Aunque el sensor VL53L8CH ofrece una elevada precisión milimétrica y herramientas avanzadas de IA, su resolución máxima es de 8×8 zonas (64 puntos), que limita su capacidad de identificar la geometría detallada de objetos complejos sin aplicar técnicas avanzadas de interpolación.

El módulo MaixSense A010 fue seleccionado como la opción óptima para el proyecto. A diferencia del VL53L8CH, el MaixSense A010 integra un microcontrolador dedicado que gestiona la adquisición óptica y entrega un mapa de profundidad de 100×100 puntos ya procesados.

Su alta densidad de puntos proporciona una imagen de profundidad mucho más detallada para alimentar a la Red Neuronal Convolucional (CNN), mejorando la clasificación de obstáculos. Su comunicación vía UART simplifica la integración tanto en etapas de prototipado rápido (MicroPython) como en el despliegue final (C++), permitiendo una lectura de datos eficiente mediante tramas seriales simples.

Tabla 3-1. Comparativa de sensores de tiempo de vuelo evaluados.

Característica	Arducam ToF Camera	VL53L1CX	VL53L5CX	VL53L8CH	Realsense R200	MaixSense-A010
Resolución	240×180 píxeles efectivos	No aplica (sensor de distancia puntual)	Matriz de 8×8 zonas (64 zonas)	Hasta 8×8 zonas (64 zonas)	Cámara de profundidad: 640×480 Cámara de color: 1920×1080	100×100 píxeles
Rango de medición	Modo cercano: hasta 2 m Modo lejano: hasta 4 m	Hasta 4 m	Hasta 4 m	2 cm a 4 m por zona	Interior: hasta 3.5 m Exterior: hasta 10 m	0.2 m a 2.5 m
Campo de visión (FoV)	Diagonal de 70°	27° típico	65° diagonal	65° diagonal (45° x 45°)	No especificado	Horizontal: 70° Vertical: 60°
Frecuencia de cuadro	Sensor: hasta 120 fps Raspberry Pi: hasta 30 fps	Hasta 50 Hz	Hasta 60 Hz	Hasta 30 Hz	30 fps	Hasta 30 fps
Interfaz	MIPI 2-Lane	I²C	I²C	I²C o SPI	USB 3.0	USB Tipo-C, UART
Fuente de luz	VCSEL de 940 nm	Láser invisible de 940 nm (Clase 1)	VCSEL de 940 nm	VCSEL de 940 nm	Láser de clase 1	VCSEL de 940 nm
Dimensiones	38 x 38 mm	4.9 x 2.5 x 1.56 mm	6.4 x 3.0 x 1.5 mm	6.4 x 3.0 x 1.75 mm	20 x 130 x 7 mm	23.25 x 40.70 x 10.50 mm
Características adicionales	Compatible con Raspberry Pi y Nvidia Jetson; salida de nube de puntos	Programación de región de interés (ROI); bajo consumo	Detección de múltiples objetos; inmunidad al crosstalk	Salida de histograma compacta y normalizada; indicador de movimiento por zona	Cámara RGB y de profundidad; adecuada para aplicaciones de realidad aumentada	Pantalla LCD de 1.14” Integración con ROS

3.1.2. Selección de la unidad de procesamiento (Nicla Vision)

La unidad de procesamiento central actúa como el cerebro del sistema, encargada de tres tareas importantes: la adquisición de datos de alta velocidad provenientes del sensor ToF vía UART, la ejecución de la inferencia del modelo de red neuronal (CNN) y la gestión de la respuesta al usuario. Dado que el objetivo es desarrollar un dispositivo portátil, la elección del microcontrolador estuvo condicionada por la necesidad de maximizar la potencia de cómputo minimizando el tamaño físico y el consumo energético.

Se evaluaron diversas plataformas de desarrollo populares en el ámbito del IoT y la visión embebida, incluyendo el ESP32-S3 y la Raspberry Pi Pico. Sin embargo, se seleccionó la placa Arduino Nicla Vision como la mejor opción para implementar este proyecto debido a que cumple con las características que se definieron para este proyecto.

A diferencia de microcontroladores estándar de un solo núcleo, la Nicla Vision integra el procesador STM32H747XI con una arquitectura de procesamiento dual-core, el cual combina dos núcleos independientes:

- **Arm Cortex-M7 (480 MHz):** Un núcleo de alto rendimiento diseñado para tareas intensivas. En este proyecto, se dedica exclusivamente a la ejecución del modelo TinyML y el procesamiento de señales digitales.
- **Arm Cortex-M4 (240 MHz):** Un núcleo de menor consumo y alta eficiencia, ideal para gestionar la comunicación UART con el sensor, el control de periféricos y la lógica de supervisión del sistema.

Esta arquitectura en paralelo permite optimizar la adquisición de datos de la inferencia, evitando cuellos de botella que podrían generar latencia en la detección de obstáculos.

Otro factor importante que se considero fue su tamaño compacto. Con dimensiones de apenas $22.86\ \text{mm} \times 22.86\ \text{mm}$, la Nicla Vision es una de las placas de desarrollo industrial más pequeñas del mercado. Su tamaño reducido facilita la integración en un prototipo portátil como el que se busca en este trabajo, sin sacrificar la potencia necesaria para ejecutar algoritmos de visión artificial.

En el apartado de alimentación del sistema, la tarjeta incluye un circuito de gestión de energía (PMIC) y la capacidad para cargar baterías Li-Po directamente, lo que simplifica el diseño electrónico del sistema portátil al eliminar la necesidad de módulos de carga externos.

Finalmente, esta tarjeta ofrece soporte para los principales ecosistemas utilizados en la programación de microcontroladores y modelos de aprendizaje integrado.

- **MicroPython (OpenMV IDE):** Gracias al intérprete de MicroPython, este entorno abstrae la complejidad de los registros del hardware, ofreciendo bibliotecas de alto nivel optimizadas para visión por computadora. Esto puede facilitar algunas aplicaciones, como la visualización de los mapas de profundidad del sensor y la gestión de archivos para la creación del dataset, permitiendo validar la lógica de detección de obstáculos de una manera más fácil (en comparación a los lenguajes compilados), sin la necesidad de realizar una compilación del código.
- **C/C++ (Arduino IDE y STM32Cube IDE):** Este lenguaje compilado ofrece un mayor control sobre el hardware, permitiendo la gestión estática de la memoria y el acceso directo a las instrucciones DSP del procesador. El uso de C/C++ es indispensable para implementar la comunicación entre núcleos (RPC), optimizar los tiempos de inferencia del modelo cuantizado y asegurar una ejecución en tiempo real.

Nota: Es importante destacar que ambos lenguajes de programación y sus respectivos entornos de desarrollo fueron implementados en las distintas fases de este proyecto. Tal como se describió en sus características, cada lenguaje tiene puntos fuertes dependiendo de la aplicación. Por tanto, no se trata de definir qué lenguaje es mejor o peor en términos absolutos, sino de tener la capacidad de elegir la herramienta adecuada para optimizar el trabajo en cada etapa del desarrollo.

En la Tabla 3-2 se observan las cinco tarjetas con arquitectura ARM, así como sus principales características que se analizaron en esta primera etapa del trabajo.

Considerando los criterios de diseño del sistema y las características que presentan las tarjetas de la Tabla 3-2, se eligió la Nicla Vision ya que no sólo provee la potencia de cómputo necesaria para TinyML, sino que su flexibilidad de software permite adaptar las herramientas de desarrollo a las necesidades específicas de cada etapa de la investigación: desde la caracterización de sensor, la adquisición y construcción del dataset, hasta la implementación del firmware final.

Tabla 3-2. Tabla comparativa de las tarjetas de desarrollo propuestas para el sistema.

Característica	Nicla Vision	Nucleo-H755ZI-Q	OpenMV Cam H7 Plus	Discovery kit STM32F746NG	Nucleo-F401RE
MCU (Arquitectura Arm)	STM32H747AII6 Dual-Core Cortex-M7 (480 MHz) Cortex-M4 (240 MHz)	STM32H755ZIT6 Dual-Core Cortex-M7 (480 MHz) Cortex-M4 (240 MHz)	STM32H743II Cortex-M7 (480 MHz)	STM32F746NGH6 Cortex-M7 (216 MHz)	STM32F401RET6 Cortex-M4 (84 MHz)
Memoria Flash	2 MB interna + 16 MB QSPI	2 MB	2 MB interna + 32 MB externa	1 MB	512 KB
RAM	1 MB	864 KB SRAM + 1 MB RAM	1 MB SRAM + 32 MB SDRAM	340 KB	96 KB
Cámara	2 MP (GC2145)	No incluida	5 MP (OV5640)	Conector para cámara	No incluida
Conectividad	Wi-Fi y Bluetooth Low Energy (BLE)	Ethernet	No incluida	Ethernet	No incluida
Sensores adicionales	Micrófono omnidireccional, sensor de movimiento de 6 ejes (IMU), sensor de distancia	No incluidos	No incluidos	Múltiples sensores integrados, incluyendo audio y táctiles	No incluidos
Pantalla	No incluida	No incluida	No incluida	LCD-TFT de 4.3" (480x272) con pantalla táctil capacitiva	No incluida
Conectores de expansión	Compatibilidad con productos Arduino MKR y Portenta	Conectores ST Zio y ST Morpho; compatibilidad con Arduino Uno V3	Compatible con la distribución OpenMV	Conectores para microSD, USB OTG, Ethernet y más	Conectores Arduino Uno V3 y ST Morpho
Alimentación	Puede ser alimentada por batería	Opciones flexibles: USB VBUS, conector USB o fuentes externas	Puede ser alimentada por batería	Varias opciones: ST-LINK/V2-1, conectores USB, VIN desde conector Arduino, fuente externa de 5 V	Opciones flexibles: ST-LINK USB VBUS o fuentes externas
Imagen					

3.1.3. Pruebas experimentales de los sensores ToF

La selección del sensor de distancia no se limitó a una comparación superficial de las hojas de datos, sino que involucró un proceso iterativo de investigación. Inicialmente, se trabajó con sensores de la familia STMicroelectronics debido a su amplia documentación y recursos de consulta, lo que permitió comprender a fondo el comportamiento físico de la tecnología de Tiempo de Vuelo. Posteriormente, este conocimiento sirvió como base para validar e integrar el módulo MaixSense A010, el cual, a pesar de carecer de información técnica detallada, ofrecía la resolución espacial necesaria para los objetivos de este trabajo.

Nota: La tarjeta Nucleo-F401RE se emplea únicamente para la caracterización y evaluación del sensor ToF, gracias a su configuración de pines, requiere menos tiempo implementar aplicaciones básicas en esta tarjeta. Aunque se menciona en la Tabla 3-2, desde el inicio no se contempló para ser utilizada en la implementación del sistema final, ya que el microcontrolador que integra (que tiene un buen rendimiento en muchos casos) restringe su uso en aplicaciones más avanzadas, como el sistema que se pretende desarrollar en esta tesis.

3.1.3.1. Estudio paramétrico de la tecnología ToF (VL53L8CH)

En la primera parte del desarrollo, se caracterizó el sensor VL53L8CH. Este dispositivo permitió acceder a parámetros de configuración avanzados que influyen en la calidad de la medición ToF. En la Figura 3.1 se observan las conexiones físicas entre la placa SATEL-VL53L8 y una tarjeta NUCLEO-F401RE. La placa SATEL dispone de dos modos de comunicación; I2C y SPI. Para esta prueba se optó por una configuración SPI.

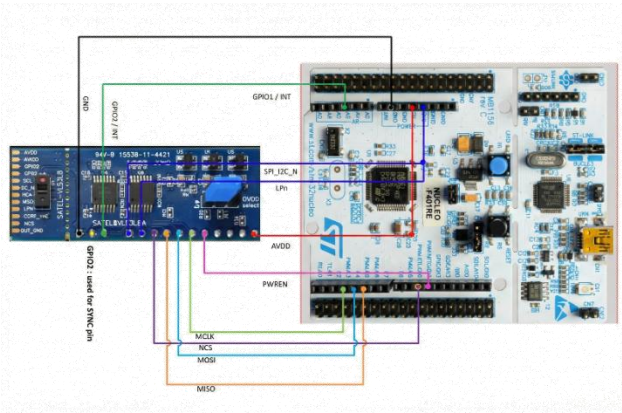

Figura 3.1. Conexión física del módulo SATEL-VL53L8 y la NUCLEO-F401RE mediante comunicación SPI.

El hardware se montó en un pequeño trípode, que, a su vez, se colocó sobre un mini elevador. De esta forma era posible manipular la posición y distancia del sistema sin afectar la estabilidad y conexiones de los componentes (véase Figura 3.2). Adicionalmente, se agregó un puntero laser, con el objetivo de obtener una mejor calibración del centro del campo de visión del sensor ToF, lo que aportaba mejores condiciones de repetibilidad a la prueba.

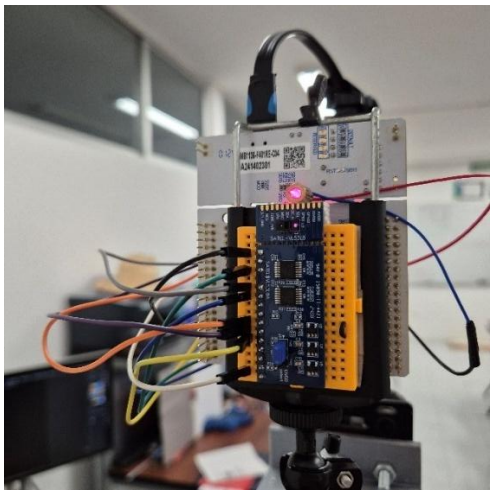

Figura 3.2. Montaje final para caracterización y pruebas del sensor VL53L8CH.

El sensor tiene un VCSEL que emite luz infrarroja a 940 nm en forma cónica, por otro lado, el receptor SPAD tiene una superficie cuadrada. Esta diferencia geométrica hace que las zonas cercanas a la esquina del SPAD reciban menor intensidad de la luz infrarroja emitida, lo que ocasiona mediciones con un error mayor que en las zonas centrales (véase Figura 3.3).

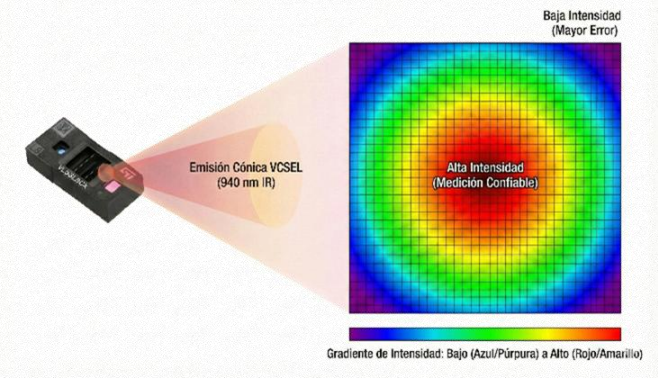

Figura 3.3. Desajuste entre emisión cónica VCSEL y matriz SPAD cuadrada.

Tomando en cuenta este comportamiento en el sensor ToF, las mediciones para esta prueba se dividieron en dos zonas: internas y esquinas.

Descripción general de la prueba – Configuración óptima del dispositivo

Con el fin de buscar la configuración más adecuada para el sensor ToF, se realizaron mediciones variando algunos parámetros en la configuración del sensor. Para tener un punto de referencia, se medió con una cinta métrica la distancia de 2 metros, ubicando el sensor en un extremo y utilizando un fondo blanco (pared del laboratorio) en el otro. De esta forma se asegura que la superficie donde rebotará la señal infrarroja será la misma para todas las zonas del SPAD.

Las configuraciones constantes en el sensor fueron:

- Resolución: 8x8 zonas
- Frecuencia de muestreo: 15 Hz
- Distancia de medición: 2000 milímetros → 2 metros

Considerando los tres parámetros contantes para esta prueba, en la Tabla A-1 se agregan las configuraciones disponibles para el VL53L8CH.

Nota: Para entender de mejor forma la Tabla A-1 y la Tabla A-2, se recomienda revisar la definición de los siguientes términos: factor de agrupamiento, tamaño del contenedor y ventana del contenedor.

- **Factor de agrupamiento (*binning factor*).** Es el número de bins contiguos que se suman para formar un bin nuevo. Al agrupar 2, 4 u 8 bins se profundiza el histograma (se gana alcance) reduciendo la resolución fina, pues cada valor resultante representa un tramo más largo.

- **Tamaño del contenedor (*bin size*).** Indica la distancia física que cubre cada bin. En el sensor sin agrupar un bin = 37.5 mm, al aplicar un factor de agrupamiento se multiplica (por ejemplo. con factor 2 el bin = 75 mm, con factor 8 el bin =300 mm), de modo que un solo bin obtiene retornos sobre un tramo mayor.

- **Ventana del contenedor (*bin window*).** Es la profundidad total que abarcan los bins seleccionados: tamaño del contenedor × número de bins. Así, 18 bins con un factor de agrupamiento de 1 cubren ≈ 675 mm, mientras que 18 bins con factor 3 (112.5 mm cada uno) abarcan ≈ 1.9 m, permitiendo medir objetos más lejanos a costa de perder un poco de detalle.

En la Tabla A-1 se destacan en color verde las opciones que teóricamente se consideran más adecuadas, sobre todo en función de la distancia máxima de alcance, ya que para este proyecto se busca detectar objetos dentro de un rango aproximado de 2 metros, se tomó como referencia esta distancia.

De esas nueve configuraciones seleccionadas, se tomaron cinco muestras para cada una. Los valores de interés fueron: error promedio, error mínimo y error máximo (véase la Tabla A-2). Con base en estos resultados, se hace una comparación gráfica de los valores obtenidos (sección 4.1).

3.1.3.2. ¿Por qué cambiar el sensor VL53L8CH por el módulo MaixSense A010?

Si bien al inicio del proyecto este sensor era la mejor opción disponible, fue necesario revisar otras alternativas que solventaran las limitaciones que se habían presentado, sobre todo la baja resolución multizona. Fue entonces que se optó por explorar un nuevo dispositivo; el módulo ToF MaixSense-A010.

El principal problema que se encontró fue el limitado acceso a la información del módulo y sensor ToF, ya que sólo se mencionan características generales, pero no es posible acceder a recursos como el código de fuente del módulo, los esquemas eléctricos de la tarjeta, las hojas de datos y las bibliotecas para modificar los parámetros del sensor ToF. Sin embargo, cómo se mencionó en la sección 3.1.3.1, el conocimiento adquirido con el sensor VL53L8CH fue de gran ayuda para entender el principio de funcionamiento de este módulo. Lo que permitió hacer el cambio del sensor VL53L8CH por el módulo MaixSense-A010, sin volver a iniciar desde cero.

3.1.3.3. Evaluación del módulo MaixSense A-010

El MaixSense-A010 es un módulo sensor 3D de bajo costo desarrollado por Sipeed, que combina un sensor de tiempo de vuelo (ToF) OPN8001 de 100x100 píxeles con el microcontrolador BL702 de 32 bits basado en RISC-V a 144 MHz.

Este dispositivo ofrece una resolución de 8 bits y un rango de medición de 0.2 a 2.5 metros, haciendo posible detectar objetos con una precisión de hasta 1 cm de profundidad. Además, cuenta con compatibilidad de interfaces USB tipo C y comunicación UART para implementarse junto a microcontroladores.

