UNIVERSIDAD ESTATAL
PENÍNSULA
DE SANTA ELENA

FACSISTEL

INGENIERÍA EN ELECTRÓNICA Y
AUTOMATIZACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR

Construcción de una máquina clasificadora de camarón automatizada con visión artificial y monitoreo de datos en la nube

Saul Ismael Muñoz Pinyui

Dirigido por:
Ing. Franklin Cesar Ramirez, Mgtr.

La Libertad - 2025

DEDICATORIA

Primeramente agradecido con Dios por todo el tiempo vivido en esta etapa bella de la universidad, también agradezco a mis padres José Roberto muñoz y Carmen Lourdes rosales, que me han apoyado incondicionalmente en esta etapa de mi vida, también a mis hermanos ing Jefferson, Gael, bella, que también han sido importantes y me han inspirado a seguir adelante con mis estudios, también espero que en un futuro no tan lejano poder devolver todo el apoyo que me han brindado incondicionalmente, por eso digo gracias familia de todo corazón.

También a mi abuela Amalia que no está presente y que en algún momento partió de este mundo, pero siempre recordando sus palabras de aliento y fortaleza que las he llevado de inspiración a lo largo de la carrera para seguir continuando con mis estudios y tambien a mi otra abuela Jacinta que ha aportado un gran valor significatico en mi vida.

También quiero agradecerle a mi especialista el ingeniero Luis enrique Chuquimarca y a mi tutor revisor Franklin Cesar Ramirez que gracias a sus enseñanzas he avanzado mucho en mis estudios de nivel superior gracias por su apoyo académico y tambien a los demas ingenieros que han aportado un gran valor academico para la especialidad en mi carrera.

A mis amigos Edu, Sergio, Orlin que desde fuera y dentro del colegio hemos venido teniendo una linda amistad espero seguir contando con ustedes y tambien para esas personas que por alguna razón ya nos hemos distanciado pero siempre deseandonos el bien.

esta dedicatoria es para mi ya que siempre luche hasta poder obtener este tan valioso y muy representativo titulo el cual simboliza dias y años de mucha constancia y a mi familia y familiares que me han servido de mucha inspiración para poder llegar hasta aqui.

Saul Ismael Muñoz Pinyui

AGRADECIMIENTO

En primer lugar, deseo expresar mi más profundo agradecimiento a Dios, fuente de sabiduría, fortaleza y esperanza, por brindarme la salud, la paciencia y la perseverancia necesarias para culminar con éxito esta etapa tan importante de mi vida académica. Cada dificultad superada y cada logro alcanzado fueron posibles gracias a su guía y bendición constante en mi camino.

A mi familia, especialmente a mis padres, les debo un reconocimiento inmenso por su amor incondicional, comprensión y sacrificio. Ellos han sido el pilar fundamental en cada paso de mi formación, brindándome no solo su apoyo económico, sino también su ejemplo de esfuerzo, responsabilidad y dedicación. A mis hermanos y demás familiares, gracias por su compañía, sus palabras de ánimo y por creer siempre en mis capacidades incluso en los momentos más difíciles. Sin ustedes, este logro no habría sido posible.

También quiero expresar mi gratitud a mis amigos, quienes me acompañaron a lo largo de este proceso con su apoyo sincero, su amistad desinteresada y su disposición para compartir experiencias, conocimientos y momentos de alegría. Su presencia fue esencial para mantener la motivación y la confianza necesarias para seguir adelante, convirtiendo los desafíos en aprendizajes y las horas de trabajo en oportunidades para crecer juntos.

A mis profesores, extiendo mi reconocimiento por su compromiso, paciencia y dedicación al transmitir sus conocimientos y experiencias. Cada enseñanza recibida ha contribuido al fortalecimiento de mis competencias profesionales y personales. En especial, agradezco a mi tutor de tesis por su orientación constante, su apoyo técnico y académico, así como por su tiempo y valiosas sugerencias que permitieron el correcto desarrollo de este proyecto.

Finalmente, agradezco a todas las personas que de una u otra manera colaboraron directa o indirectamente en la realización de esta investigación. A cada institución, compañero y colaborador que brindó su ayuda o compartió su conocimiento, les expreso mi más sincero reconocimiento. Este trabajo es el resultado del esfuerzo colectivo, de la cooperación y del compromiso compartido con el aprendizaje y la innovación. Gracias a todos por formar parte de este camino que marca el inicio de una nueva etapa profesional y personal, llena de retos, sueños y nuevas oportunidades.

Saul Ismael Muñoz Pinyui

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación denominado: "Construcción de una máquina clasificadora de camarón automatizada con visión artificial y monitoreo de datos en la nube", elaborado por el estudiante Saul Ismael Muñoz Pinyui, de la carrera de Ingeniería en Electrónica y Automatización de la Universidad Estatal Península de Santa Elena, declaro que luego de haber orientado, estudiado y revisado, lo apruebo en todas sus partes y autorizo al estudiante que inicie los trámites legales correspondientes.

Ing. Franklin Cesar Ramirez, Mgtr.
Tutor

TRIBUNAL DE GRADO

Ing. Ronald Humberto Rovira, Ph.D.
DIRECTOR DE CARRERA

Ing. Luis Enrique Chuquimarca, Mgtr.
DOCENTE ESPECIALISTA

Ing. Franklin César Ramírez, Mgtr.
TUTOR

Ing. Luis Enrique Chuquimarca, Mgtr.
DOCENTE GUÍA UIC

Ing. Corina Gonzabay De La A, Mgtr.
SECRETARIA DEL TRIBUNAL

Resumen

El siguiente proyecto consiste en la construcción de una maquina para clasificar camarones de distintos tamaños la cual se diseño una red neuronal para poder realizar la clasificación por visión artificial, estos datos serán monitoreados en tiempo real desde un sitio web de todo lo que este detectando tanto por la web cam y los camarones que pasan por los sensores infrarrojos. La construcción de la maquina consiste en realizar una estructura esta consiste en poder realizar una banda transportadora que contenga servomotores los cuales por medio de la detección del camarón enviarán una señal y estos moverán al camarón para enviarlos a la sección correspondiente a esa clase. El modelo de red neuronal se base en una detección de objetos por lo cual el modelo solo clasifica un camarón, este modelo tiene 7 capas de salidas los cuales son camarón pequeño, mediano grande también esas clases se le añade con y sin cabeza, la arquitectura utilizada para este modelo es MobileNetv2 muy buena para la implementación del modelo en la Raspberry pi 4, esta red fue diseñada en la plataforma Edge Impulse especialidad para diseño de modelo de redes neuronal para equipos embebidos. Para la implantación y él envió de datos en tiempo real se utilizó la plataforma Firebase la cual es de uso gratuito y sin límites, se creó 3 códigos el código Python para realizar la interfaz de la Raspberry y la implementación del modelo de manera local(solo acceso desde la Raspberry), código .Json para poder enviar los datos en tiempo real a la plataforma Firebase y código HTML para diseñar una interfaz y una pagina web que este monitoreando en tiempo real desde cualquier lado del Mundo con acceso a internet todo los datos que esta recibiendo la Raspberry pi en tiempo real. Cabe recalcar que este proyecto es innovador, no es costo como grades equipos de automatización dedicados al sector del camarón, se utilizan herramientas y tecnología actual, con lo cual el sector dedicado al camarón se puede modernizar su producción.

Palabras clave: clasificación, visión artificial, camarón, redes neuronales, nube, Rasbperry.

Abstract

The following project involves building a machine to classify shrimp of different sizes. A neural network was designed to perform the classification using computer vision. This data will be monitored in real time from a website, tracking everything detected by both the webcam and the shrimp passing through the infrared sensors. The machine's construction consists of a conveyor belt containing servomotors. Upon detecting a shrimp, the servomotors will send a signal, moving the shrimp to the section corresponding to its size. The neural network model is based on object detection, so it only classifies one shrimp at a time. This model has seven output layers: small, medium, and large shrimp. These classes are further categorized as having and without heads. The architecture used for this model is MobileNetv1, which is well-suited for implementation on the Raspberry Pi 4. This network was designed on the Edge Impulse platform, which specializes in designing neural network models for embedded systems.

For implementation and real-time data transmission, the Firebase platform was used, which is free and unlimited. Three pieces of code were created: Python code to create the Raspberry Pi interface and implement the model locally (accessible only from the Raspberry Pi); JSON code to send real-time data to the Firebase platform; and HTML code to design an interface and a website that monitors all the data received by the Raspberry Pi in real time from anywhere in the world with internet access.

It is worth noting that this project is innovative and does not have the cost of large automation systems dedicated to the shrimp sector. It uses current tools and technology, allowing the shrimp industry to modernize its production.

Keywords: classification, computer vision, shrimp, neural networks, cloud, Raspberry.