Estructura del paquete para la lectura de la imagen de profundidad:

Los datos de imagen son encapsulados en paquetes como se muestra en la Figura 3.4:

- Encabezado 2 bytes: 0x00, 0xFF
- Longitud del paquete 2 bytes: el número de bytes de datos restantes en el paquete actual
- Otro contenido de 16 bytes: incluido el número de serie del paquete, la longitud del paquete, la resolución, etc.
- Marco de imagen: 1 byte por cada pixel de la imagen (para 100x100 pixeles = 10,000 bytes)
- Comprobación de 1 byte: *Checksum*
- 1 byte al final del paquete: 0xDD

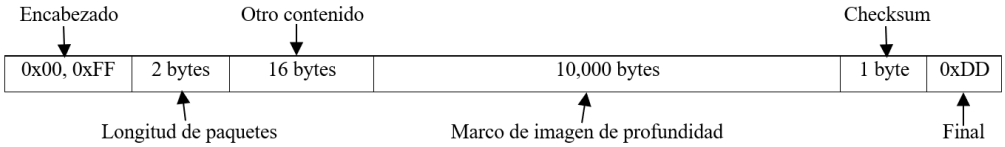

Figura 3.4. Estructura del paquete para lectura de la imagen de profundidad.

El sensor de tiempo de vuelo OPN8001 entrega valores de 16 bits para cada píxel, sin embargo, para reducir la carga computacional en el procesamiento de las imágenes ToF, estos valores se cuantifican a 8 bits. Esto permite disminuir el tamaño de los datos, mejorar los tiempos de transferencia y optimizar el procesamiento de las imágenes de profundidad.

Descripción general de la prueba – identificación del error

Similar a la prueba realizada con el sensor VL53L8CH, se determinó un punto de referencia, desde donde se medió con una cinta métrica la distancia de 2.5 metros (distancia máxima del módulo ToF), ubicando el sensor en un extremo y utilizando un fondo blanco (pared del laboratorio) en el otro.

En esta prueba, los parámetros en el módulo se conservaron constantes, variando únicamente la distancia al objetivo. Mediante comandos AT, se configuro el módulo a 19 fps, con una resolución de 100×100 pixeles. En la Figura 3.5 se muestra la estructura de pruebas que se utilizó para realizar esta caracterización.

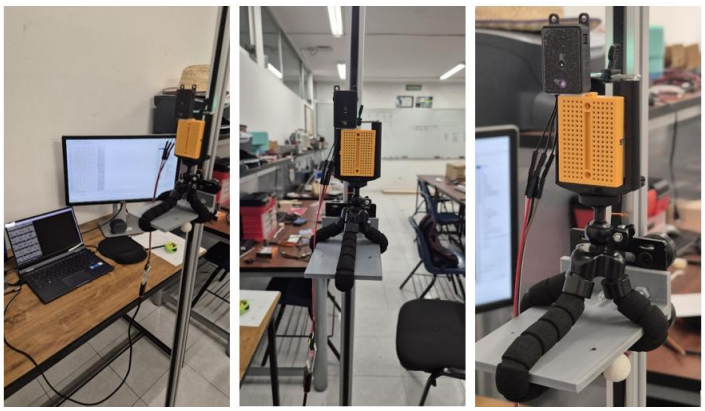

Figura 3.5. Hardware utilizado para pruebas con el módulo MaixSense A010.

El hardware se montó sobre un pequeño trípode flexible, que, a su vez, se colocó sobre un elevador que permitía ajustar la altura del módulo. Para realizar la configuración del dispositivo y capturar los datos del sensor, se utilizó el software Tera Term. Este software permite abrir una terminal serial para interactuar con el módulo. Por defecto, el módulo está configurado a 115 200 baudios, por lo que, al inicio la comunicación se deberá configurar a esa frecuencia.

Una vez guardados los datos, se pueden analizar de diferentes formas. En este trabajo, se utilizó el software HxD. Que permite leer el paquete de datos enviado y revisar de forma manual la estructura y valores que contiene (véase la Figura 3.6).

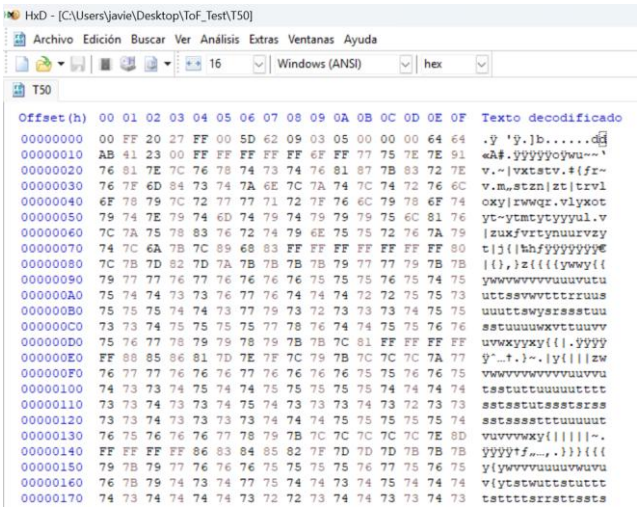

Figura 3.6. Análisis del paquete de imagen ToF del buffer de transmisión UART.

Para esta prueba, se comenzó con una distancia de 10 cm, con cambios también de 10 cm en cada nueva medición,

3.2. Recolección y limpieza de datos (*Dataset*)

La eficacia de un modelo de TinyML no reside únicamente en la arquitectura de la red neuronal, sino fundamentalmente en la calidad y representatividad de los datos con los que es entrenado. Para garantizar la construcción de un dataset robusto, fue necesario desarrollar una herramienta de hardware dedicada capaz de capturar la variabilidad del entorno real sin introducir artefactos o pérdidas de información.

3.2.1. Implementación del prototipo de recolección de datos

Para la fase de construcción del dataset, se requirió desarrollar un prototipo de adquisición de datos que fuese portable y facilitara la captura controlada de imágenes ToF. Si bien la tarjeta Nicla Vision fue seleccionada como la unidad de procesamiento para el sistema final, para esta etapa específica se optó por utilizar la tarjeta de desarrollo OpenMV Cam H7 Plus.

> Es importante destacar que, si bien la literatura sugiere realizar la recolección de datos con el hardware final de despliegue para evitar discrepancias, en este caso la sustitución de la tarjeta no afecta la integridad de la información. Ya que el componente fundamental que define la naturaleza de los datos es el sensor de adquisición (la fuente de la imagen de profundidad). Dado que tanto en este prototipo de recolección como en el sistema final se utiliza exactamente el mismo módulo MaixSense A010, la geometría y las características de las matrices capturadas son idénticas.

El prototipo que se propone desarrollar contempla los siguientes elementos:

- Tarjeta OpenMV Cam H7 Plus
- Modulo ToF MaixSense A010
- Batería de litio de 3.7 V y 1000 mAh
- Convertidor boost dc-dc
- Pantalla lcd tft de 1.8 pulgadas

La elección de la OpenMV Cam H7 Plus respondió a razones prácticas de ingeniería: cuenta con un puerto para tarjeta MicroSD (simplificando el almacenamiento del dataset) y compatibilidad directa con *shields* LCD, lo que permitió implementar una previsualización de la imagen ToF.

El prototipo se diseñó bajo el concepto de una cámara de profundidad portátil. Para alojar los componentes electrónicos y asegurar la estabilidad del sensor durante la captura, se diseñó y fabricó una carcasa personalizada mediante impresión 3D. El diseño integra rejillas de ventilación laterales para disipar el calor generado por el convertidor de voltaje y el procesador durante sesiones largas de uso.

La distribución de los componentes se organizó de la siguiente forma:

Interfaz de usuario (vista posterior). Como se observa en la Figura 3.7, la cara posterior del dispositivo aloja la pantalla LCD, protegida por un marco empotrado. Esto permite al operador visualizar en tiempo real el mapa de profundidad para encuadrar correctamente el objeto de interés antes de guardar la muestra.

Figura 3.7. Vista posterior del prototipo para recolección de datos.

Sistema de captura (vista frontal). La Figura 3.8 muestra la cara frontal donde se encuentra montado el módulo MaixSense A010. Esta disposición externa asegura que el campo de visión (FoV) del sensor no tenga obstrucciones físicas.

Figura 3.8. Vista frontal del prototipo para recolección de datos.

Controles físicos

Se integró un botón pulsador (*push-button*) en la parte superior del chasis, simulando el disparador de una cámara comercial para facilitar la ergonomía.

En el lateral derecho se ubicaron dos interruptores deslizantes (switches) independientes: uno para energizar la tarjeta de desarrollo y otro para el sensor ToF, permitiendo un reinicio selectivo de los módulos en caso de fallo.

Lógica de operación

La captura de muestras se gestionó mediante una máquina de estados señalizada por un LED RGB integrado en la placa, garantizando que sólo se almacenaran datos válidos:

- Estado de espera (LED azul parpadeante): Al encender el sistema mediante los interruptores laterales, la LCD muestra el video en tiempo real, permitiendo buscar el encuadre sin grabar.
- Captura y escritura (LED verde fijo): Al pulsar el botón superior (obturador), se congela el fotograma actual y se escribe la matriz de datos en la tarjeta MicroSD.

Esta configuración permitió recolectar un conjunto de datos controlado y balanceado (aproximadamente 280 muestras por clase), minimizando el ruido por movimiento y mejorando la calidad de cada imagen.

3.2.2. Definición de clases y protocolo de adquisición de datos

Para el entrenamiento del modelo de red neuronal, se definió un problema de clasificación supervisada de tres categorías. La selección de estas clases responde a la necesidad de identificar elementos que pudieran resultar relevantes para la navegación segura de una persona con discapacidad visual en un entorno de interiores. La Figura 3.9 muestra de forma general la secuencia que se siguió para generar el *dataset* de imágenes ToF.

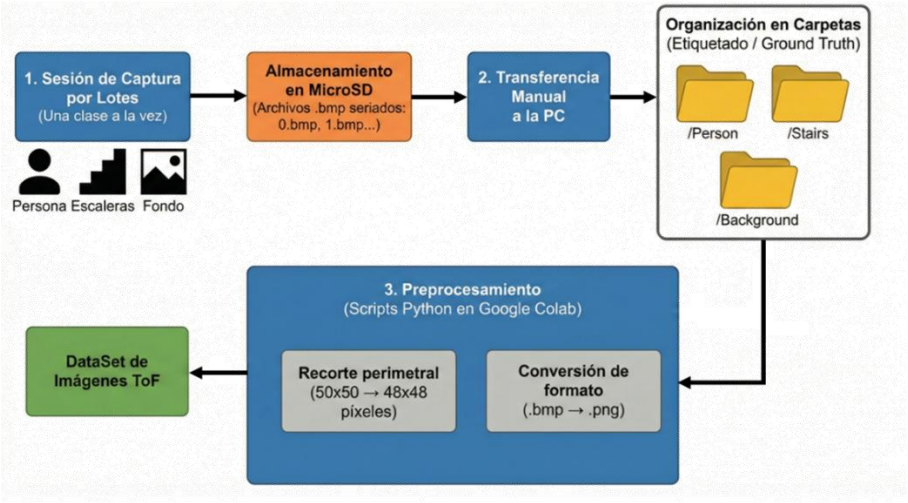

Figura 3.9. Diagrama de bloques para recolección y limpieza de datos.

Distribución del Dataset

Se planteo construir un conjunto de datos balanceado, compuesto por un total de 836 muestras, distribuidas entre las tres clases objetivo, como se indica en la Tabla 3-3. El balanceo de datos es un aspecto importante en aplicaciones de TinyML para evitar que el modelo genere sesgos de predicción hacia la clase mayoritaria.

Tabla 3-3. Distribución de clases y número de muestras.

ID de Clase	Etiqueta (*Label*)	Descripción	Muestras
0	*Background*	Fondo general, pasillos libres y paredes sin obstáculos inmediatos.	278
1	*Person*	Sujetos estáticos o en movimiento frente al sensor (obstáculo dinámico).	280
2	*Stairs*	Escaleras (obstáculo estructural/cambio de nivel).	278
Total			836

La adquisición de datos se restringió estrictamente a ambientes controlados en interiores. Esta decisión se tomó para aislar variables complejas como la saturación por luz solar directa (común en sensores ToF en exteriores) y centrar esta etapa de la investigación en la validación de la capacidad de clasificación geométrica del sistema.

Para garantizar que las muestras fuesen representativas del uso real del dispositivo, se estableció el siguiente protocolo de captura.

Considerando que el prototipo final está pensado para ser un dispositivo portátil, se determinó que las posiciones más ergonómicas para su colocación serían el pecho (mediante un arnés) o la cabeza (en gafas/diadema). Por consiguiente, el prototipo para recolección de datos se situó a una altura promedio de 1.50 metros sobre el nivel del suelo. Esto simula la perspectiva que tendrá el sistema al ser usado por el usuario.

También, para mejorar la robustez del modelo, se introdujeron variaciones controladas dentro de cada categoría:

- **Clase persona:** Se capturaron imágenes de sujetos de género masculino y femenino con diferentes complexiones físicas. Además, se variaron las poses (frente, perfil, brazos cruzados) para evitar que el modelo memorizara una silueta única.
- **Clase escaleras:** Se tomaron muestras desde diversos ángulos de aproximación (frontal y diagonal), simulando cómo un usuario podría acercarse a una escalera en la vida real.

En ambos casos, se descartaron tomas con ángulos de inclinación excesivos hacia el techo o el suelo. El protocolo de recolección indicaba mantener el sensor en una posición horizontal natural, coherente con la mirada de una persona al caminar, evitando posiciones raras que no aportarían valor al contexto de navegación y podrían introducir ruido en el entrenamiento.

3.3. Diseño y entrenamiento del modelo TinyML

El núcleo inteligente del sistema de asistencia recae en un modelo de aprendizaje automático capaz de interpretar la geometría del entorno a partir de los mapas de profundidad proporcionados por el sensor ToF. Dado que las imágenes preprocesadas poseen una estructura espacial bidimensional, se optó por implementar una Red Neuronal Convolucional (CNN).

A diferencia de las Redes Neuronales Densas (MLP) o Redes Neuronales Profundas (DNN), las CNN son especialmente eficientes para tareas de visión artificial, ya que pueden extraer características jerárquicas, como bordes verticales de escaleras o siluetas de personas, manteniendo la relación espacial de los píxeles y reduciendo drásticamente el número de parámetros de entrenamiento.

Sin embargo, el despliegue de este modelo en un sistema empotrado presenta desafíos únicos asociados al paradigma TinyML. El microcontrolador STM32H7, aunque potente, posee recursos de memoria finitos (RAM y FLASH) y un manejo de energía limitado. Por lo cual, el diseño de la arquitectura de la red no buscó únicamente maximizar la precisión de la clasificación, sino también optimizar el tamaño del modelo y la carga computacional para lograr una inferencia en tiempo real.

3.3.1. Entorno de desarrollo y framework

El proceso de creación del modelo TinyML no se limitó al uso de una única plataforma, sino que se optó por una estrategia híbrida que aprovecha la flexibilidad de los entornos de investigación (Google Colab) y la capacidad de optimización de las herramientas de despliegue empotrado (Edge Impulse). Esta metodología se dividió en dos partes.

Diseño y prototipado (Google Colab)

En una primera etapa, se utilizó Google Colab [39] como laboratorio experimental. Este entorno permitió prototipar la arquitectura de la red neuronal utilizando las bibliotecas de alto nivel Keras [40] sobre el *backend* de TensorFlow [41].

El objetivo principal de esta fase fue diseñar una arquitectura de red que no solo fuera precisa, sino que fuese intrínsecamente compatible con la arquitectura del hardware de destino. Considerando que el microcontrolador STM32H7 trabaja con un conjunto de instrucciones de 32 bits, se diseñaron las capas convolucionales utilizando cantidades de filtros múltiplos de este valor. Esta decisión de diseño busca optimizar el uso de los registros del procesador y la alineación en memoria durante las operaciones de multiplicación de matrices, facilitando el trabajo posterior del compilador.

- **Google Colaboratory**, o simplemente Colab, es un entorno de bibliotecas Jupyter alojado en la nube de Google que permite escribir y ejecutar código Python directamente en el navegador sin necesidad de realizar configuración alguna en un equipo local. Esta herramienta es ampliamente utilizada por estudiantes, investigadores y científicos de datos, ya que ofrece acceso gratuito a recursos computacionales de alto rendimiento, como GPUs y TPUs, facilitando el entrenamiento de modelos de aprendizaje automático y el análisis de datos.

> ➢ **TensorFlow** funciona como el motor o plataforma de infraestructura de código abierto que gestiona los cálculos matemáticos complejos y el despliegue de modelos, mientras que **Keras** actúa como la API de alto nivel (la interfaz de usuario) diseñada para la ergonomía humana. En la práctica, Keras abstrae la complejidad técnica permitiendo construir y entrenar redes neuronales profundas (*Deep Learning*) de manera intuitiva y rápida, mientras que TensorFlow ejecuta todo el procesamiento pesado en el fondo, optimizando el rendimiento tanto en CPUs como en GPUs.

Fase de implementación y compilación (Edge Impulse)

Una vez validada la arquitectura en el entorno de pruebas, la configuración del modelo se trasladó a la plataforma Edge Impulse [42]. Si bien es posible exportar modelos directamente desde TensorFlow Lite, se eligió Edge Impulse por su capacidad para realizar una validación comparativa entre los dos motores de inferencia utilizados para este proyecto:

- **TensorFlow Lite for Microcontrollers (TFLite)** es un formato que genera un modelo interpretado (.tflite) [38]. Se identificó como la opción ideal para las etapas de pruebas rápidas en OpenMV IDE (basado en MicroPython), ya que permite cargar y ejecutar el modelo sin necesidad de recompilar todo el firmware, facilitando la depuración del sistema.

- **EON Compiler (C++)** es un compilador utilizado para la versión final del firmware del sistema [43]. A diferencia de TFLite, que requiere un intérprete cargado en memoria RAM para leer el modelo, el EON Compiler compila la red neuronal directamente a código C++. Esto elimina la sobrecarga del intérprete, reduciendo el uso de memoria RAM y Flash entre un 25% y 55% en comparación con TFLite estándar, además de permitir optimizaciones por defecto de DSP específicas para el núcleo Cortex-M7.

> ➢ **Edge Impulse** es una plataforma enfocada al desarrollo de aplicaciones de aprendizaje automático en dispositivos empotrados, conocido como TinyML, diseñada para permitir a los desarrolladores crear aplicaciones inteligentes que se ejecutan en el borde (*edge*) del mundo físico. Esta herramienta abarca todo el ciclo de vida del proyecto: desde la recolección de datos a través de sensores y el procesamiento avanzado de señales, hasta el entrenamiento de modelos y su despliegue.

3.3.2. Arquitectura de la Red Neuronal Convolucional (CNN)

El diseño del modelo CNN se realizó con base en tres criterios relevantes para la optimización de hardware empotrado:

1) Se seleccionó una cantidad de filtros que fuese múltiplo de 16 y 32. Dado que el núcleo ARM Cortex-M7 opera con un bus y registros de 32 bits, esta dimensión facilita la paralelización de operaciones (SIMD) y optimiza el acceso a memoria durante la inferencia, evitando ciclos de reloj adicionales por realineación de datos.

2) Considerando que el *dataset* recolectado consta de aproximadamente 836 muestras, existe un riesgo de sobreajuste. Para mitigarlo, se introdujo una capa de *Dropout* del 25% antes de la clasificación. Esta técnica desactiva aleatoriamente una cuarta parte de las neuronas durante cada paso del entrenamiento, forzando a la red a aprender características más robustas y generalizables en lugar de memorizar el ruido de las imágenes de entrenamiento.

3) El uso de capas de *MaxPooling* 2×2 después de cada convolución permite reducir la cantidad de parámetros entrenables y la carga computacional en un 75% por cada bloque, manteniendo las características más notorias detectadas por los filtros.

Tabla 3-4. Arquitectura de la Red Neuronal Convolucional (CNN).

Capa	Tipo	Filtros/Nodos	Configuración	Activación	Descripción
Entrada	*Input*	-	48×48×1	-	Imagen de profundidad.
1	*Conv2D*	16	Kernel 3×3 Pad: Same	ReLU	Extracción de características. Restricción MaxNorm (1).
2	*MaxPooling2D*	-	Pool 2×2 Stride 2	-	Reducción espacial (24×24).
3	*Conv2D*	32	Kernel 3×3 Pad: Same	ReLU	Extracción de patrones complejos. Restricción MaxNorm (1).
4	*MaxPooling2D*	-	Pool 2×2 Stride 2	-	Reducción espacial (12×12).
5	*Flatten*	-	-	-	Aplanado del tensor (12×12×32 → 4608).
6	*Dropout*	-	Tasa: 0.25	-	Regularización por desconexión aleatoria.
Salida	*Dense*	3	-	Softmax	Vector de probabilidad de clases.