Índice

Índice de figuras

Índice de cuadros

1. Introducción

La industria alimentaria ha vivido diversas transformaciones y significativos progresos tecnológicos en los años recientes, los cuales han optimizado la efectividad y el calibre de las compañías que han empleado esta tecnología. La sociedad acepta cada vez más el uso de la visión artificial, especialmente en la industria pesquera, que es una de las industrias donde las empresas más se han beneficiado de los equipos avanzados.

La construcción de la máquina clasificadora automatizada con visión artificial permite al camarón aumentar la productividad y al mismo tiempo reducir los costos operativos; Además, reduce los errores humanos durante todo el proceso productivo, lo que garantiza y asegura una mayor calidad del producto final.

Esto es posible gracias al uso de hardware especializado, software programable e inteligencia artificial, para realizar tareas de mayor grado de complejidad de manera automática, sin la necesidad de intervención humana. Asimismo, asegura un nivel de producción medio, también llamadas constantes, así como las características físicas que los consumidores finales quieren ver en un camarón.

Finalmente gracias al almacenamiento en la nube, será posible supervisar de manera práctica todos los aspectos de importancia relacionados con el funcionamiento general, desde cualquier lugar donde se encuentre, siempre y cuando se tenga acceso a internet. Esta es una ventaja muy valiosa para resguardar información crítica.

1.1. Justificación

El camarón posee el primer lugar de exportación dentro del Ecuador. Durante los seis primeros meses del presente año se llegó a alcanzar un valor de USD 18.813 millones de ingresos para el país, lo cual representa un incremento del 11 % respecto al año anterior, posicionándose por encima del petróleo. Se han enviado 296 millones de libras hacia los principales destinos en el exterior, que son países como China, Estados Unidos y Europa [17].

En gran parte de la industria dedicada al camarón se continúa realizando el proceso de clasificación de forma manual, lo cual ocasiona que se utilice más tiempo del habitual. Por esta razón, surge la necesidad de automatizar la tarea de clasificación. Incorporando estas tecnologías se puede reducir el

costo de operación, mejorar la precisión y alcanzar una producción a mayor escala [18].

El tema estudiado en el presente proyecto resulta pertinente, ya que se busca desarrollar un sistema de clasificación de camarón enfocado en la inspección y monitoreo de calidad. Este sistema aplica parámetros de análisis visual, como forma y tamaño, con el fin de evitar la variación existente cuando la clasificación se realiza de manera manual. De esta forma se puede garantizar que los camarones cumplan los requisitos necesarios para su respectiva comercialización, tanto a nivel local como internacional.

La viabilidad del presente proyecto se basa en la utilización de nuevas tecnologías, como la visión artificial, que permite reconocer y revisar en tiempo real los camarones y clasificarlos por tamaño. Mediante este proceso es posible realizar un monitoreo constante de la clasificación, obteniendo resultados positivos. Con ello se demuestra que el proyecto es factible y, además, ofrece una alternativa para mejorar los niveles de productividad y contribuir a la modernización del sector acuícola dedicado al camarón [19].

1.2. Panorama actual

En el siguiente apartado se incluirá la revisión del estado del arte sobre temas de interés, cuyos autores y titulares sirvieron de orientación para vincular sus conocimientos con la presente propuesta de titulación.

El trabajo de titulación "Diseño de proyecto de automatización de una máquina clasificadora de camarón para empresa manufacturera", realizado en la ESPOL en 2025 por Derian Angulo Acosta, tiene como objetivo diseñar un sistema para automatizar las áreas de proceso y clasificación del camarón, utilizando equipos industriales como PLC Siemens, variadores de frecuencia y motores. Se plantea un sistema en el cual cada máquina cumple con un proceso específico en la clasificación; sin embargo, esta investigación se ve limitada, ya que solo se utiliza software de programación y no se emplean equipos o máquinas físicas. Esto demuestra que existen muchos factores que afectan e influyen en la implementación de un sistema real [20].