Finalmente, la arquitectura del modelo CNN consta de cinco capas principales. A diferencia de configuraciones estándar, se aplicó una restricción de norma máxima (*MaxNorm*) en los núcleos convolucionales y un relleno simétrico (`padding='same'`) para preservar la dimensionalidad espacial en las etapas de filtrado, delegando la reducción de tamaño exclusivamente a las capas de agrupamiento (*Pooling*). En la Tabla 3-4 se desglosa la estructura general del modelo CNN que se propone.

`kernel_constraint = MaxNorm(1):` Esto es una técnica de regularización adicional (además del *Dropout*) que impide que los pesos de la red crezcan demasiado, ayudando a evitar el sobreajuste y haciendo el modelo más estable numéricamente para la cuantización posterior

`padding ='same':` Esto significa que las capas convolucionales mantienen el tamaño espacial de la imagen (agregan ceros alrededor). La reducción de tamaño ($48 \rightarrow 24 \rightarrow 12$) ocurre exclusivamente en el *MaxPooling*, no en la convolución. Esto es una decisión de diseño para preservar información en los bordes.

- La paralelización **SIMD** (*Single Instruction, Multiple Data*) en un núcleo Cortex-M7 funciona aprovechando la capacidad de sus registros de 32 bits para manipular múltiples datos más pequeños simultáneamente en un solo ciclo de reloj, en lugar de procesar un solo dato a la vez (SISD).
  En la práctica, esto significa que el STM32H7 puede empaquetar dos números de 16 bits o cuatro números de 8 bits dentro de un único registro y aplicar la misma operación aritmética (como una suma o multiplicación) a todos ellos al mismo tiempo.

3.3.3. Entrenamiento del modelo CNN

Una vez definida la arquitectura, se procedió a la fase de entrenamiento supervisado. Para mejorar la capacidad de generalización del modelo, se aplicó una estrategia de validación cruzada dividiendo el dataset balanceado de 836 imágenes en dos subconjuntos:

- Conjunto de entrenamiento (*Training Set*): 80% de las muestras (669 imágenes) para el ajuste de pesos.
- Conjunto de validación (*Validation Set*): 20% de las muestras (167 imágenes) para la evaluación imparcial del desempeño.

Selección de hiperparámetros

El proceso de optimización se configuró utilizando el algoritmo Adam, seleccionado por su eficiencia en el manejo de gradientes dispersos. La función de costo a minimizar fue la entropía

cruzada categórica (*Categorical Cross-entropy*), adecuada para problemas de clasificación multiclase.

Los hiperparámetros definitivos se establecieron tras una serie de pruebas experimentales.

El tamaño del lote (*Batch Size*) se fijó en 32 muestras. Este valor proporcionó un equilibrio óptimo, permitiendo suficientes actualizaciones de los pesos por época (aprox. 21 iteraciones) para capturar los detalles finos de las clases sin introducir una inestabilidad excesiva en el gradiente.

La tasa de aprendizaje (*Learning Rate*) se estableció en 0.0005. Dado el tamaño reducido del *dataset*, esta tasa conservadora permitió que el modelo descendiera suavemente hacia el mínimo global de error, evitando oscilaciones abruptas que pudieran impedir la convergencia.

3.4. Entornos de desarrollo integrado y configuración inicial

El núcleo de procesamiento del sistema es la tarjeta Arduino Nicla Vision, basada en el microcontrolador STM32H747AII6. Este componente presenta una arquitectura de doble núcleo (Dual-Core) que combina un procesador Arm Cortex-M7 a 480 MHz y un Cortex-M4 a 240 MHz. Aprovechar esta arquitectura es una buena alternativa para evitar cuellos de botella en un sistema de tiempo real que debe procesar imágenes y datos de distancia simultáneamente.

Inicialmente, la tarjeta se configuro con el *bootloader* de OpenMV, optimizado para el desarrollo rápido en MicroPython. Sin embargo, para la implementación final del prototipo se optó por migrar al entorno Arduino IDE (C++).

Antes de pasar al entorno de Arduino IDE, en la siguiente subsección se realiza una descripción del flujo de trabajo necesario para realizar una configuración personalizada del *firmware* de la tarjeta Nicla Vision en OpenMV.

3.4.1. Configuración del firmware de Nicla Vision con OpenMV

Para compilar un firmware modificado se debe realizar a partir del código fuente de OpenMV que se mantiene en su repositorio oficial [44]. Allí se describen dos flujos de trabajo para Linux y Docker (véase Figura 3.10).

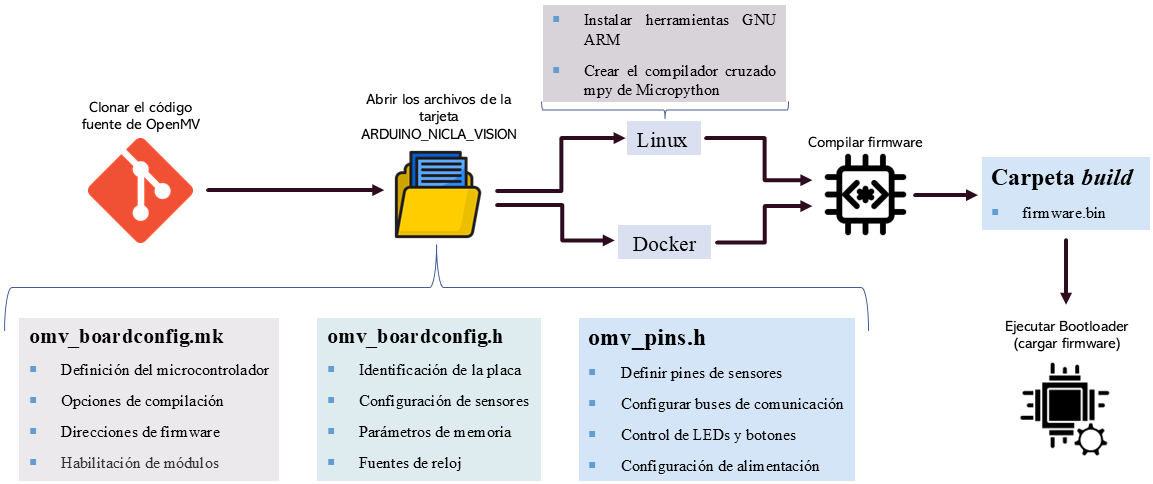

Figura 3.10. Flujo general para configuración de firmware con OpenMV.

Nota: En este proyecto se comenzó utilizando la opción Linux, corriendo Ubuntu desde el Subsistema de Windows para Linux (*Windows Subsystem for Linux, WSL*), sin embargo, después de una actualización al repositorio, se presentaron problemas al intentar compilar el *firmware*.

En el momento que se realiza este documento, aún se puede presentar este error. Por recomendación de los ingenieros de OpenMV, es mejor utilizar la opción de Docker, siguiendo los pasos que se muestran en su repositorio.

Ya sea que se utilice Linux o Docker, es necesario clonar el código fuente de OpenMV para poder modificarlo desde un entorno local.

Importante: si se utiliza Linux, se requiere tener instalado git previamente, de lo contrario, primero se instala esta paquetería y después se continua con lo que se menciona a continuación.

En la compilación con Docker se utiliza el siguiente comando para clonar el repositorio:

```
git clone https://github.com/openmv/openmv.git --depth=50
```

Para Linux:

```
git clone --recursive https://github.com/openmv/openmv.git
```

Una vez que ya se tiene el código fuente en la computadora, se puede utilizar algún editor de código para abrir todo el proyecto o sólo el archivo de interés.

Para esta primera parte, se agregará el módulo que controla la pantalla LCD. Algo bueno es que en versiones recientes de MicroPython, ya se incluye la clase `SPIDisplay` [45], utilizada para controlar pantallas LCD SPI. Sin embargo, que ya se encuentre dentro de las librerías de MicroPython no garantiza que se pueda usar en un código. Para eso se necesita habilitar el módulo del display; `MICROPY_PY_DISPLAY`, o en caso de no estar incluido, se tendrá que crear uno.

Después de descargar el código fuente, se identifica el fichero con el nombre *boards*, en esta carpeta se encuentran todas las tarjetas que son compatibles con OpenMV. Para este caso, la tarjeta que se está utilizando es la Arduino Nicla Vision, por lo que, es necesario dirigirse al fichero que tiene ese nombre.

Dentro de la carpeta ARDUINO_NICLA_VISION se encuentran varios archivos, el primero que se debe revisar es el `omv_boardconfig.mk`. Este archivo contiene variables y opciones que determinan cómo se compila el firmware para una placa en particular, incluyendo:

- Definición del microcontrolador (MCU), que especifica el modelo de procesador utilizado.
- Opciones de compilación (CPU, FPU, PORT), que configuran el tipo de arquitectura y unidad de punto flotante.
- Direcciones de firmware (`OMV_FIRM_ADDR`), que indican dónde se almacena el firmware en la memoria.
- Habilitación de módulos (`MICROPY_PY_*`), que activan o desactivan características como bluetooth, redes, procesamiento de imágenes, etc.

En la Figura 3.11 se observa cómo se debe habilitar el módulo para el display; `MICROPY_PY_DISPLAY = 1`, así mismo, si se requiere inhabilitar algún modulo con la intención de ahorrar espacio en memoria, se puede hacer desde este mismo archivo, sin la necesidad de borrar código adicional dentro del archivo fuente de OpenMV.

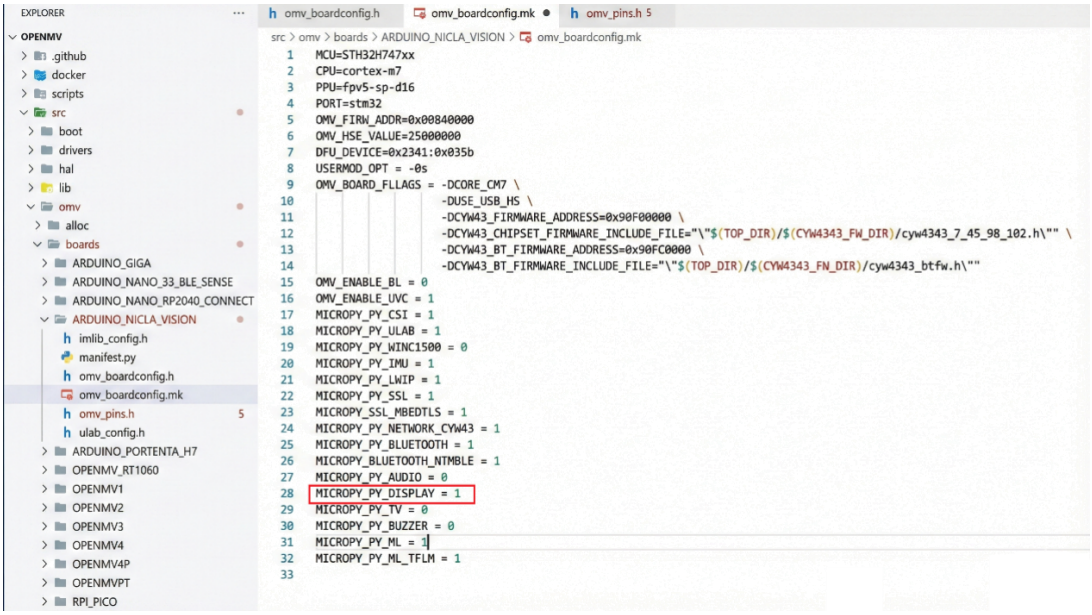

Figura 3.11. Código fuente de OpenMV, archivo omv_boardconfig.mk.

Después de asegurarse que el módulo para controlar el display está definido, es momento de pasar al siguiente archivo de interés, que es el `omv_boardconfig.h`.

El archivo `omv_boardconfig.h` en el código fuente de OpenMV se utiliza para definir la configuración de hardware y los parámetros específicos de cada placa compatible con OpenMV.

Este archivo contiene macros y definiciones que establecen aspectos como:

- Identificación de la placa (`OMV_BOARD_ARCH`, `OMV_BOARD_TYPE`), que especifica el modelo y arquitectura.
- Configuración de sensores (`OMV_OV2640_ENABLE`, `OMV_OV5640_ENABLE`), que habilita o deshabilita sensores de imagen compatibles.
- Parámetros de memoria (`OMV_OSC_PLL1N`, `OMV_OSC_PLL2N`), que configuran la velocidad del reloj y la memoria SDRAM.
- Fuentes de reloj (`OMV_OSC_PLL_CLKSOURCE`, `OMV_OSC_USB_CLKSOURCE`), que determinan cómo se sincronizan los periféricos.
- Configuración de compresión JPEG (`OMV_JPEG_CODEC_ENABLE`, `OMV_JPEG_QUALITY_HIGH`), que ajusta la calidad de las imágenes procesadas.

Dentro de este archivo, se debe identificar la sección donde se definen las macros que se utilizarán para el control del display. En el apartado SPI LCD Interface (véase Figura 3.12) se observan estas definiciones para el módulo display. Así mismo, se puede ver a qué pines se encuentran asociadas estas macros (Por ejemplo, `OMV_SPI_DISPLAY_RST_PIN` está asociada al pin G1). En caso de querer utilizar otro pin, se puede agregar uno diferente, pero antes se debe verificar que ningún otro modulo o macro lo esté utilizando.

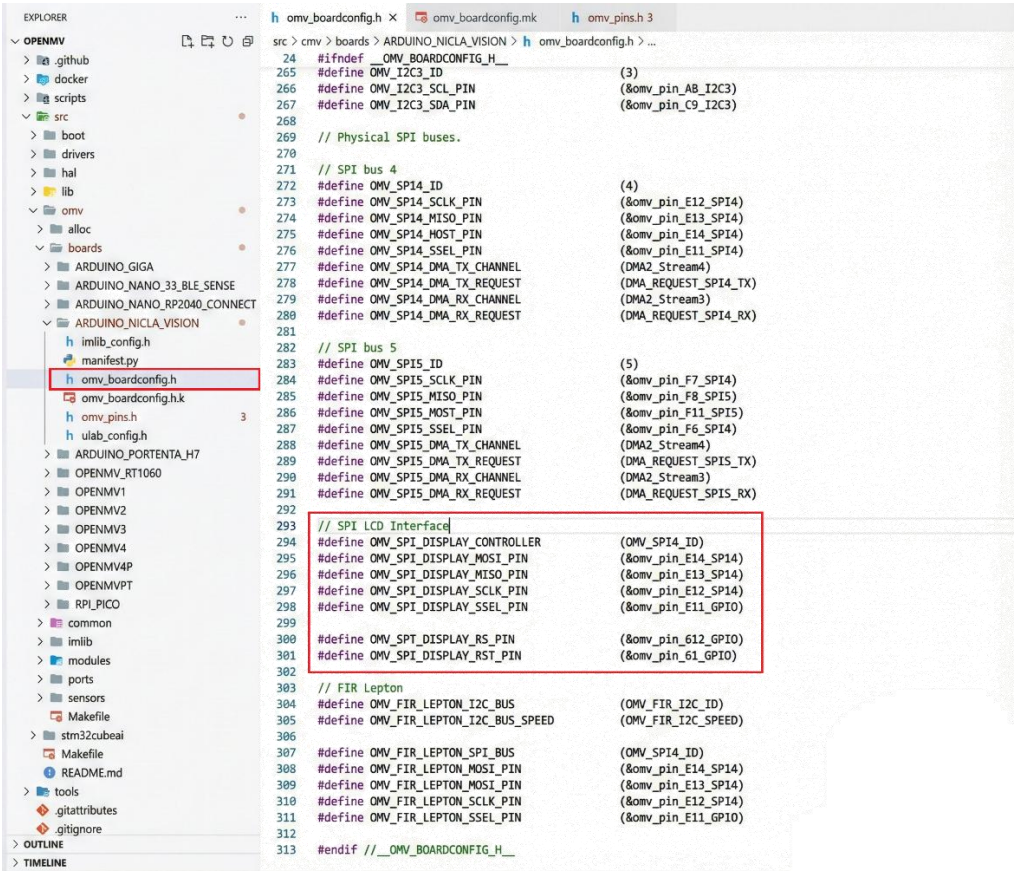

Figura 3.12. Código fuente de OpenMV abierto desde Visual Studio Code, archivo omv_boardconfig.h.

Otro archivo que puede ser interesante revisar es el `omv_pins.h`. Este archivo se utiliza para definir la asignación de pines y la configuración de hardware de la placa. Contiene macros y definiciones que establecen qué pines están conectados a periféricos como sensores, LEDs, botones y buses de comunicación (I2C, SPI, UART).

Algunas de sus funciones principales son:

- Definir pines de sensores: Especifica qué pines están conectados a sensores de imagen y otros dispositivos.
- Configurar buses de comunicación: Asigna pines para protocolos como I2C, SPI y UART.
- Control de LEDs y botones: Define pines para indicadores LED y botones de usuario.
- Configuración de alimentación: Establece pines relacionados con la gestión de energía y voltajes.

Este archivo es muy importante para garantizar que el firmware de OpenMV pueda interactuar correctamente con el hardware de cada tarjeta.

Estos son sólo algunos archivos que se pueden modificar para cambiar varias configuraciones que se requieran en una aplicación especifica. Estudiar a fondo el código fuente de OpenMV es una ardua tarea que no se explica en este trabajo, pero se puede consultar en el repositorio oficial de OpenMV [44] o la documentación de MicroPython [46].

Una vez que se ha configurado o habilitado los módulos requeridos, se guardan los cambios en el archivo y se regresa a la consola de Linux para compilar el firmware modificado.

En Docker, se utilizan los siguientes comandos:

```
cd openmv/docker
make TARGET=ARDUINO_NICLA_VISION
```

En Linux, primero se instala la cadena de herramientas GNU ARM:

```
TOOLCHAIN_PATH=${HOME}/cache/gcc
TOOLCHAIN_URL="https://developer.arm.com/-
/media/Files/downloads/gnu/13.2.rel1/binrel/arm-gnu-toolchain-13.2.rel1-
x86_64-arm-none-eabi.tar.xz"
mkdir -p ${TOOLCHAIN_PATH}
wget --no-check-certificate -O - ${TOOLCHAIN_URL} | tar --strip-
components=1 -Jx -C ${TOOLCHAIN_PATH}
export PATH=${TOOLCHAIN_PATH}/bin:${PATH}
```

Después se agregan los siguientes comandos para comenzar a compilar el *firmware*:

```
cd openmv
make -j$(nproc) -C lib/micropython/mpy-cross # Builds Micropython mpy
cross-compiler
make -j$(nproc) TARGET= ARDUINO_NICLA_VISION # Builds the OpenMV firmware
```

Al terminar la compilación aparecerá una carpeta *build* en el código fuente de openmv, dentro de esta carpeta se encontrarán estos archivos, o algo muy parecido:

- bootloader.bin – Imagen binaria del gestor de arranque (no se utiliza directamente).
- bootloader.dfu – Imagen DFU del gestor de arranque (no se utiliza directamente).
- bootloader.elf – Imagen ELF del gestor de arranque (usada para generar los archivos BIN/DFU).
- firmware.bin – Imagen binaria del firmware (usada por Herramientas -> Ejecutar Bootloader en OpenMV IDE).
- firmware.dfu – Imagen DFU del firmware (no se utiliza directamente).
- firmware.elf – Imagen ELF del firmware (usada para generar los archivos BIN/DFU).
- openmv.bin – Imagen binaria combinada del gestor de arranque + firmware (no se utiliza directamente).
- openmv.dfu – Imagen DFU combinada del gestor de arranque + firmware (usada por Herramientas -> Ejecutar Bootloader en OpenMV IDE).
- romfs<n>.img – Imagen del sistema de archivos ROM.