El trabajo de titulación "Sistema de reconocimiento y clasificación de camarón utilizando visión artificial", realizado en la Universidad Politécnica Salesiana, Quito, el 12 de agosto de 2023, por D. J. Zambrano Yagual y X. I. Toala Peña, tiene como objetivo aplicar visión artificial en la detección de características en camarones destinadas a la industria alimenticia. Se em-

plea un sistema de visión basado en redes neuronales, entrenado mediante aprendizaje automático, lo que permite reconocer características complejas entre diferentes tipos de camarones. Además, las condiciones ambientales no afectan significativamente la aplicación de este sistema; sin embargo, no se llega a una clasificación mecánica, pues se limita a una clasificación digital. A diferencia de ello, en la presente investigación se propone realizar la clasificación de forma práctica, mediante servomotores que seccionan los camarones de manera mecanizada según el grupo o clase asignado por la red neuronal implementada. En la actualidad, la utilización de visión artificial en las industrias acuícolas dedicadas al camarón permite analizar características relevantes del producto y facilita la automatización de procesos. Asimismo, esta tecnología hace posible clasificar los camarones con mayor precisión, reduciendo errores y garantizando el cumplimiento de las normas de calidad establecidas [21].

Estos antecedentes son de gran importancia para la presente investigación, ya que tratan sobre sistemas de clasificación de camarones, los cuales sirven como base para el desarrollo de una clasificadora que opere con visión artificial, un sistema de servomotores y un monitoreo en tiempo real de todo el proceso. Con ello, se busca clasificar los camarones en tres categorías de tamaño, mejorando la eficiencia y calidad del procedimiento.

1.3. Objetivos

1.3.1. Objetivo general

Construir una clasificadora de camarones por tamaño mediante dispositivos robustos y técnicas de visión artificial para el control y monitoreo del tamaño del producto.

1.3.2. Objetivos específicos

- Diseñar e implementar un sistema automatizado eficiente para la clasificación de camarones empleada.
- Desarrollar un algoritmo de visión artificial apto para que detecte y clasifique camarones según estos por su tamaño y clase a la que pertenece.
- Diseñar un sistema de control para servomotores para poder clasificar los camarones por clase y tamaño.
- Implementar una interfaz de conexión de datos entre la maquina clasificadora de camarón y una plataforma web para el monitoreo y análisis de los datos en tiempo real.

1.4. Fundamentos teóricos

En las siguientes secciones se tratarán temas importantes en clasificación, visión por computadora, redes de neuronas, aplicaciones de codificación y diversos asuntos relevantes para el avance de esta carrera académica.

1.4.1. Camarón industrializado

Es un animal acuático considerado un crustáceo; pertenece al orden de los decápodos, clasificados como animales de diez patas. Su cuerpo es segmentado y el camarón muda de piel para poder crecer. Además, cumple un papel importante en la cadena de alimentación acuática.

Existen diferentes tipos de camarón alrededor del mundo, pero los mas comercializados son: Litopenaeus vannamei (camarón Blanco), penaeus monodon (camarón tigre), camarón café o marrón (Farfantepenaeus californiensis) y camarón rojo (Farfantepenaeus brevirostris). El camarón café y el camarón rojo se muestran en la figura 1 [22].

Figura 1: Camarón de diferentes tallas

1.4.2. Clasificación del camarón por tamaño

En los siguientes items se detallará los tipos de clase o los tamaños de camarones que se emplea para el desarrollo del red neuronal, estos son:

1. **Pequeño:** Los camarones se pueden identificar por su tamaño cuando se recogen. Clasificación de camarones, el tamaño puede variar dependiendo de sectores o industrias específicas. Sin embargo, en general, los camarones se consideran pequeños si miden menos de 15 centímetros de largo. Cabe señalar que esta medida puede variar ligeramente dependiendo de los estándares y prácticas comerciales adoptadas por las diferentes empresas o instituciones del país y del extranjero.

2. **Mediano:** La medida puede cambiar dependiendo del área o de ciertos criterios, pero en general se considera camarón de tamaño medio a aquel cuya longuitud completa se encuentra aproximadamente entre 15 a 18 centímetros.

3. **Grande:** La categorización de los camarones de gran tamaño puede cambiar en función del sector y la situación. No obstante, en términos generales, se considera que los camarones son grandes cuando su longitud total excede los 18 centímetros.

En la figura 2 se pueden apreciar camarones de diferentes tallas [23].

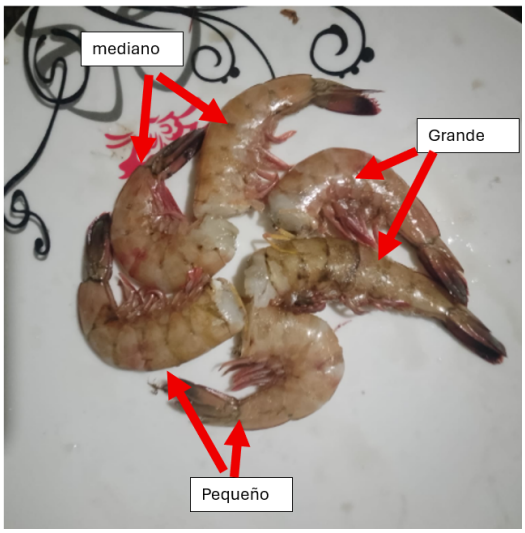


Figura 2: Clases de camarón pequeño, mediano y grande sin cabeza.

1.4.3. Normas aplicadas en el proceso

Las normas que se detallan a continuacion son normas para la garantizar un optimo desempeño en la labor de clasificación y higiene en el proceso de clasificación por visión artificial. Las principales normas empleadas son:

1. **ISO 22000:** La norma exige que las organizaciones identifiquen y evalúen los riesgos asociados con los comestibles a través de toda la línea de fabricación, y que desarrollen medidas preventivas para controlarlos. Asimismo, se deben establecer procedimientos documentados que garanticen el cumplimiento de los requisitos del sistema, incluidas las políticas, los objetivos, control contable y operativo. la comunicación interna y externa sobre La protección de la alimentación es esencial. Las entidades tienen que asegurarse de que se transmita información nítida y exacta a todas las partes interesadas. [24].

2. **INEN 456 1980-11:** Es una norma técnica ecuatoriana que define los requisitos para la clasificación y nombres comerciales del camarón. Se dio a conocer en noviembre de mil novecientos ochenta por el Instituto Ecuatoriano de Normalización (INEN). Esta norma tiene como objetivo establecer criterios para la categorización y los nombres comerciales del camarón de Ecuador. Esto incluye aspectos relacionados con sus características físicas y sensoriales, como el tamaño, el color, la textura y la apariencia. Además, la norma contiene lineamientos sobre cómo etiquetar o identificar productos de camarón por su calidad o categoría [25].

3. **FDA:** Las normas de la FDA exigen a los procesadores de camarones cumplir con los sistemas de Estudio de Amenazas y Puntos Clave de Supervisión (HACCP) y las Buenas Prácticas de Manufactura (BPM), garantizando la inocuidad del producto. Para una banda transportadora de camarones, esto significa que debe estar fabricada con materiales sanitarios, no porosos y de grado alimentario, diseñados para ser fáciles de limpiar y evitar la acumulación de bacterias. En el marco del HACCP, la limpieza y desinfección de la banda se establece como un punto de control crítico para mitigar riesgos de contaminación bacteriana y cruzada, requiriendo monitoreo y registros detallados. Las BPM aseguran que el equipo, como la banda transportadora, mantenga condiciones higiénicas óptimas, complementando así el control preventivo establecido por el sistema HACCP [26].

1.4.4. Sistemas de control industrial

En este apartado se integrarán métodos de sistemas de control industrial que garantizarán eficiencia y registro en tiempo real, incluyendo también el monitoreo en la nube. Existen diversos software de uso gratuito que permiten realizar dicho monitoreo en línea, además de las conexiones adecuadas para comunicarse en tiempo real con plataformas de IoT. Las principales etapas que se utilizán son:

1. **Adquisición de datos:** Es un proceso fundamental en el sector que se refiere a la recolección de datos de múltiples orígenes para su análisis y aplicación posterior. implica la adquisición de señales o características físicas usando sensores, transductores o dispositivos similares. Durante el proceso de recolección de datos, se realizan mediciones exactas que se almacenan de forma digital para su análisis posterior. Estos datos pueden abarcar variables como la temperatura, presión, flujo u otras variables relevantes para el sistema que se esta controlando. según la experiencia práctica, los sistemas modernos de recopilación de datos permiten una recopilación rápida y precisa, además de almacenar grandes volúmenes de información para su análisis posterior uso utilizando herrramientas avanzadas. El objetivo principal del proceso de adquisición es conseguir una representación clara de cómo opera el sistema que se está analizando. Esto a su vez, ofrece datos importantes sobre el rendimiento operativo y permite la toma de decisiones fundamentadas en hechos, aumentando la eficiciencia de los sistemas industriales en general [27].

2. **Monitoreo y control:** Es una actividad que asegura el correcto funcionamiento del proceso y el logro de los objetivos, implica el seguimiento continuo de las actividades y la evaluación constante de las actividades realizadas y que no existan errores que puedan poner en peligro el logro de los objetivos [28].