El archivo que se cargará en la tarjeta es el *firmware.bin*. La forma más rápida para hacer esto es desde el OpenMV IDE. Como se muestra en la Figura 3.13, desde la pestaña Herramientas se encuentra la opción cargar firmware. Únicamente se debe seleccionar el archivo correcto.

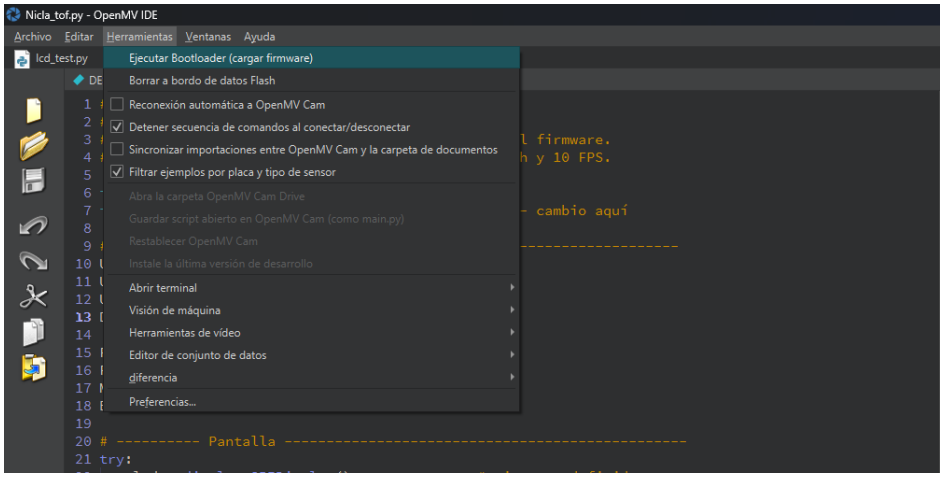

Figura 3.13. Cargar firmware de la tarjeta desde OpenMV IDE.

Después de haber implementado con éxito el nuevo firmware en la tarjeta Nicla Visión, es bueno realizar una prueba rápida para verificar que el módulo del display fue cargado correctamente. A continuación, se agrega un pequeño código de ejemplo que hace uso de la cámara de la tarjeta para posteriormente transmitir la imagen en el LCD.

```
import sensor
import display

sensor.reset()  # Inicializa el sensor de la cámara.
sensor.set_pixformat(sensor.RGB565)  # Imágenes en formato RGB565.
sensor.set_framesize(sensor.QQVGA2)  # Para un tamaño de pantalla de 128x160.

lcd = display.SPIDisplay()

while True:
    #lcd.0
    lcd.write(sensor.snapshot().rotation_corr(0,0,270))  # Toma una imagen y la
muestra en el display
```

La Figura 3.14 muestra el diagrama de conexiones completo que también incluye el sensor ToF, para probar el código que se menciona en esta subsección se puede omitir el sensor, utilizando sólo la cámara que ya integra la tarjeta Nicla Vision. También, el suministro de energía puede diferente, pero se recomiendo alimentar al módulo lcd con una fuente independiente.

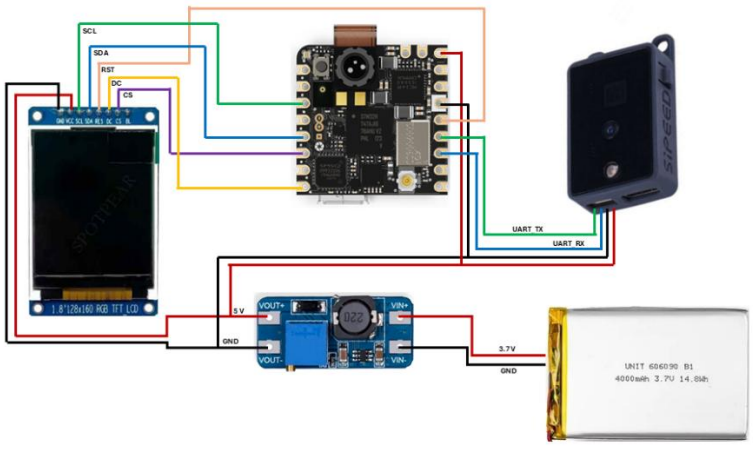

Figura 3.14. Conexión de los elementos del hardware (versión lcd).

3.4.2. Protocolo de comunicación y análisis del ancho de banda

En esta subsección, se mostrará cómo calcular y gestionar la integridad de los datos durante la comunicación entre el sensor ToF y la unidad de procesamiento, para esto, se realizó un análisis de los recursos de comunicación disponibles.

Inicialmente se evaluó utilizar la resolución máxima de 100×100 píxeles. Sin embargo, el tamaño de la trama resultante ($\sim$10 kB) representaba un riesgo elevado de desbordamiento de memoria y saturación del canal. Por ello, se decidió configurar el sensor para operar con una resolución de 50×50 pixeles, lo que reduce el tamaño del paquete a una dimensión manejable, la Ecuación 11 muestra cómo calcular el tamaño del paquete transmitido:

$$\text{Tamaño}_{Pkt} = Encabezado\ (2\text{B}) + Longitud\ (2\text{B}) + Info\ (16\text{B}) + Datos_{Img} + Checksum\ (1\text{B}) + Final\ (1\text{B}) \qquad \textit{Ecuación 11.}$$

Donde Tamaño_{Pkt} representa el tamaño del buffer para un fotograma.

Con base en la Ecuación 11. Se calculan los bytes que se ocuparán con una configuración de 50×50 pixeles.

$$\text{Tamaño}_{Pkt(50)} = 2\,500\ bytes\ (Datos) + 22\ bytes\ (Información\ adicional) = 2\,522\ bytes$$

Considerando que se está trabajando con comunicación UART, se estableció como restricción de diseño mantener una velocidad de comunicación de 921 600 baudios (BPS). Esto implica que al iniciar el programa se deba cambiar la velocidad estándar del sensor (115 200 baudios) a la nueva configuración.

Para determinar la frecuencia de adquisición máxima permitida, se calculó el ancho de banda requerido por cada fotograma, considerando que la transmisión UART asíncrona añade bits de control ($1\text{ bit de inicio} + 8\text{ de datos} + 1\text{ de parada} = 10\text{ bits por byte}$) [48]:

$$Bits_{Frame} = \text{Tamaño}_{Pkt(50)} \times 10\,\frac{bits}{byte} = 2\,522 \times 10 = 25\,220\ bits$$

Al comparar este requerimiento con el ancho de banda disponible:

Basándose en este cálculo, se estableció la tasa de muestreo en 19 fotogramas por segundo (FPS), que es la velocidad máxima que permite el módulo MaixSense A010 para comunicación por UART.

$$FPS_{Max} = \frac{Velocidad_{UART}}{Bits_{Frame}} = \frac{921\ 600\ \text{bps}}{25\ 220\ \text{bits}} \approx 36.54\ \text{FPS}$$

$$Velocidad_{Req} = 25\ 220 \times 19 \approx 479\ 180\ \text{bps}$$

Como se observa, el consumo de ancho de banda (479 180 bps) se mantiene por debajo del límite físico del canal (921 600 bps), dejando un margen de seguridad del 48% que garantiza una transmisión fluida y libre de pérdida de datos durante la construcción del *dataset*.

A diferencia de las etapas de caracterización y prototipo final donde se suele buscar el máximo rendimiento teórico, para la recolección de datos se priorizó la estabilidad de la conexión y la simplicidad del hardware, decidiendo operar con la configuración por defecto del sensor (115 200 bps).

Capítulo 4

4. Resultados

Este capítulo presenta los resultados obtenidos a lo largo del desarrollo del sistema, estructurados lógicamente desde la validación de los componentes de percepción hasta la evaluación integral del prototipo final. El análisis comienza con la caracterización técnica de los sensores de tiempo de vuelo (VL53L8 y MaixSense A010), estableciendo la viabilidad del hardware seleccionado para la adquisición de datos. A partir de ello, se detalla el proceso de generación del dataset, abarcando las etapas de preprocesamiento, etiquetado, normalización y recorte necesarias para adaptar la información de profundidad a los requerimientos de la red neuronal.

Posteriormente, se examina el desempeño del modelo de clasificación de imágenes, presentando las curvas de aprendizaje, las métricas de evaluación en los conjuntos de validación y prueba, así como una comparativa de rendimiento que justifica la arquitectura elegida. La discusión avanza hacia la implementación en el sistema embebido, describiendo la compilación de la biblioteca C++ mediante la herramienta EON Compiler y la configuración del firmware, donde se profundiza en la arquitectura de control distribuida entre los núcleos Cortex-M7 y Cortex-M4. Finalmente, se documenta la integración física de los elementos y se exponen los hallazgos de las pruebas en ambiente controlado, verificando la funcionalidad y robustez del dispositivo ante obstáculos y escenarios reales.

4.1. Caracterización de los sensores de tiempo de vuelo

Con el propósito analizar los componentes de percepción para la navegación asistida, se realizó una caracterización detallada de dos sensores basados en tecnología de Tiempo de Vuelo (ToF): el VL53L8CH y el módulo MaixSense A010. Esta etapa experimental se enfocó en evaluar el desempeño de ambos dispositivos bajo condiciones controladas, analizando parámetros como la linealidad en la medición de distancia, la inmunidad al ruido ambiental y la estabilidad de la matriz de profundidad, permitiendo seleccionar el hardware adecuado en función de su precisión, alcance y resolución para imágenes ToF multizona. A continuación, se describen los resultados obtenidos de la sección 3.1.

4.1.1. Sensor ToF VL53L8CH

En la Figura 4.1 se puede observar que en las esquinas del SPAD, como era de esperarse, se presentan errores significativos en comparación con los valores de las zonas internas.

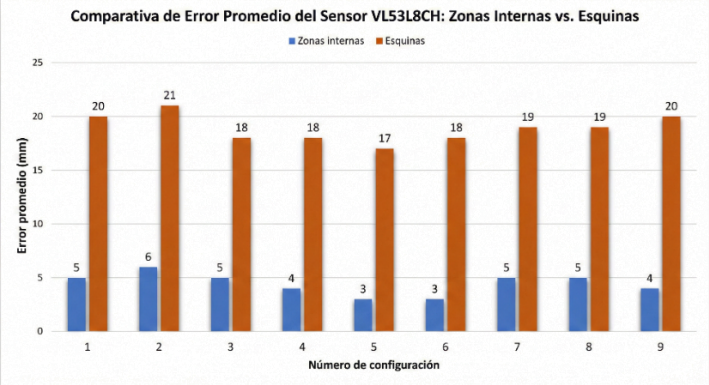

Figura 4.1. Error promedio por cada número de configuración.

Al promediar los datos de las cinco muestras que se tomaron, se obtuvieron mejores resultados. En las zonas internas los errores se mantuvieron por debajo de los 7 mm, incluso llegando a un mínimo de 3mm. Valores muy buenos para sensores ToF. En las esquinas, si bien el error aumento, se mantuvo por debajo de los 22 mm, que incluso también se podría considerar como un error aceptable. Para esta prueba, las configuraciones 5 y 6 obtuvieron los mejores resultados.

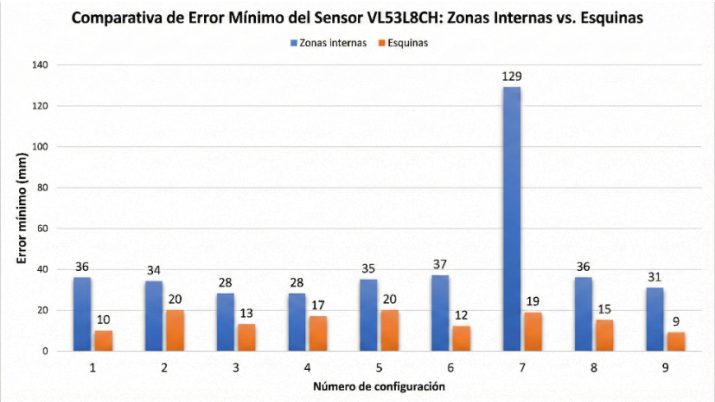

Figura 4.2. Error mínimo por cada número de configuración.

En la Figura 4.2 y Figura 4.3 se observan los errores mínimos y máximos, respectivamente, obtenidos por el sensor ToF, si bien hay un pico en la configuración 7, lo más probable es que fuese causado por ruido externo, ya que sólo se registró en una de las 5 muestras recolectadas.

En este apartado, el error mínimo en las zonas internas se mantuvo por debajo de los 40 mm, y el error máximo se mantuvo por debajo de los 50 mm.

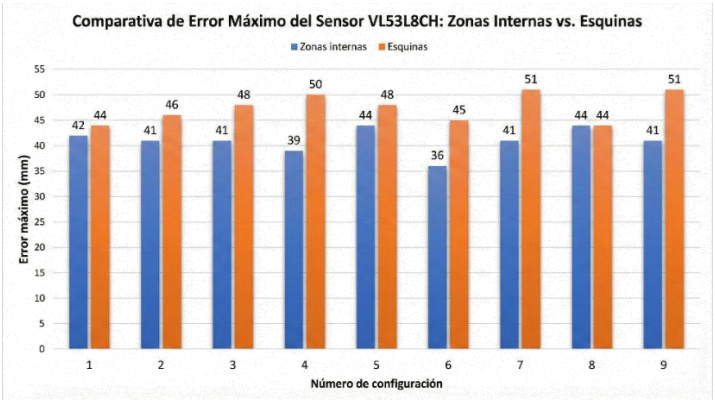

Figura 4.3. Error máximo por cada número de configuración.

Después de analizar la tres graficas anteriores, se identificó que la configuración 6 fue la que presento un mejor desempeño.

Parámetros de la configuración 6:

- Resolución: 8x8 zonas
- Frecuencia de muestreo: 15 Hz
- Factor de agrupamiento: x5
- Tamaño del contenedor: 187.5 mm

- Ventana del contenedor: 2250 milímetros → 2.25 metros

De acuerdo con las características establecidas para este proyecto, esta sería la configuración recomendada.

El objetivo principal de las pruebas era analizar el comportamiento que tiene el sensor VL53L8CH, y en general, la tecnología de tiempo de vuelo directo. Entender cómo afectan las diferentes condiciones del entorno y las distintas configuraciones internas del dispositivo era un aspecto importante en esta etapa de desarrollo.

A diferencia de sensores básicos que entregan solo una distancia, el sensor multizona construye histogramas de retorno de fotones para cada zona. Se analizó cómo la modificación del tamaño de bin (configurado por defecto a ~37.5 mm) afecta la precisión y cómo el sensor discrimina entre la señal real y el ruido ambiental. Sin embargo, a pesar de la fiabilidad y la gran cantidad de información técnica disponible para el VL53L8CH, las pruebas en el laboratorio revelaron una limitación física importante para este proyecto: su resolución máxima de 8×8 zonas (64 puntos). Al intentar recolectar imágenes ToF con esta matriz, la geometría de los objetos resultaba ambigua, lo que aumentaba la probabilidad de confusión en el modelo de clasificación.

4.1.2. Modulo ToF MaixSense A010

En la Tabla 4-1 se muestran los valores obtenidos con el módulo MaixSense A010.

Tabla 4-1. Distancias obtenidas con el módulo MaixSense A-010.

Distancia real (cm)	Distancia medida (cm)	Distancia real (cm)	Distancia medida (cm)	Distancia real (cm)	Distancia medida (cm)	Distancia real (cm)	Distancia medida (cm)
10	10	80	83	150	160	220	239
20	20	90	94	160	170	230	249
30	30	100	106	170	181	240	250
40	40	110	117	180	193	250	250
50	52	120	128	190	204	-	-
60	62	130	139	200	215	-	-
70	73	140	149	210	227	-	-

En la Figura 4.4 se muestra cómo las lecturas del sensor (línea continua) se apartan de la línea ideal (línea discontinua), aproximadamente, en 90 cm se comienza a notar una diferencia más significativa.

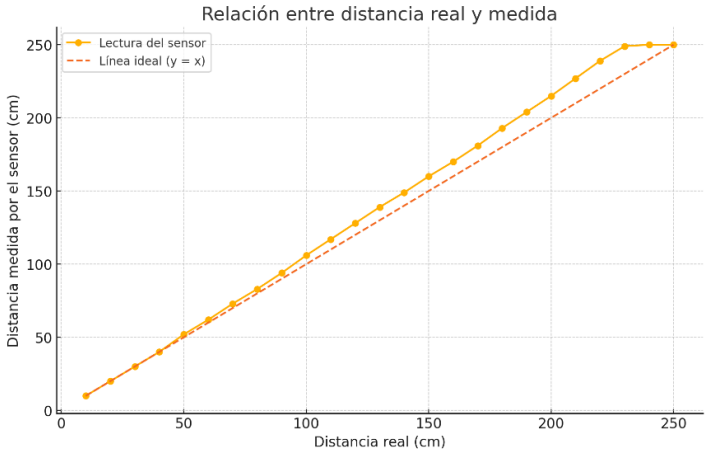

Figura 4.4. Distancia real vs. distancia medida.

En la Figura 4.5 se observa de forma directa cuántos centímetros se desvía la lectura a cada distancia; el error crece progresivamente a partir de ~50 cm y ronda 10–19 cm por encima de 180 cm, antes de saturarse a 250 cm.

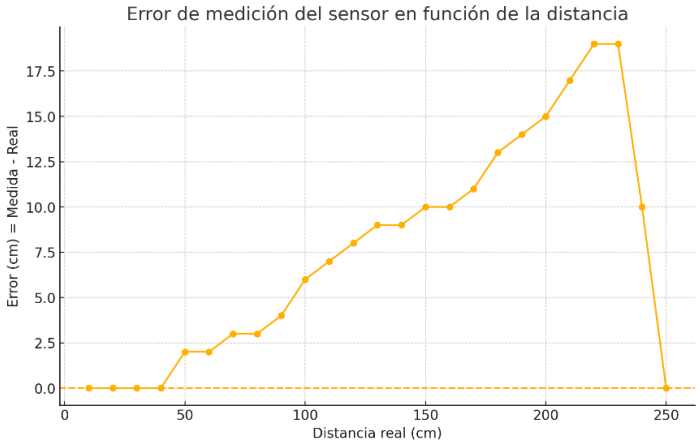

Figura 4.5. Error de medición = (medida – real).

4.2. Generación del dataset de imágenes ToF

Esta sección describe el flujo de trabajo para la conformación del dataset, el cual transforma las matrices de distancia en imágenes en escala de grises para el modelo TinyML. El proceso abarca desde las etapas de preprocesamiento y etiquetado manual de las clases objetivo, hasta la adecuación de las muestras mediante normalización y conversión de distancias a un formato de imagen de 8 bits. Asimismo, se detalla la operación de recorte necesaria para ajustar la resolución a la entrada de la red, obteniendo finalmente un conjunto de imágenes normalizado y optimizado para la fase de entrenamiento.

4.2.1. Preprocesamiento y etiquetado (*Ground Truth*).

Una vez definido el protocolo de captura, se procedió a la construcción del conjunto de datos. Dado que el prototipo de recolección (descrito en la sección 3.2.1) no contaba con una interfaz para ingresar etiquetas de texto en tiempo real, se implementó una estrategia para guardar las imágenes ToF de las clases por lotes.

El firmware de recolección fue programado para guardar las imágenes de profundidad asignándoles automáticamente un nombre seriado numérico (iniciando en 0.bmp con incrementos unitarios). Al no existir un identificador de clase en el nombre del archivo, fue necesario que en cada sesión de grabación se capturarán exclusivamente muestras pertenecientes a una sola clase (por ejemplo, solo *Escaleras*). Esto evitó la contaminación del dataset con información cruzada o basura.

Al finalizar cada sesión, los archivos almacenados en la raíz de la memoria MicroSD eran transferidos manualmente a una estructura de directorios en el ordenador, organizándolos en carpetas etiquetadas con el nombre de la clase correspondiente (*/Person*, */Stairs*, */Background*). Esta organización física de los archivos constituye el etiquetado (*Ground Truth*) que posteriormente utiliza el algoritmo de entrenamiento supervisado.

4.2.2. Formato de imagen y cuantización

Las imágenes ToF se almacenaron originalmente en formato .bmp (Bitmap) para evitar la compresión con pérdida en la etapa inicial. Cada píxel de la matriz ToF, que representa una

medición de distancia, fue interpretado como un valor de intensidad en escala de grises con una profundidad de color de 8 bits.

Esta cuantización previa (realizada por el hardware del sensor) simplificó el manejo de los datos, mapeando las distancias a un rango entero de [0 - 255]. En esta representación, los valores bajos (cercanos a 0) corresponden a objetos próximos al sensor, mientras que los valores altos (cercanos a 255) representan el fondo o el límite del rango de medición (véase Figura 4.6).

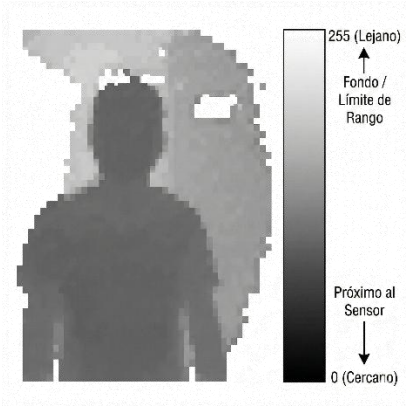

Figura 4.6. Interpretación de imagen ToF a imagen en escala de grises.

4.2.3. Normalización y recorte

Para preparar los datos para la red neuronal convolucional (CNN), se aplicó un flujo de preprocesamiento utilizando scripts de Python en Google Colab. Este proceso incluyó dos transformaciones importantes.

1) Las imágenes fueron convertidas de .bmp a .png, un formato más eficiente y ampliamente soportado por las bibliotecas de aprendizaje profundo como TensorFlow y Keras.

2) Aunque el sensor entrega una resolución de 50×50 píxeles, la arquitectura de la CNN se diseñó para una entrada de 48×48 píxeles (un tamaño de entrada que facilita las operaciones de convolución y agrupamiento). Para ajustar las imágenes sin deformarlas mediante reescalado, se aplicó un recorte perimetral, eliminando un borde de 1 píxel en los cuatro lados de la imagen. Este recorte tiene el beneficio adicional de descartar las zonas más ruidosas del sensor, que suelen concentrarse en los bordes de la matriz SPAD.

Tabla 4-2. Conjunto de imágenes ToF que integran el dataset.

Clase	Imágenes ToF
Persona	
Escaleras	
Fondo general	

4.3. Modelo de clasificación de imágenes ToF

Esta sección presenta la evaluación del desempeño del modelo de clasificación de imágenes CNN, abarcando desde su fase de aprendizaje hasta su rendimiento en el microcontrolador simulado. En primer lugar, se analizan las métricas de entrenamiento y validación, examinando las curvas de exactitud y pérdida para descartar fenómenos de sobreajuste. Posteriormente, el algoritmo se pasa por una etapa de pruebas (*model testing*) con datos inéditos para verificar su capacidad de generalización ante nuevas muestras. Finalmente, se realiza un estudio comparativo de rendimiento en el microcontrolador, contrastando los tiempos de inferencia y el consumo de recursos (memoria RAM y FLASH) entre las implementaciones de TensorFlow Lite (Float32 e Int8) y la optimización mediante EON Compiler, con el objetivo de seleccionar la opción más eficiente para el sistema.

4.3.1. Entrenamiento y validación del modelo

Se configuró un ciclo inicial de 100 épocas. Durante este proceso, se monitorearon las curvas de precisión (*accuracy*) y pérdida (*loss*) para identificar el punto óptimo de aprendizaje.

El análisis de las gráficas de entrenamiento reveló un comportamiento de convergencia rápida.

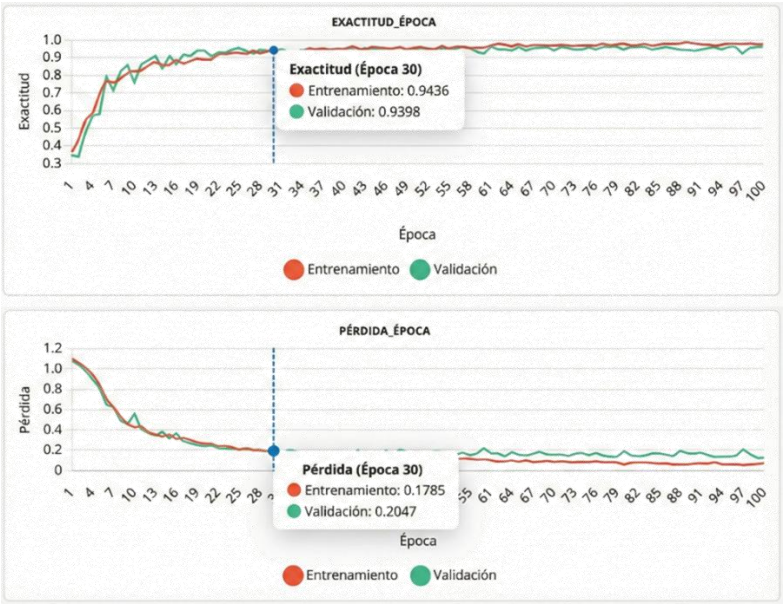

Figura 4.7. Gráficas de exactitud y pérdida del modelo CNN con convergencia rápida (100 épocas).

Se observó que el modelo alcanzaba una estabilidad significativa alrededor de la época 30, logrando una exactitud de validación del 93.9% y una pérdida de 0.20 (véase Figura 4.7). Para comprobar estos valores, se volvió a entrenar el modelo, pero en esta ocasión se configuro a 30 épocas, dando como resultado una exactitud de validación del 94.7% y una pérdida de 0.19 (véase Figura D.2). Esto indica que la arquitectura propuesta es capaz de extraer las características relevantes de manera eficiente sin requerir tiempos de cómputo excesivos.

Sin embargo, al permitir que el entrenamiento continuara hasta las 100 épocas, se logró un refinamiento adicional en los pesos, alcanzando una precisión final de 96.2% y reduciendo la pérdida a 0.12 en el conjunto de validación (véase Figura 4.8).

Este comportamiento confirma que el modelo no presenta signos de sobreajuste, ya que la pérdida de validación continuó disminuyendo junto con la de entrenamiento, estabilizándose en un rendimiento superior al 96%.

Para comprobar la fiabilidad del modelo antes de su implementación en el microcontrolador, se ejecutó un protocolo de evaluación dividido en dos etapas: validación técnica del entrenamiento y pruebas de desempeño con datos no vistos por el modelo, analizando métricas globales y la eficiencia de recursos en el dispositivo.

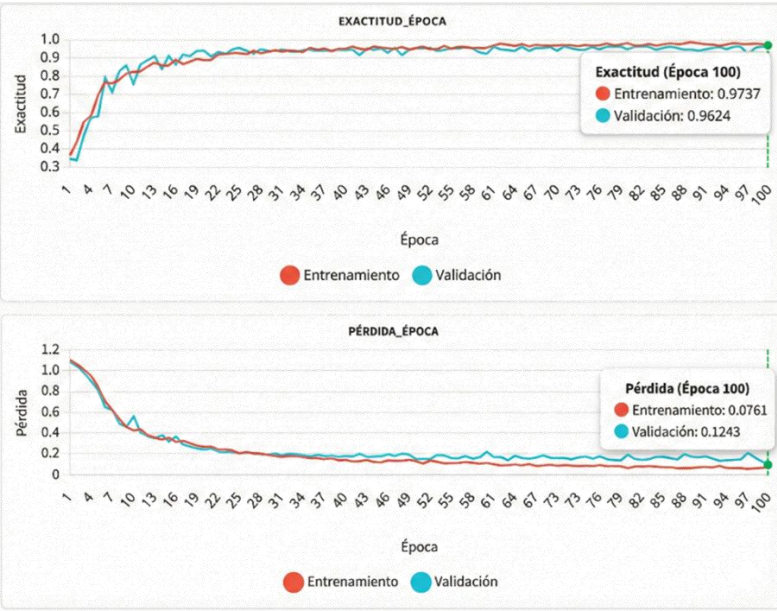

Figura 4.8. Gráficas de exactitud y pérdida durante el entrenamiento y validación del modelo CNN (100 épocas).

Este análisis no se limitó a una única versión del modelo, sino que comparó el desempeño de dos representaciones numéricas distintas para verificar la integridad de las predicciones:

- **Modelo no optimizado (Float32):** Utiliza aritmética de punto flotante de 32 bits. Representa la capacidad de procesamiento de la red neuronal con una elevada precisión matemática.
- **Modelo cuantizado (Int8):** Convierte los pesos y activaciones a enteros de 8 bits. Esta conversión es importante para la eficiencia en aplicaciones de sistemas empotrados, pero conlleva el riesgo de degradar la precisión.

4.3.1.1. Métricas de desempeño en el conjunto de validación

Durante la fase de validación, utilizando un subconjunto de 133 muestras reservadas del dataset original, se monitoreó el comportamiento de ambas versiones.

Los resultados mostraron una convergencia idéntica entre el modelo float32 y el int8. Esto es un buen indicador, lo que sugiere que los patrones aprendidos por la red son lo suficientemente distintivos como para mantenerse después de la reducción de precisión numérica, sin alterar la clasificación final en este conjunto de datos.

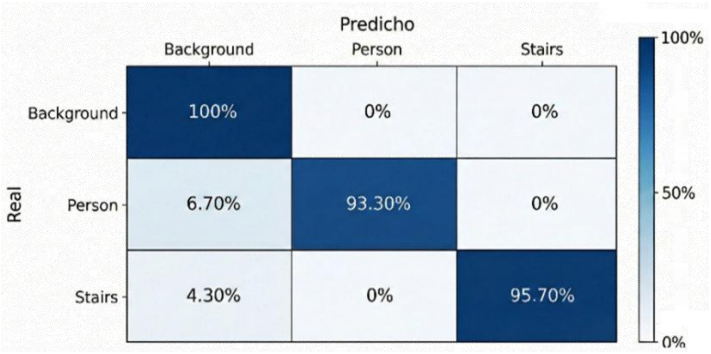

Figura 4.9. Matriz de confusión para conjunto de validación (float32 e int8).

Tabla 4-3. Métricas de validación por clase (float32 e int8).

Clase	Precisión	Recall	F1 Score	Interpretación
Background	0.89	1	0.94	Detecta todo el fondo, aunque absorbe algunos falsos negativos de otras clases.
Person	1	0.93	0.97	Alta fiabilidad: Si predice Persona, es correcto el 100% de las veces.
Stairs	1	0.96	0.98	Alta fiabilidad: Si predice Escalera, es correcto el 100% de las veces.

Tabla 4-4. Métricas globales para conjunto de validación (Float32 e Int8).

Exactitud	Pérdida	Precisión	Recall	F1-Score	Área bajo la curva ROC
0.962	0.12	0.97	0.96	0.96	0.99

El análisis de estas métricas revela un área bajo la curva ROC de 0.99 y un F1-Score promedio de 0.96 (véase Tabla 4-4). Con base en estos resultados, se pueden identificar dos comportamientos que sobresalen en esta prueba.1) La clase Background fue identificada con un 100% de efectividad, sin confundir en ningún caso el fondo con un obstáculo u objeto, lo que sugiere una fuerte solidez ante falsos positivos (FP). 2) Los únicos errores registrados, 3 personas (6.7%) y 2 escaleras (4.3%) clasificadas como fondo, corresponden a falsos negativos. Si bien idealmente se busca reducir estos casos, la alta precisión global sugiere que el modelo es confiable, y estos fallos puntuales podrán ser mitigados mediante el umbral de confianza en la etapa de pruebas.

4.3.1.2. Métricas de desempeño en el conjunto de pruebas

Para simular las condiciones operativas reales y evaluar la capacidad de generalización del sistema, se sometió al modelo a un conjunto de prueba independiente (*Test Set*) compuesto por 171 imágenes que el modelo no había visto, aisladas del proceso de entrenamiento, la matriz de confusión correspondiente a esta prueba se observa en la Figura 4.10).

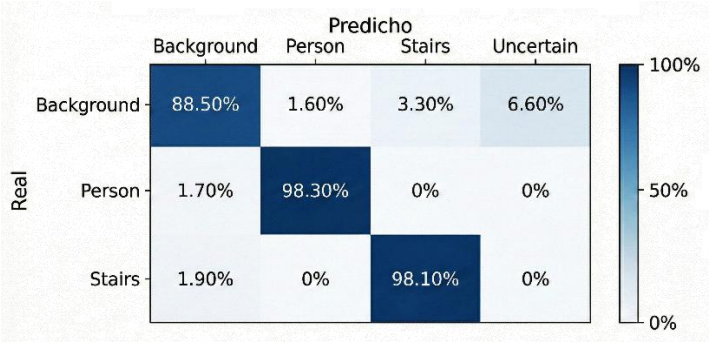

Figura 4.10. Matriz de confusión para conjunto de prueba (int8).

En esta fase, se introdujo un criterio de seguridad adicional: un umbral de confianza (θ) del 60%. Cualquier predicción cuya probabilidad máxima no superara este valor ($P < 0.60$) fue clasificada como *UNCERTAIN* (incierto).

El modelo demostró una alta estabilidad en este conjunto de datos desconocido. Las métricas globales ponderadas (*Weighted Average*) para el conjunto de prueba alcanzaron un F1-Score, Precisión y *Recall* de 0.96, manteniendo el Área bajo la Curva ROC (AUC) en 0.99 (véase Tabla 4-5). Esto confirma que la capacidad de separación entre clases se mantiene intacta fuera del entorno de entrenamiento.

Tabla 4-5. Métricas globales para conjunto de prueba (Float32 e Int8).

Exactitud	Precisión	Recall	F1-Score	Área bajo la curva ROC
Float → 0.9415 | Int8 → 0.9474	0.97	0.96	0.96	0.99

Posteriormente, se evaluó el impacto de la cuantización comparando la inferencia del modelo original (Float32) contra la versión optimizada de 8 bits (Int8). Como se detalla en la Tabla 4-6, la cuantización no degradó el rendimiento del clasificador. Por el contrario, la versión int8 presentó una ligera mejora en la exactitud global (94.74% frente a 94.15%), logrando resolver correctamente una muestra de fondo que la versión Float32 había descartado como incierta.

Tabla 4-6. Comparativa de matrices de confusión en conjunto de prueba (umbral 0.60)

	Tipo de modelo	Aciertos	Errores (Confusión)	UNCERTAIN (Incertidumbre)	Interpretación
Background	Float32	53	3	5	El modelo Float32 presenta mayor entropía (duda más) en fondos complejos.
	Int8	54	3	4	Int8 mejora la definición, reduciendo la incertidumbre sin aumentar el error.
Person	Float32 / Int8	57	1	0	Alta seguridad: El sistema nunca duda ante la presencia humana.
Stairs	Float32 / Int8	51	1	0	Alta seguridad: El sistema detecta escaleras con decisión firme.

El comportamiento observado en la columna UNCERTAIN valida la robustez del sistema para la aplicación final. Ya que en ninguna de las dos versiones el modelo marcó como incierto a una persona o unas escaleras. Si el obstáculo existe, el modelo lo clasifica.

Los casos de incertidumbre (4 en int8) se concentran exclusivamente en la clase *Background*. Esto es favorable, ya que indica que el sistema prefiere no actuar ante un fondo ambiguo en lugar de enviar una alerta falsa de obstáculo al usuario. Este umbral de confianza se puede modificar posteriormente en el código del firmware, dependiendo del comportamiento que se requiera para la implementación del sistema, por el momento, para estas pruebas se estableció en 60%.

4.3.2. Comparativa del rendimiento del modelo

Para finalizar la etapa de diseño, se evaluó la viabilidad de despliegue en el microcontrolador STM32H7. Si bien la precisión del modelo es prioritaria, en sistemas empotrados (TinyML) es igualmente importante asegurarse que el algoritmo se ejecute dentro de las estrictas restricciones de memoria y tiempo del dispositivo.

Como se introdujo en la sección 3.3.1, se comparó el rendimiento del motor intérprete estándar (*TensorFlow Lite for Microcontrollers*) frente a la compilación que ofrece Edge Impulse (*EON Compiler*). La Figura 4.11 presenta los resultados de esta comparativa, contrastando el modelo original (Float32) con las versiones optimizadas (Int8).

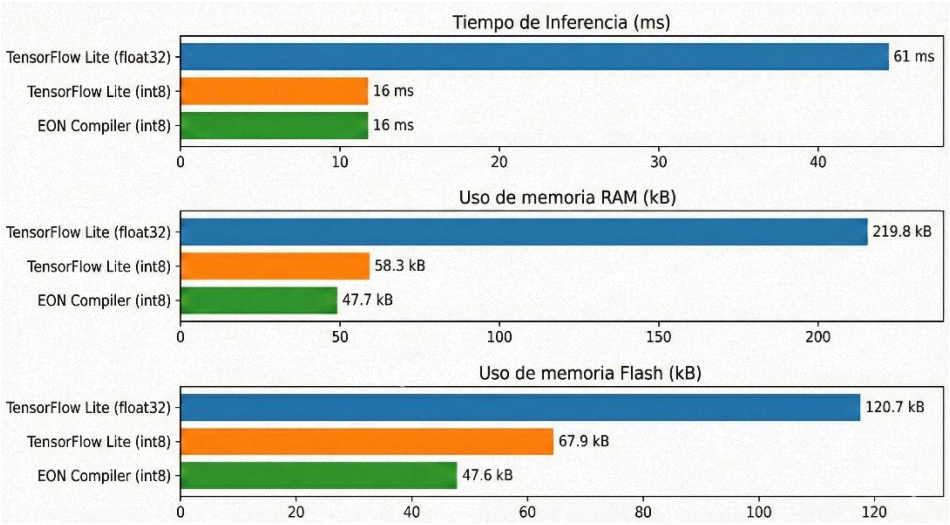

Figura 4.11. Comparativa de rendimiento en dispositivo (STM32H7, Cortex-M7).

Ambas implementaciones cuantizadas (Int8) logran un tiempo de inferencia idéntico de 16 ms. Esto representa una aceleración de casi 4x respecto al modelo Float32 original (61 ms). Este rendimiento se debe a que ambas soluciones aprovechan las instrucciones DSP del núcleo Cortex-M7 mediante la biblioteca CMSIS-NN de ARM. Con una velocidad potencial de 62 inferencias por segundo, el sistema supera holgadamente el requisito de tiempo real para la navegación humana.

Aunque la velocidad es similar, el uso del EON Compiler ofrece una ventaja en el manejo de recursos de memoria. Como se aprecia en la gráfica, reduce el consumo de memoria RAM en un 18% (47.7 kB vs 58.3 kB) y el almacenamiento Flash en casi un 30% (47.6 kB vs 67.9 kB) respecto al intérprete de TFLite. Esta reducción es una característica muy relevante para este trabajo, ya que libera espacio en memoria del microcontrolador para otras tareas del firmware, como la gestión del sensor de distancia, la lógica de control háptico y la administración de energía, minimizando el riesgo de desbordamiento de pila.

Basándose en la evidencia de las métricas de validación y prueba (secciones 4.3.1.1 y 4.3.1.2), así como en el rendimiento computacional (véase Figura 4.11), se procede a la implementación del modelo cuantizado (Int8) desplegado mediante EON Compiler. Esta configuración ofrece el equilibrio óptimo entre una alta fiabilidad de detección (>96% F1-Score) y un consumo de recursos minimizado, validando teóricamente la viabilidad del sistema antes de su construcción física.

4.4. Compilación del modelo a una biblioteca C++

Una vez validado el desempeño del modelo cuantizado, el siguiente paso es la exportación de la red neuronal desde el entorno de entrenamiento en la nube hacia el entorno de desarrollo local. Para este proyecto, de opto por el software Arduino IDE.

Para la implementación en la tarjeta Nicla Vision, se utilizó la herramienta de exportación *Arduino Library* con la tecnología EON Compiler activada (véase Figura 4.12). Este proceso no sólo descarga los pesos de la red, sino que transforma la topología del modelo en código fuente C++.

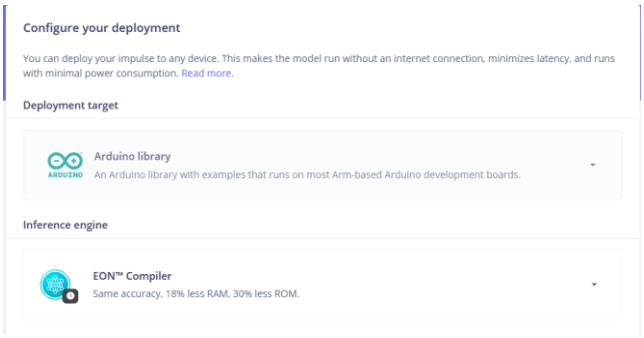

Figura 4.12. Biblioteca de Arduino generada con EON Compiler en la plataforma Edge Impulse.

El paquete generado incluye varios archivos que se integran directamente en el árbol de directorios del proyecto en Arduino IDE (véase Figura 4.13), tres de los componentes principales son:

- *SDK de Inferencia*: Es el motor de ejecución ligero que gestiona la carga de datos y el flujo de tensores.
- *Model Parameters*: Son archivos de cabecera que contienen los pesos cuantizados (int8) y la arquitectura de la red en arreglos de memoria estática, lo que permite eliminar la necesidad de *malloc* dinámico excesivo durante el arranque.
- Núcleos DSP (CMSIS-NN): Al activar esta opción, las operaciones matemáticas (convoluciones, capas densas) se reemplazan automáticamente por instrucciones vectoriales optimizadas para la arquitectura ARM Cortex-M7.

Figura 4.13. Agregar biblioteca de Arduino al árbol de directorio del proyecto.

4.5. Configuración del firmware del sistema

Trabajar con C/C++ nos permite una manipulación más eficiente de la memoria y los registros del microcontrolador, si bien en OpenMV también se puede realizar esta manipulación de componentes (véase sección 3.4.1), el intérprete que se utiliza ocupa recursos adicionales del sistema (memoria RAM y FLASH). 2) Aunque MicroPython facilita la visión por computadora, su soporte para la gestión de los dos núcleos y la comunicación entre ellos (*Remote Procedure Call, RPC*) es limitado en la implementación actual para STM32H7. El entorno Arduino, mediante la herramienta Mbed OS, proporciona una biblioteca RPC.h que permite a los núcleos Cortex-M7 y Cortex-M4 intercambiar datos y comandos de manera sincronizada.

Para habilitar este entorno, se realizó un proceso de actualización de *firmware*, reemplazando el gestor de arranque de OpenMV por el *bootloader* compatible con Arduino, permitiendo la programación a través del puerto USB.

La implementación del *firmware* utiliza la biblioteca RPC para distribuir la carga computacional y mejorar la fluidez del sistema. La arquitectura se configura bajo el esquema *Master-Slave* (véase Figura 4.14):

- El núcleo Cortex-M7 (*Main*) actúa como el procesador principal. Es responsable de ejecutar el modelo TinyML (debido a su mayor velocidad de reloj y set de instrucciones DSP).
- El núcleo Cortex-M4 (*Secondary*) se reserva para tareas de control determinista. Su función puede extenderse a la gestión de periféricos de comunicación (como la lectura UART del sensor ToF) o la generación de señales PWM para los motores hápticos, liberando al M7 de interrupciones constantes.

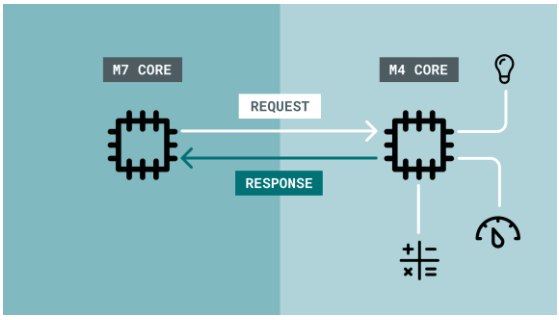

Figura 4.14. Comunicación entre los núcleos Cortex-M7 y Cortex-M4 [47].

El mecanismo de RPC establece una zona de memoria compartida que permite a ambos procesadores llamar funciones en el otro núcleo como si fueran locales. Por ejemplo, el Cortex-M7 puede enviar una variable con la clase detectada al Cortex-M4, y este último ejecuta la función de vibración correspondiente, lo que permite que el proceso de inferencia no se detenga por la latencia de los actuadores.

4.5.1. Arquitectura del algoritmo de control y configuración (Núcleo M4)

El firmware desplegado en el núcleo Cortex-M4 desempeña el rol de controlador de tiempo real y respuesta al usuario. Su lógica de operación, ilustrada en el diagrama de flujo de la Figura 4.15 y codificada en el listado adjunto, se divide en dos etapas: la secuencia de inicialización y configuración con el sensor, y el bucle de control para indicaciones al usuario.

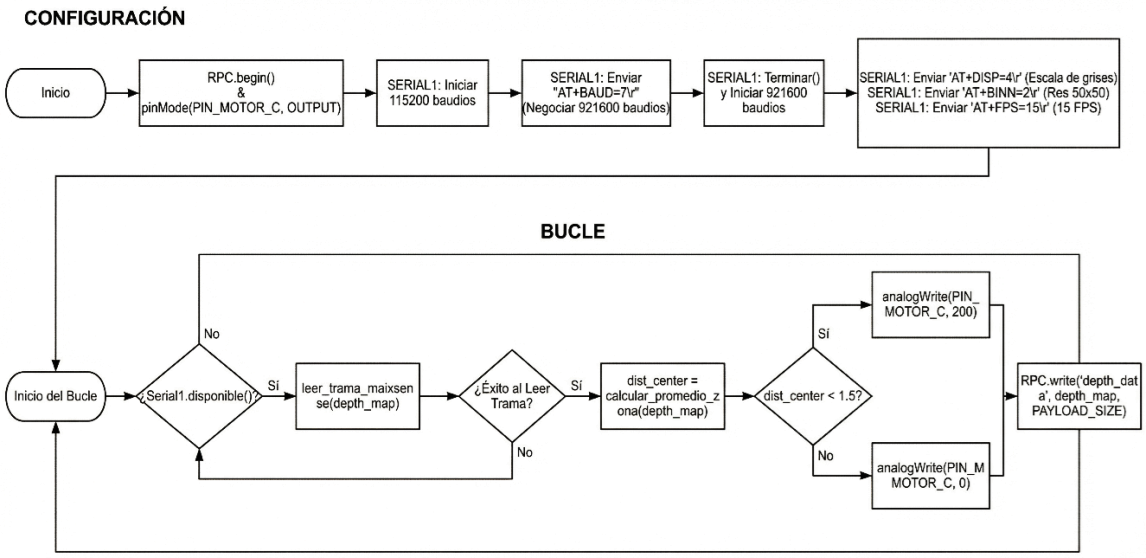

Figura 4.15. Diagrama de flujo del código en el núcleo Cortex-M4.

Etapa de configuración e inicialización

Al arrancar el sistema, la función `setup()` ejecuta una rutina de configuración para adaptar el sensor MaixSense A010 a los requerimientos de ancho de banda del proyecto. Como se observa en el bloque superior del diagrama:

- ➔ Inicialización de periféricos: Se habilita el bus de comunicación entre núcleos (`RPC.begin`) y se configuran los pines de los actuadores (motor háptico).

- ➔ Negociación de velocidad: Dado que el sensor inicia por defecto a 115 200 baudios, el microcontrolador abre el puerto serie a esta velocidad y envía el comando AT+BAUD=7. Este comando instruye al sensor para cambiar su tasa de transferencia a 921 600 baudios.

- ➔ Reinicio del puerto: El microcontrolador cierra y reinicia su propia interfaz UART a la nueva velocidad de 921 600 baudios para sincronizarse con el sensor.

- ➔ Parametrización de imagen: Una vez restablecida la comunicación a alta velocidad, se envían comandos secuenciales para configurar el formato de datos:
  - AT+DISP=4: Activa el flujo de datos a través de la interfaz UART. Cabe destacar que, aunque este modo también habilita la señal para la pantalla LCD integrada, dicho componente fue removido físicamente de la tarjeta para minimizar el consumo de energía del sistema.
  - AT+BINN=2: Activa el *binning* 2x2, reduciendo la resolución por defecto a 50×50 píxeles para disminuir la carga de datos.
  - AT+FPS=15: Fija la tasa de refresco en 19 fotogramas por segundo, asegurando estabilidad temporal.

Bucle de adquisición de datos

La función `loop()` implementa un ciclo continuo de monitoreo que prioriza la integridad física del usuario. Este proceso consta de tres fases secuenciales:

- ➔ Decodificación de trama: El sistema verifica constantemente la disponibilidad de datos en el puerto serial. Mediante la función `leer_trama_maixsense`, se busca la cabecera del protocolo y se reconstruye la matriz de profundidad (`depth_map`) de 2500 bytes.

- ➔ Lógica de evasión determinista: Inmediatamente después de obtener una trama válida, se ejecuta el algoritmo de detección de obstáculos. Se calcula la distancia promedio en la zona central de la matriz. Si este valor es inferior al umbral de 1.5 metros, el sistema activa el motor de vibración (`analogWrite(PIN, 200)`) de manera directa, sin intervención de la inteligencia artificial. Si la distancia es mayor, el motor se apaga.

- ➔ Sincronización inter-núcleo: Finalmente, la matriz de profundidad validada se envía al núcleo maestro (Cortex-M7) a través de la llamada `RPC.write`. Esto asegura que el modelo TinyML reciba únicamente datos íntegros y actualizados para realizar la clasificación de imágenes (escalera/persona) en paralelo, sin bloquear la detección de obstáculos básica.

4.5.2. Arquitectura del software para el núcleo Cortex-M7

El núcleo maestro Cortex-M7 actúa como el motor de inteligencia artificial del sistema. Su funcionamiento, representado en la Figura 4.16, opera bajo una lógica de ejecución condicional para ahorrar recursos computacionales. A diferencia del M4, que opera en un bucle continuo, el M7 condiciona la ejecución del modelo TinyML al estado del **modo inteligente**.

Gestión de estado y sincronización

El ciclo principal (`loop`) inicia verificando el estado del interruptor físico (`PIN_SWITCH_MODE`).

- ➔ Modo espera (*Low*): Si el interruptor está desactivado, el núcleo omite el bloque de procesamiento, reduciendo el consumo de energía.

➔ Modo activo (*High*): Si se activa el modo inteligente, el M7 solicita acceso a la memoria compartida mediante `RPC.read`. Esta función recupera el mapa de profundidad (`depth_data`) que el núcleo M4 ya ha validado y decodificado, almacenándolo en un buffer temporal (`raw_buffer`).

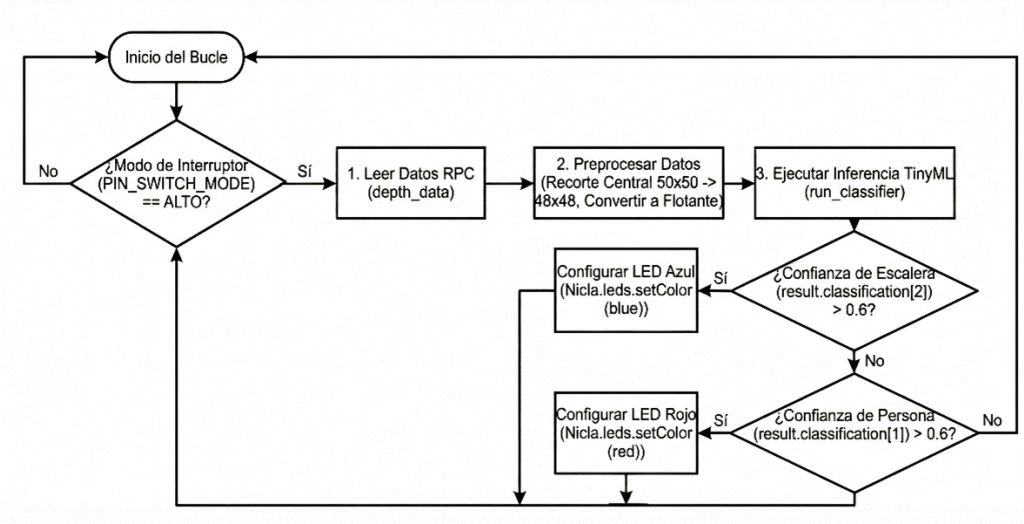

Figura 4.16. Diagrama de flujo para inferencia en el núcleo Cortex-M7.

Preprocesamiento de datos (recorte y normalización)

Una vez obtenidos los datos crudos de 50×50 píxeles, se ejecuta una etapa de preprocesamiento antes de alimentar la red neuronal. Dado que el modelo fue entrenado con una entrada de 48×48 píxeles, el algoritmo aplica un recorte central mediante un doble bucle for anidado. Este proceso descarta el primer y último píxel de cada fila y columna (índices 0 y 49), eliminando el ruido de borde característico de los sensores ToF. Simultáneamente, se realiza la conversión de tipo de dato de `uint8_t` a `float`, normalizando los valores para cumplir con los requisitos de entrada de la función `run_classifier` del SDK de Edge Impulse.

Inferencia y lógica de decisión

Con el tensor de entrada (`input_tensor`) listo, se llama al motor de inferencia optimizado por el compilador EON (`run_classifier`). El modelo retorna un arreglo de probabilidades asociado a cada clase entrenada. La interpretación de resultados sigue una lógica jerárquica basada en un umbral de confianza del 60% (0.6):

➔ Detección de escaleras (Clase 2): Se verifica prioritariamente si la probabilidad de **escalera** supera el umbral. De ser positivo, se activa el LED RGB en color azul, alertando al usuario sobre un cambio de nivel adelante.

➔ Detección de personas (Clase 1): Si no hay escaleras, se evalúa la probabilidad de **persona**. Si supera el umbral, el LED se torna rojo, indicando la presencia de un sujeto estático o dinámico en la trayectoria.

Esta estructura permite que el sistema proporcione retroalimentación solo cuando la certeza de la red neuronal es alta, minimizando las falsas alarmas y complementando la señal táctil proporcionada por el núcleo M4.

4.6. Integración de los elementos del sistema (prototipo)

La arquitectura física del sistema, detallada en el diagrama de conexión de la Figura 4.17, integra la unidad de procesamiento central, el sistema de percepción y los actuadores mediante una estrategia de energía dividida. Este diseño desacopla la alimentación lógica de la alimentación de potencia para minimizar el ruido eléctrico y mejorar la estabilidad del microcontrolador durante los picos de consumo del mini motor.

El prototipo que se propone desarrollar contempla los siguientes elementos:

- Tarjeta Nicla Vision
- Modulo ToF MaixSense A010
- 2 baterías de polímero de litio de 3.7 V y 1000 mAh
- Convertidor boost dc-dc (MT3608)
- Modulo para gestión de carga y descarga de batería (TP4056)
- Actuador háptico (mini motor de vibración dc)
- 3 interruptores (*switch*)

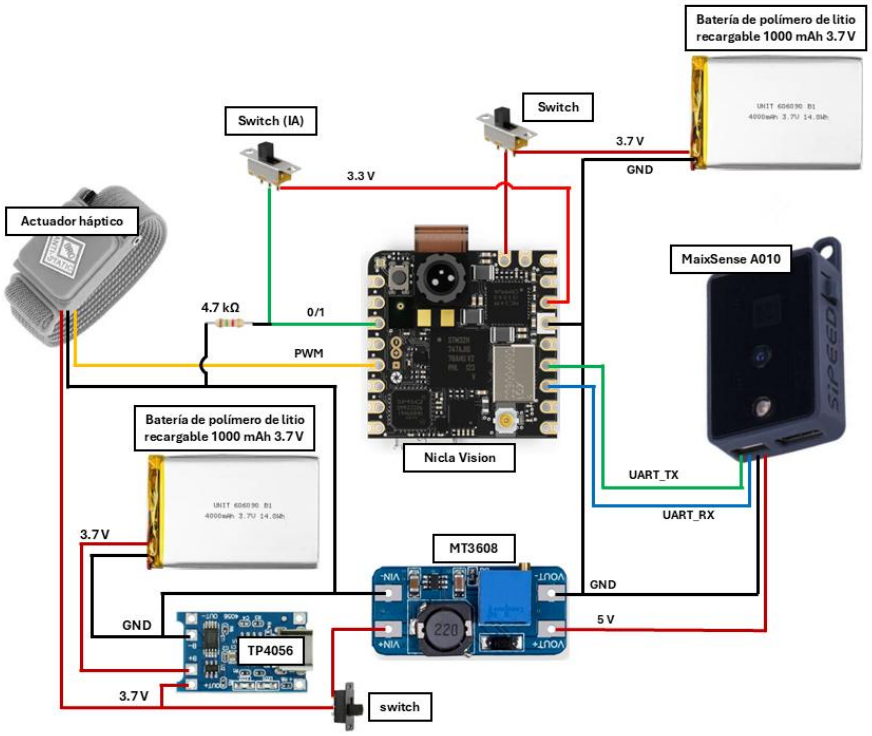

Figura 4.17. Conexión de los elementos del hardware (prototipo).

Sistema de procesamiento y percepción

El núcleo del prototipo es la tarjeta de desarrollo Nicla Vision, la cual integra el microcontrolador STM32H7.

- ➔ Interfaz de datos (UART): La adquisición de datos se realiza mediante el puerto serial. El módulo MaixSense A010 transmite las tramas de profundidad a través de sus pines TX/RX, los cuales se conectan a los pines RX/TX correspondientes de la Nicla Vision.

- ➔ Interfaz de control (GPIO/PWM): El control del actuador háptico se gestiona a través de un pin digital configurado como salida PWM. Esto permite al núcleo Cortex-M4 modular la intensidad de vibración en función de la cercanía del obstáculo.

Gestión de energía

Para asegurar la autonomía y la integridad de la señal, se implementó un esquema de alimentación con dos bancos de energía independientes utilizando baterías de polímero de litio de 3.7 V y 1000 mAh:

➔ Subsistema lógico (Batería 1): Dedicada exclusivamente a alimentar la tarjeta Nicla Vision. La gestión de carga de esta batería es manejada internamente por el circuito PMIC (*Power Management IC)* integrado en la propia tarjeta Nicla, simplificando el circuito externo. Un interruptor principal (*switch*) corta el flujo de energía hacia la tarjeta para el apagado general del sistema.

➔ Subsistema de periféricos (Batería 2): Alimenta los componentes externos y su carga es gestionada por un módulo externo TP4056, que protege contra sobrecargas y sobredescargas. La distribución de voltaje en esta etapa se divide tras pasar por un segundo interruptor de potencia:

- Línea de 5V: Un convertidor DC-DC Boost (MT3608) eleva los 3.7 V nominales de la batería a los 5 V requeridos para la operación estable del sensor MaixSense A010.
- Línea de 3.7V: El actuador háptico se alimenta directamente del voltaje de la batería, ya que su rango operativo (2.5 - 4.0 V) es compatible, evitando pérdidas de eficiencia por conversión innecesaria.

Interfaz de usuario y modos de operación

La interacción con el dispositivo se controla mediante componentes electromecánicos y visuales que se comunican directamente con la lógica del *firmware Dual-Core*:

➔ Selector de modo inteligente (*switch IA*): Un tercer interruptor conecta un pin digital de la Nicla Vision a una resistencia *pull-up* de 4.7 kΩ. El estado de este pin es monitoreado por el núcleo Cortex-M7.

- Cuando el circuito está cerrado (nivel alto), el M7 despierta y ejecuta la inferencia de la red neuronal (Modo Inteligente).
- Cuando está abierto, el M7 entra en reposo, dejando activo solo al Cortex-M4 (modo eficiente).

➔ Retroalimentación visual: No se utilizan LEDs externos. Se emplea el LED RGB integrado en la placa Nicla Vision para indicar los resultados de la clasificación de imágenes (rojo para **persona**, azul para **escalera**), gestionado directamente por el núcleo Cortex-M7.

Integración física y ensamblaje del prototipo

La Figura 4.18 presenta la vista frontal del prototipo ensamblado, demostrando la integración de los módulos sobre una placa de circuito universal perforada. Esta configuración de doble cara fue seleccionada para minimizar el área superficial del dispositivo, logrando un factor de forma compacto (tipo sándwich).

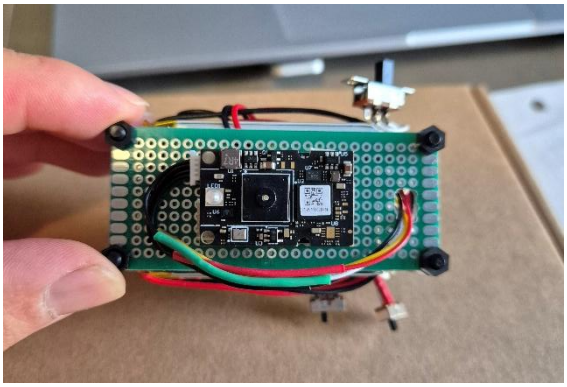

Figura 4.18. Prototipo del sistema de detección de obstáculos.

La vista frontal de la placa incluye los componentes activos que requieren línea de vista o interacción directa:

- Unidad de percepción: El sensor ToF MaixSense A010 se ubica en la parte central superior, garantizando un campo de visión (FoV) despejado para la detección de obstáculos.
- Unidad de procesamiento: La tarjeta Nicla Vision se encuentra en la parte inferior derecha, orientando su LED RGB hacia el frente.
- Gestión de carga: El módulo TP4056 se sitúa en la parte inferior izquierda, permitiendo la conexión del cable de carga USB-C sin interferir con los sensores.

Para optimizar el espacio y equilibrar el centro de gravedad del dispositivo, los componentes de potencia se alojaron en la cara posterior de la placa base:

- Bancos de energía: Las dos baterías de polímero de litio de 1000 mAh se fijaron en paralelo a la superficie posterior. Esta disposición actúa como un contrapeso natural, mejorando la estabilidad del dispositivo.
- Regulación de voltaje: El convertidor *DC-DC Boost* (MT3608) se instaló en el espacio entre las baterías y la placa, conectándose a través de perforaciones con el módulo TP4056 y el sensor MaixSense situados en la cara opuesta.

Esta arquitectura de montaje permite aislar físicamente las líneas de alta corriente de las líneas de señal digital, reduciendo la posibilidad de interferencias electromagnéticas en el bus de datos UART.

4.7. Pruebas en ambiente controlado

Se llevó a cabo una serie de pruebas experimentales en las instalaciones del edificio de posgrados (FCE6) de la Facultad de Ciencias de la Electrónica de la BUAP. Las pruebas de detección de personas se realizaron dentro del laboratorio SLED y pasillos, mientras que la detección de escaleras se ejecutó en los cubos de escaleras del mismo edificio, en ambos casos bajo iluminación artificial (LED).

Para comprobar la precisión de las mediciones, se estableció una metodología de comparación directa utilizando una cinta métrica para medir la distancia real del objeto frente a la alerta generada por el dispositivo.

Rango de detección y ángulo de incidencia

Se evaluó la capacidad del modelo para clasificar correctamente las clases objetivo a diferentes distancias. Los resultados promedio obtenidos se detallan a continuación:

Detección de personas: El sistema logró identificar consistentemente la presencia humana (silueta completa) a una distancia máxima promedio de 1.80 metros. Dado el tamaño relativo de la silueta humana respecto a la resolución del sensor (50x50), la clasificación se mantuvo robusta sin necesidad de ajustes posturales por parte del usuario.

Detección de escaleras: La distancia efectiva de detección se registró en aproximadamente 1.50 metros. Sin embargo, se observó dificultad de detección por el ángulo de visión. Para lograr una clasificación correcta, fue necesario inclinar ligeramente el prototipo hacia el suelo, ajustando el ángulo de incidencia para que los escalones ocuparan una porción significativa del campo de visión del sensor.

Respuesta ante materiales y geometrías complejas

Durante las pruebas de detección de obstáculos, se confirmaron comportamientos inherentes a la física de los sensores de Tiempo de Vuelo (ToF), los cuales presentaron limitaciones ante ciertas propiedades de los materiales:

Superficies especulares y transparentes: Elementos como puertas de cristal o ventanales grandes fueron interpretados erróneamente como espacio vacío o distancias infinitas, debido a que la luz infrarroja atraviesa el material sin rebotar hacia el receptor.

Materiales absorbentes: Objetos recubiertos con telas oscuras o mate (como sillas y cubiertas) mostraron una tasa de detección reducida, atribuible a la absorción de la longitud de onda IR, lo que debilita la señal de retorno.

Objetos pequeños y complejos: Se observo que objetos con geometrías irregulares o dimensiones reducidas requieren un enfoque directo y una distancia menor para ser detectados por la matriz de 50x50 píxeles.

Pruebas dinámicas y velocidad de sujeto

Se analizó el desempeño del sistema ante sujetos en movimiento. Para velocidades de caminata natural (aprox. 1.0 - 1.4 m/s), el sistema detectó y clasificó a las personas sin latencia perceptible. No obstante, al aumentar la velocidad a paso de carrera, se registraron picos de inestabilidad en la detección. Esto sugiere que la tasa de refresco configurada (19 FPS) y el tiempo de inferencia acumulado podrían estar cerca del límite temporal para objetos en movimiento rápido, requiriendo pruebas adicionales para caracterizar completamente este fenómeno.

Influencia de las condiciones de iluminación

El desempeño del sensor MaixSense A010 mostró una correlación directa con la iluminación ambiental:

Iluminación artificial (LED): Bajo la luz de las lámparas del laboratorio, el sensor operó de forma normal, indicando que el espectro de la iluminación LED no genera interferencia significativa en la banda infrarroja del dispositivo.

Entornos de alta luminosidad (luz solar): En zonas próximas a ventanales o entradas con incidencia directa de luz solar, se observó un aumento considerable en el ruido de la medición, resultando en lecturas fluctuantes debido a la saturación por la componente IR de la luz solar.

Entornos de baja luminosidad: Por el contrario, las pruebas realizadas en condiciones de oscuridad total (luces apagadas) proporcionaron el mejor desempeño del sistema. La ausencia de ruido lumínico ambiental mejoro la relación señal-ruido (SNR) del emisor IR del sensor, lo que confirmaría que el entorno óptimo son los espacio con poca luz.

Capítulo 5

5. Conclusiones

El desarrollo de este trabajo de investigación ha culminado con la implementación exitosa de un prototipo de asistencia háptica funcional, logrando los objetivos planteados al inicio del proyecto, sobre todo el que se refiere a la aplicación de técnicas de aprendizaje automático integrado (TinyML) en sistemas empotrados con recursos de cómputo limitados, permitiendo solventar problemáticas de accesibilidad en personas con discapacidad visual. A través de un diseño metodológico detallado, que abarcó desde la selección de hardware hasta la optimización de algoritmos de visión por computadora, con esto, se ha conseguido un sistema capaz de operar en tiempo real sin depender de conectividad externa.

5.1. Conclusiones generales

El análisis de los resultados experimentales permite corroborar la viabilidad técnica del procesamiento de redes neuronales convolucionales en microcontroladores de la familia STM32H7. Los tiempos de inferencia obtenidos, situados en los 16 milisegundos, demuestran que la arquitectura Cortex-M7, potenciada por el compilador EON y las instrucciones DSP, posee la capacidad de cómputo necesaria para ejecutar tareas de visión artificial complejas con una latencia mínima. Este hallazgo es significativo pues desafía la noción convencional de que el reconocimiento de imágenes requiere procesadores de alto consumo energético o enlaces a la nube, estableciendo que la inteligencia artificial empotrada es una alternativa confiable y autónoma para dispositivos portátiles (*wearables*).

Asimismo, se ha comprobado que la estrategia de cuantización es un factor relevante para el despliegue eficiente de modelos en dispositivos con recursos de cómputo limitados. La conversión del modelo de precisión flotante de 32 bits a enteros de 8 bits no solo resultó en una reducción del uso de memoria RAM cercana al 18%, sino que mantuvo la métrica de desempeño F1-Score por encima del 96%. Esto confirma que, para aplicaciones de detección de obstáculos grandes como escaleras y personas, la pérdida de precisión inherente a la cuantización es mínima en comparación con el rendimiento en velocidad y gestión de memoria.

Un aspecto importante del diseño fue la implementación de una arquitectura de *firmware* asimétrica que aprovecha la dualidad de los núcleos del microcontrolador. Al dividir las tareas de adquisición de datos y gestión de respuestas en el núcleo Cortex-M4 de las tareas de inferencia en el núcleo Cortex-M7, se logró construir un sistema tolerante a fallos. Esta configuración permite que la funcionalidad básica de evasión de colisiones por proximidad permanezca operativa de manera constante, independientemente de la carga de trabajo o el estado del modelo TinyML.

Finalmente, la integración del módulo ToF MaixSense A010 hizo posible utilizar mapas de profundidad de baja resolución para esta aplicación específica. A pesar de trabajar con una matriz reducida de 50×50 píxeles, la información espacial resultó suficiente para la identificación de estructuras relevantes en el entorno de navegación. Además, este enfoque técnico ofrece una ventaja en términos de privacidad, ya que, al procesar únicamente datos de distancia y no imágenes RGB convencionales, el dispositivo no captura rostros ni detalles sensibles del entorno, alineándose con los principios de diseño ético y privacidad por diseño.

5.2. Oportunidades de mejora y trabajo a futuro

A partir de los hallazgos experimentales y las limitaciones identificadas durante las pruebas en el laboratorio, se han detectado cuatro áreas que se pueden optimizar en un futuro:

1. Interfaz háptica y ergonomía del dispositivo
2. Migración a arquitecturas MCU + NPU
3. Actualización sensor ToF (Serie VL53L9)
4. Diseño de PCB a medida

Estas propuestas no solo buscan mejorar el desempeño actual, sino evolucionar el sistema hacia un posible producto comercial.

1. Interfaz háptica y ergonomía del dispositivo

De acuerdo con el alcance de esta tesis, el diseño actual se centró en la validación funcional del algoritmo de detección. Sin embargo, la interacción física con el usuario requiere una investigación profunda en el campo de la psicofísica táctil. La percepción de las vibraciones varía drásticamente según la ubicación del actuador (mano, pecho, cabeza) y la presión de contacto. Asimismo, se propone desarrollar un estudio ergonómico para diseñar una interfaz portátil que sea capaz de transmitir indicaciones sin causar fatiga sensorial.

2. Migración a arquitecturas MCU + NPU

El modelo de red neuronal diseñado en este trabajo fue pensado estructuralmente utilizando filtros múltiplos de 16 y 32, logrando un rendimiento excelente en el núcleo Cortex-M7. No obstante, la evolución lógica del sistema apunta hacia la migración a microcontroladores de nueva generación equipados con Unidades de Procesamiento Neuronal (NPU), como la arquitectura basada en Arm Cortex-M55 y el acelerador Ethos-U55. Esta actualización de hardware permitiría reducir los tiempos de inferencia a niveles de microsegundos, liberando recursos para implementar técnicas de fusión de sensores. Esto abriría la puerta a integrar una cámara RGB compacta al sistema. En esta configuración híbrida, el sensor de Tiempo de Vuelo (ToF) mantendría el rol de detección de obstáculos en modo de bajo consumo, mientras que la cámara y la NPU se activarían para resolver

las limitaciones ópticas detectadas en este trabajo, tales como la absorción de luz infrarroja en telas negras, la refracción en superficies de vidrio o la saturación del sensor bajo luz solar directa en exteriores.

3. Actualización del sensor ToF (Serie VL53L9)

Durante el desarrollo se identificaron restricciones en la configuración del módulo MaixSense A010 debido a su ecosistema de software cerrado. Se propone la sustitución de este componente por sensores de última generación de STMicroelectronics, específicamente la serie VL53L9 (sucesora de la VL53L8), cuya disponibilidad en el mercado se proyecta para mediados de 2026. Este nuevo sensor promete integrar una matriz multizona de alta resolución (aproximándose a las 50x50 zonas) con un amplio soporte de documentación y bibliotecas. A diferencia del sensor actual, la serie VL53 permitiría un mayor control sobre los parámetros de emisión y recepción, mejorando el alcance y la inmunidad al ruido ambiental.

4. Diseño de PCB a medida

Para pasar de un prototipo de desarrollo a un dispositivo final, es necesario dejar a un lado el uso de tarjetas como la Nicla Vision. Aunque esta plataforma fue ideal para la etapa de validación, integra múltiples sensores y circuitos auxiliares que no son necesarios para la aplicación especifica que se está trabajando, ocupando espacio y consumiendo energía innecesariamente. El siguiente paso es el diseño de una Placa de Circuito Impreso (PCB) dedicada. Esto permitirá integrar exclusivamente el microcontrolador objetivo, el sensor ToF, los drivers de los actuadores y demás componentes necesarios.

Bibliografía

[1] V. Janapa Reddi, *Introduction to Machine Learning Systems*. US: Harvard University, 2025. [En línea]. Disponible en: https://mlsysbook.ai/assets/downloads/Machine-Learning-Systems.pdf

[2] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, y S. Han, "Tiny Machine Learning: Progress and Futures [Feature]", *IEEE Circuits Syst. Mag.*, vol. 23, n.º 3, pp. 8-34, 2023, doi: 10.1109/MCAS.2023.3302182.

[3] D. Situnayake y J. Plunkett, *AI at the Edge*. O'Reilly Media, Inc., 2023. Accedido: 13 de enero de 2026. [En línea]. Disponible en: https://learning.oreilly.com/library/view/ai-at-the/9781098120191/

[4] E. White, *Making Embedded Systems, 2nd Edition*. O'Reilly Media, Inc., 2024. Accedido: 13 de enero de 2026. [En línea]. Disponible en: https://learningoreilly.bibliotecabuap.elogim.com/library/view/making-embedded-systems/9781098151539/

[5] ams OSRAM, «Understanding time-of-flight sensing, White Paper». Accedido: 13 de enero de 2026. [En línea]. Disponible en: https://www.mouser.com/pdfDocs/understandingtime-of-flightsensing_disti2024_final.pdf

[6] M. Hansard, S. Lee, O. Choi, y R. Horaud, *Time-of-Flight Cameras: Principles, Methods and Applications*. en SpringerBriefs in Computer Science. London: Springer, 2013. doi: 10.1007/978-1-4471-4658-2.

[7] Naciones Unidas, Departamento de Asuntos Económicos y Sociales, "Transformar nuestro mundo: la Agenda 2030 para el Desarrollo Sostenible". Accedido: 25 de noviembre de 2025. [En línea]. Disponible en: https://sdgs.un.org/2030agenda

[8] Organización Mundial de la Salud, "Ceguera y discapacidad visual". Accedido: 18 de diciembre de 2025. [En línea]. Disponible en: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

[9] INEGI, "Censo de Población y Vivienda (CPV) 2020, Discapacidad". Accedido: 18 de diciembre de 2025. [En línea]. Disponible en: https://www.inegi.org.mx/temas/discapacidad/

[10] INEGI, "Panorama sociodemográfico de Puebla. Censo de Población y Vivienda 2020". Accedido: 18 de diciembre de 2025. [En línea]. Disponible en: https://www.inegi.org.mx/app/biblioteca/ficha.html?upc=702825197940

[11] N. Lakovic, M. Brkic, B. Batinic, J. Bajic, V. Rajs, y N. Kulundzic, "Application of low-cost VL53L0X ToF sensor for robot environment detection", en *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, East Sarajevo, Bosnia and Herzegovina: IEEE, mar. 2019, pp. 1-4. doi: 10.1109/INFOTEH.2019.8717779.

[12] R. M. Jans, A. S. Green, y L. J. Koerner, "Characterization of a Miniaturized IR Depth Sensor With a Programmable Region-of-Interest That Enables Hazard Mapping Applications", *IEEE Sens. J.*, vol. 20, n.º 10, pp. 5213-5220, may 2020, doi: 10.1109/JSEN.2020.2971595.

[13] X. Cheng y M. Mamishev, "Characterization of a Wearable LiDAR Sensor for Social Distancing", en *2022 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*, Arad, Romania: IEEE, may 2022, pp. 115-120. doi: 10.1109/GlobConET53749.2022.9872387.

[14] C. Sifferman, D. Mehrotra, M. Gupta, y M. Gleicher, "Geometric Calibration of Single-Pixel Distance Sensors", *IEEE Robot. Autom. Lett.*, vol. 7, n.º 3, pp. 6598-6605, jul. 2022, doi: 10.1109/LRA.2022.3176453.

[15] Z. Li, J. H. Marsh, y L. Hou, "High Precision Laser Ranging Based on STM32 Microcontroller", en *2020 International Conference on UK-China Emerging Technologies (UCET)*, Glasgow, United Kingdom: IEEE, ago. 2020, pp. 1-4. doi: 10.1109/UCET51115.2020.9205377.

[16] A. Fasolino *et al.*, "Multiclass Object Classification Using Ultra-Low Resolution Time-of-Flight Sensors", *IEEE Sens. Lett.*, vol. 8, n.º 10, pp. 1-4, oct. 2024, doi: 10.1109/LSENS.2024.3467165.

[17] V. Niculescu, T. Polonelli, M. Magno, y L. Benini, "NanoSLAM: Enabling Fully Onboard SLAM for Tiny Robots", *IEEE Internet Things J.*, vol. 11, n.º 8, pp. 13584-13607, abr. 2024, doi: 10.1109/JIOT.2023.3339254.

[18] J. Pleterski, G. Škulj, C. Esnault, J. Puc, R. Vrabič, y P. Podržaj, "Miniature Mobile Robot Detection Using an Ultralow-Resolution Time-of-Flight Sensor", *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1-9, 2023, doi: 10.1109/TIM.2023.3318710.

[19] D. P. Pau, G. Fiorenza, y M. Garzola, "Detecting Mice Depth Shapes with Deployable Tiny Neural Networks in Small Cages", en *2024 Zooming Innovation in Consumer Technologies Conference (ZINC)*, Novi Sad, Serbia: IEEE, may 2024, pp. 72-77. doi: 10.1109/ZINC61849.2024.10579406.

[20] A. Fasolino *et al.*, "Object Classification using Ultra Low Resolution Time-of-Flight Sensor and Tiny Convolutional Neural Network", en *2024 IEEE Sensors Applications Symposium (SAS)*, Naples, Italy: IEEE, jul. 2024, pp. 1-6. doi: 10.1109/SAS60918.2024.10636597.

[21] S. E. Arslan, "Tiny machine learning model for obstacle detection with multi-zone time of flight sensors", *Recent Adv. Sci. Eng.*, pp. 9-14, 2023, doi: 10.14744/rase.2023.0002.

[22] S.-H. Chen, H.-Y. Chen, Y.-M. Cheng, R.-Y. Li, P.-X. Wang, y Y.-H. Teng, "DeepEye: Design and Implementation of Embedded System for Environment Perception and Navigation of Visually Impaired People", en *2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, PingTung, Taiwan: IEEE, jul. 2023, pp. 759-760. doi: 10.1109/ICCE-Taiwan58799.2023.10226719.

[23] F. Barontini, M. G. Catalano, L. Pallottino, B. Leporini, y M. Bianchi, "Integrating Wearable Haptics and Obstacle Avoidance for the Visually Impaired in Indoor Navigation: A User-Centered Approach", *IEEE Trans. Haptics*, vol. 14, n.º 1, pp. 109-122, ene. 2021, doi: 10.1109/TOH.2020.2996748.

[24] I.-H. Hsieh, H.-C. Cheng, H.-H. Ke, H.-C. Chen, y W.-J. Wang, "Outdoor walking guide for the visually-impaired people based on semantic segmentation and depth map", en *2020 International Conference on Pervasive Artificial Intelligence (ICPAI)*, Taipei, Taiwan: IEEE, dic. 2020, pp. 144-147. doi: 10.1109/ICPAI51961.2020.00034.

[25] E. B. Kaiser y M. Lawo, "Wearable Navigation System for the Visually Impaired and Blind People", en *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, Shanghai, China: IEEE, may 2012, pp. 230-233. doi: 10.1109/ICIS.2012.118.

[26] Conexión CINVESTAV, "Lentes Inteligentes - SmartOjo", Lentes Inteligentes. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://conexion.cinvestav.mx/Publicaciones/lentes-inteligentes

[27] 7Sense, "SuperBrain 1", 7Sense. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://7sense.ee/

[28] LUMEN, "Lentes para asistencia visual", .lumen. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: https://www.dotlumen.com

[29] STMicroelectronics, "UM3183, User Manual". Accedido: 13 de enero de 2026. [En línea]. Disponible en: https://www.st.com/en/imaging-and-photonics-solutions/time-of-flight-sensors.html
[30] STMicroelectronics, "AN5894, Application Note". Accedido: 13 de enero de 2026. [En línea]. Disponible en: https://www.st.com/en/imaging-and-photonics-solutions/time-of-flight-sensors.html
[31] OPNOUS, "Manual del producto, IoT ToF". Accedido: 13 de enero de 2026. [En línea]. Disponible en: http://www.opnous.com/product?cat=1&pid=23&lang=en
[32] Sipeed, "MaixSense-A010 - Sipeed Wiki". Accedido: 13 de enero de 2026. [En línea]. Disponible en: https://wiki.sipeed.com/hardware/en/maixsense/maixsense-a010/maixsense-a010.html#
[33] STMicroelectronics, "AN5897, Application Note". Accedido: 13 de enero de 2026. [En línea]. Disponible en: https://www.st.com/en/imaging-and-photonics-solutions/time-of-flight-sensors.html
[34] P. Agarwal y A. Bijalwan, *Edge Artificial Intelligence*. Wiley-Scrivener, 2025. Accedido: 13 de enero de 2026. [En línea]. Disponible en: https://learningoreilly.bibliotecabuap.elogim.com/library/view/edge-artificial-intelligence/9781394355006/
[35] Edge Impulse, "Edge AI Fundamentals", Edge Impulse Documentation. Accedido: 27 de diciembre de 2025. [En línea]. Disponible en: https://docs.edgeimpulse.com/knowledge/courses/edge-ai-fundamentals
[36] S. Salerno, *Tiny Machine Learning Quickstart: Machine Learning for Arduino Microcontrollers*. Apress, 2025. Accedido: 13 de enero de 2026. [En línea]. Disponible en: https://learningoreilly.bibliotecabuap.elogim.com/library/view/tiny-machine-learning/9798868812941/
[37] P. Warden y D. Situnayake, *TinyML*. O'Reilly Media, Inc., 2019. Accedido: 24 de diciembre de 2025. [En línea]. Disponible en: https://learning.oreilly.com/library/view/tinyml/9781492052036/
[38] TensorFlow, "TensorFlow Lite para microcontroladores", TensorFlow. Accedido: 24 de diciembre de 2025. [En línea]. Disponible en: https://www.tensorflow.org/lite/microcontrollers?hl=es-419
[39] Google, "Colab | Google for Developers". Accedido: 17 de enero de 2026. [En línea]. Disponible en: https://developers.google.com/colab
[40] Keras, "Keras documentation: Developer guides". Accedido: 17 de enero de 2026. [En línea]. Disponible en: https://keras.io/guides/
[41] TensorFlow, "TensorFlow Documentation", TensorFlow. Accedido: 17 de enero de 2026. [En línea]. Disponible en: https://www.tensorflow.org/?hl=es-419
[42] Edge Impulse, "Edge Impulse - The Leading Edge AI Platform". Accedido: 17 de enero de 2026. [En línea]. Disponible en: https://www.edgeimpulse.com
[43] "EON Compiler", Edge Impulse Documentation. Accedido: 17 de enero de 2026. [En línea]. Disponible en: https://docs.edgeimpulse.com/studio/projects/deployment/eon-compiler
[44] "openmv/docs/firmware.md at master · openmv/openmv", GitHub. Accedido: 17 de mayo de 2025. [En línea]. Disponible en: https://github.com/openmv/openmv/blob/master/docs/firmware.md
[45] "class SPIDisplay – SPI Display Driver — MicroPython 1.24 documentation". Accedido: 18 de mayo de 2025. [En línea]. Disponible en: https://docs.openmv.io/library/omv.display.spidisplay.html

[46] "Overview — MicroPython 1.24 documentation". Accedido: 18 de mayo de 2025. [En línea]. Disponible en: https://docs.openmv.io/index.html

[47] Arduino, "Guide to GIGA R1 Dual Cores | Arduino Documentation". Accedido: 20 de enero de 2026. [En línea]. Disponible en: https://docs.arduino.cc/tutorials/giga-r1-wifi/giga-dual-core/#using-the-rpc-library-with-micropython

[48] Arduino, "Universal Asynchronous Receiver-Transmitter (UART) | Arduino Documentation". Accedido: 20 de enero de 2026. [En línea]. Disponible en: https://docs.arduino.cc/learn/communication/uart/

Apéndices

Apéndice A. Configuraciones del sensor VL53L8CH

A partir de los requerimientos del proyecto y las especificaciones técnicas del sensor VL53L8CH, se definieron diversas configuraciones paramétricas. La Tabla A-1 detalla todas las combinaciones posibles tomando como estáticos tres parámetros: 8x8 zonas, frecuencia de 15 Hz y rango de medición de 2 metros. Se señalan en verde aquellas que se acercan más a la distancia de referencia.

En la Tabla A-2 se detallan los resultados obtenidos con las 9 combinaciones que, teóricamente, ofrecen el rendimiento óptimo para los objetivos de esta investigación.

Tabla A-1. Configuración de parámetros disponibles para el sensor VL53L8CH.

Configuración del sensor en modo 8x8 | 15 Hz			
Número de contenedores (bins)	**Factor de agrupamiento**	**Tamaño del contenedor (mm)**	**Ventana del contenedor (mm)**
8	1	37.5	300
8	2	75	600
8	3	112.5	900
8	4	150	1200
8	5	187.5	1500
8	6	225	1800
8	7	262.5	2100
8	8	300	2400
12	1	37.5	450
12	2	75	900
12	3	112.5	1350
12	4	150	1800
12	5	187.5	2250
12	6	225	2700
12	7	262.5	3150
12	8	300	3600
18	1	37.5	675
18	2	75	1350
18	3	112.5	2025
18	4	150	2700
18	5	187.5	3375
18	6	225	4050
18	7	262.5	4725
18	8	300	5400

Tabla A-2. Resultados obtenidos de las 9 mejores configuraciones del sensor VL53L8CH.

No.	Configuración del sensor	Zonas	Valores	Distancia (mm)	Error (mm)
1	8x8 | 8 bins | x5 | 187.5 mm | 1500 mm	Internas	Promedio	2005	5
			Mínimo	1964	36
			Máximo	2042	42
		Esquinas	Promedio	2020	20
			Mínimo	1990	10
			Máximo	2044	44
2	8x8 | 8 bins | x6 | 225 mm | 1800 mm	Internas	Promedio	2006	6
			Mínimo	1966	34
			Máximo	2041	41
		Esquinas	Promedio	2021	21
			Mínimo	1980	20
			Máximo	2046	46
3	8x8 | 8 bins | x7 | 262.5 mm | 2100 mm	Internas	Promedio	2005	5
			Mínimo	1972	28
			Máximo	2041	41
		Esquinas	Promedio	2018	18
			Mínimo	1987	13
			Máximo	2048	48
4	8x8 | 8 bins | x8 | 300 mm | 2400 mm	Internas	Promedio	2004	4
			Mínimo	1972	28
			Máximo	2039	39
		Esquinas	Promedio	2018	18
			Mínimo	1983	17
			Máximo	2050	50
5	8x8 | 12 bins | x4 | 150 mm | 1800 mm	Internas	Promedio	2003	3
			Mínimo	1965	35
			Máximo	2044	44
		Esquinas	Promedio	2017	17
			Mínimo	1980	20
			Máximo	2048	48
6	8x8 | 12 bins | x5 | 187.5 mm | 2250 mm	Internas	Promedio	2003	3
			Mínimo	1963	37
			Máximo	2036	36
		Esquinas	Promedio	2018	18
			Mínimo	1988	12
			Máximo	2045	45
7	8x8 | 12 bins | x6 | 225 mm | 2700 mm	Internas	Promedio	2005	5
			Mínimo	1871	129
			Máximo	2041	41
		Esquinas	Promedio	2019	19
			Mínimo	1981	19
			Máximo	2051	51
8	8x8 | 18 bins | x3 | 112.5 mm | 2025 mm	Internas	Promedio	2005	5
			Mínimo	1964	36
			Máximo	2044	44
		Esquinas	Promedio	2019	19
			Mínimo	1985	15
			Máximo	2044	44
9	8x8 | 18 bins | x4 | 150 mm | 2700 mm	Internas	Promedio	2004	4
			Mínimo	1969	31
			Máximo	2041	41
		Esquinas	Promedio	2020	20
			Mínimo	1991	9
			Máximo	2051	51

Apéndice B. Código para captura de imágenes de profundidad

Código completo en MicroPython para guardar un conjunto de imágenes de profundidad en la memoria de la tarjeta OpenMV Cam H7 Plus, la comunicación entre la tarjeta y el módulo MaixSense A010 se realizo por medio de UART, enviando comandos AT para configurar del sensor ToF.

```
# MaixSense A010 ToF Dataset Capture - OpenMV H7 Plus
# RESOLUCIÓN: 50x50 (Binning 2x2)

import pyb, time, struct, gc, image, os
import display

# ================= CONFIGURACIÓN =================
UART_ID         = 3
UART_BAUD       = 115200  # Velocidad ESTANDAR (Suficiente para 50x50)
DESIRED_FPS     = 4

SHUTTER_PIN_NAME = "P1"

FRAME_HEAD      = b'\x00\xFF'
FRAME_TAIL      = 0xDD
META_LEN        = 16
BUF_LIMIT       = 32000

# ================= INICIALIZACIÓN =================
print("--- Iniciando Capturadora ToF (50x50) ---")

# 1. LCD
try:
    lcd = display.SPIDisplay()
    lcd.write(image.Image(50, 50, image.RGB565).draw_string(2, 20, "50x50 Mode",
color=(255,255,255), scale=1))
except:
    lcd = None

# 2. Botón y LEDs
shutter_btn = pyb.Pin(SHUTTER_PIN_NAME, pyb.Pin.IN, pyb.Pin.PULL_UP)
led_grn = pyb.LED(2)
led_blu = pyb.LED(3)

# 3. Configuración UART
uart = pyb.UART(UART_ID, UART_BAUD, timeout_char=1000)

def send_cmd(cmd):
    uart.write(cmd)
    time.sleep_ms(50)

print("Configurando sensor...")
send_cmd(b'AT+DISP=4\r')       # 8-bit grayscale
send_cmd(b'AT+BINN=2\r')       # Modo 50x50 pixeles
send_cmd(b'AT+FPS=%d\r' % DESIRED_FPS)
```

```
# ================ GESTIÓN DE ARCHIVOS ================
def get_next_filename():
    i = 0
    while True:
        filename = "{:01d}.bmp".format(i)
        try:
            os.stat(filename)
            i += 1
        except OSError:
            return filename

# ================ BUCLE PRINCIPAL ================
buf = bytearray()
# Creamos imagen base de 50x50
img = image.Image(50, 50, image.GRAYSCALE)
last_toggle = time.ticks_ms()

while True:
    # --- 1. INTERFAZ ---
    if time.ticks_diff(time.ticks_ms(), last_toggle) > 500: # Parpadeo más rápido (0.5s)
        led_blu.toggle()
        last_toggle = time.ticks_ms()

    # Botón obturador
    if shutter_btn.value() == 0:
        led_blu.off()
        led_grn.on()

        fname = get_next_filename()
        print("Guardando: " + fname)
        img.save(fname)

        time.sleep_ms(150)
        led_grn.off()
        while shutter_btn.value() == 0: time.sleep_ms(10)

    # --- 2. SENSOR ToF COMUNICACIÓN (UART) ---
    if uart.any():
        buf.extend(uart.read())

    start_idx = buf.find(FRAME_HEAD)

    if start_idx == -1:
        if len(buf) > BUF_LIMIT: buf = bytearray()
        continue

    if start_idx > 0:
        buf = buf[start_idx:]

    if len(buf) < 4: continue

    data_len = struct.unpack_from('<H', buf, 2)[0]
```

```
    pkt_len = 4 + data_len + 2

    if len(buf) < pkt_len: continue

    frame = buf[:pkt_len]
    buf = buf[pkt_len:]

    if frame[-1] != FRAME_TAIL: continue

    # Decodificar
    # En modo 50x50, el payload debe ser de 2500 bytes
    payload = frame[20:20 + 2500]

    if len(payload) == 2500:
        # Llenar imagen 50x50
        for i in range(2500):
            # i % 50 = x, i // 50 = y
            img.set_pixel(i % 50, i // 50, payload[i])

        # Actualizar LCD
        # (La LCD escalará automáticamente la imagen de 50px a pantalla completa)
        if lcd:
            lcd.write(img, hint=image.CENTER | image.SCALE_ASPECT_KEEP)

    gc.collect()
```

Código desarrollado en OpenMV IDE 4.7.0

Apéndice C. Arquitectura del modelo CNN.

Arquitectura del modelo

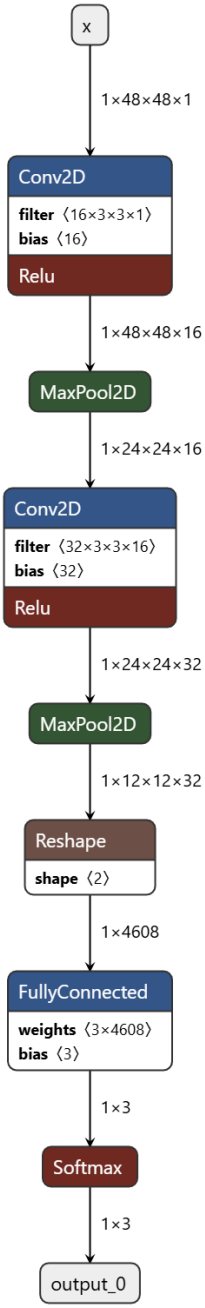

Figura C.1. Arquitectura del modelo CNN del sistema.

Apéndice D. Graficas de exactitud y pérdida para CNN (30 épocas)

Después de entrenar el modelo CNN con 100 épocas, se observó que la gráfica de exactitud y la función de perdida se estabilizaron cerca de la época 30, por lo cual, se volvió a entrenar el modelo, esta vez con sólo 30 épocas, obteniendo una exactitud de 94.7% y una pérdida de 0.19. Comparado con los resultados del primer entrenamiento (exactitud: 96.2% y pérdida: 0.12), se observa que la diferencia es mínima, pero ahorrando tiempo de entrenamiento (~70% menos).

Los resultados del segundo entrenamiento (30 épocas) se muestran en la Figura D.2.

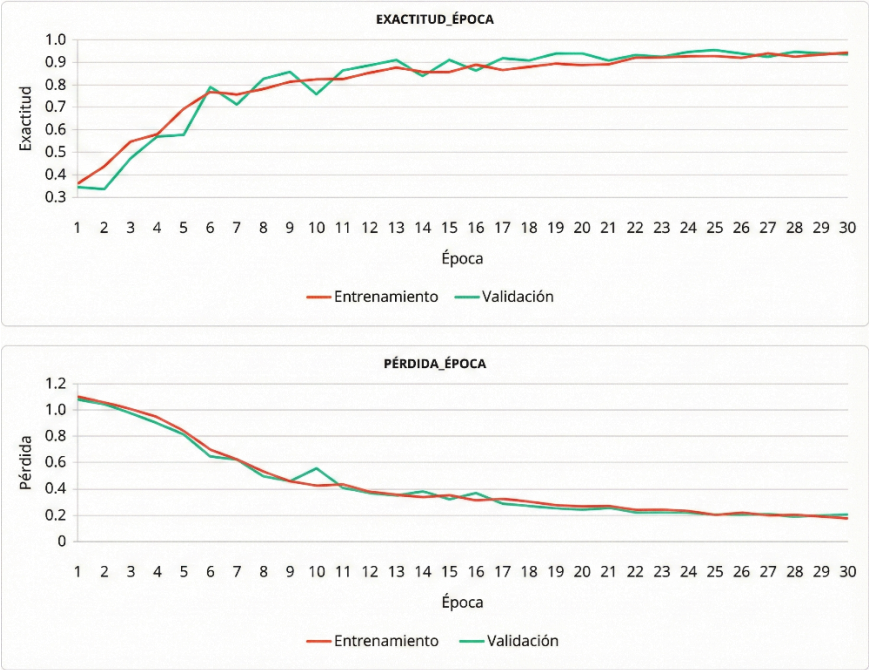

Figura D.2. Gráficas de exactitud y pérdida durante el entrenamiento y validación del modelo CNN (30 épocas).

Anexos

Artículo 1. Diseño y Validación de un Sistema IoT para el Monitoreo de Residuos Sólidos en Contenedores Utilizando un Sensor ToF Multi-Zona

Quinto Congreso Virtual

Los Microcontroladores y sus Aplicaciones

Certificamos que **Javier Martínez Fernández** ha participado en calidad de **Autor** en el **Quinto Congreso Virtual Los Microcontroladores y sus Aplicaciones** desarrollado entre el 06 de Octubre y el 09 de Octubre del año 2025, organizado por el Laboratorio de Técnicas Digitales y el Departamento de Electrónica de la Universidad Tecnológica Nacional Facultad Regional Paraná con su trabajo **Diseño y Validación de un Sistema IoT para el Monitoreo de Residuos Sólidos en Contenedores Utilizando un Sensor ToF Multi-Zona** publicado con ISBN 978-950-42-0256-1 y DOI https://doi.org/10.33414/ajea.1942.2025

Se extiende este certificado desde la ciudad de Paraná, Provincia de Entre Ríos, República Argentina con fecha 09 de Octubre de 2025

Resolución de Consejo Directivo FRP UTN Nº 547/2024

Ing. Raúl M. Caballero
Director Congreso
UTN – Facultad Regional Paraná

Ing. Enrique S. Burgos
Director de Departamento Electrónica
UTN - Facultad Regional Paraná

Ing. Alejandro Carrere
Decano
UTN - Facultad Regional Paraná

Libro	Folio	Nº Asistente
5	164	11917

Artículo 2. Calibración de Crosstalk para Sensores ToF Multi-zona con Cubiertas Protectoras

La Benemérita Universidad Autónoma de Puebla

A través de la Facultad de Ciencias de la Computación
otorga la presente

CONSTANCIA

A: Javier Martínez Fernández, Nicolás Quiroz Hernández, Ana María Rodríguez Domínguez, Roberto Olmos Pimentel

Por presentar la ponencia, "Calibración de Crosstalk para Sensores ToF Multi-zona con Cubiertas Protectoras",
en el evento XV Congreso Nacional de Ciencias de la Computación CONACIC 2025 de la Facultad de Ciencias de la Computación.

2025 Facultad de Ciencias de la Computación

"Pensar bien, para vivir mejor"
Puebla de Z., a de 10 de octubre de 2025

M.I. María del Consuelo Molina García
Directora

Facultad de Ciencias de la Computación 2025