1.4.5. Protocolos de comunicación

A continuación, se definirán algunos protocolos de comunicación que se tomaron en cuenta para la elaboración de este trabajo de titulación. Estos protocolos son importantes, ya que permiten intercambiar información entre dos o más dispositivos y periféricos, además de establecer el orden en los procesos de envío y recepción de datos.

Las principales etapas que se utilizán son:

1. **HDMI:** Su abreviatura en inglés corresponde a HDMI (High-Definition Multimedia Interface), conocido en español como interfaz multimedia de alta definición. Este es el protocolo estándar que permite transmitir video y audio digital de alta calidad entre dispositivos, siendo muy utilizado en la conexión de equipos portátiles. Los cables HDMI pueden transmitir señales de audio tanto comprimidas como no comprimidas, lo que asegura versatilidad y fidelidad en la comunicación entre dispositivos. En la Figura 3 se presenta el circuito integrado del HDMI, el cual es ampliamente utilizado en aplicaciones, dicho cable cuenta con 19 pines,[29].

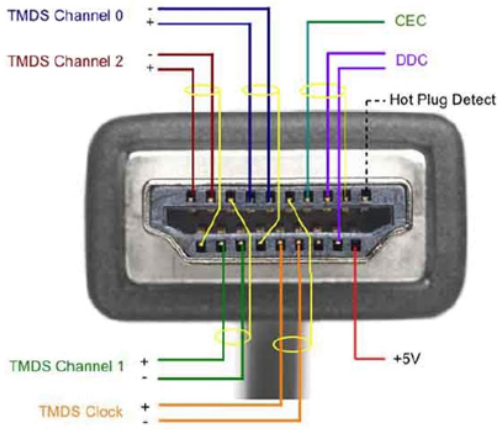


Figura 3: Comunicación HDMI con Rasperry [1].

2. **HTTP:** Este protocolo permite obtener recursos, datos y documentos HTML, constituyendo la base de la interacción en la web. Durante el intercambio de información en línea, sigue una arquitectura cliente-servidor. En este modelo, la solicitud de información se origina en el cliente —también denominado navegador web—, el cual envía peticiones al servidor. El servidor, a su vez, responde a dichas solicitudes mediante mensajes que contienen los recursos requeridos, conocidos como respuestas. De esta manera, se establece la comunicación entre el cliente y el servidor, lo que posibilita la visualización e interacción con páginas web. En la figura 4 se puede ver con mas detalle la comunicacion HTTP [30].

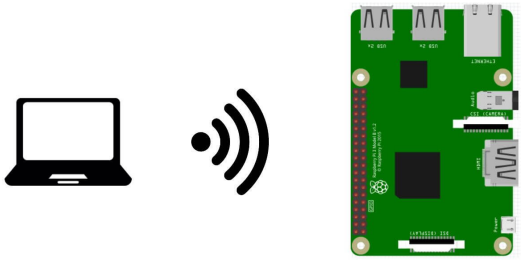


Figura 4: Comunicación HTTP con Rasperry [2].

1.4.6. Visión artificial

Permite diseñar sistemas capaces de procesar y comprender imágenes o videos de manera similar a la visión humana en un entorno determinado. Mediante algoritmos y técnicas avanzadas, es posible tener información útil de los datos para tareas de reconocimiento de objetos, el reconocimiento facial, el monitoreo de movimiento, entre otras. Existen diversas aplicaciones de la visión artificial en la industria, que abarcan desde el control de calidad en procesos de manufactura hasta la conducción automática de vehículos. En la figura 5 se detalla de mejor manera el proceso a realizar.

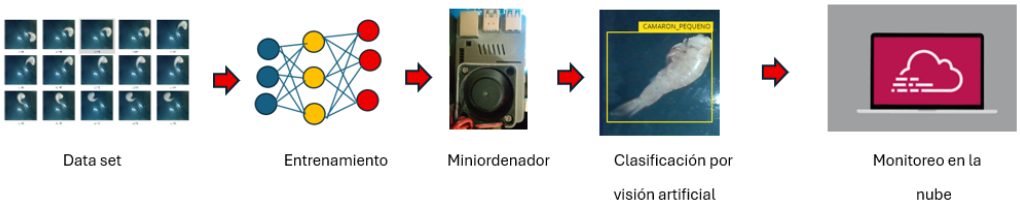


Figura 5: Etapas del proyecto de sistema de visión artificial para la clasificación de camarones y monitero en tiempo real a travez de la nube.

1.4.7. Etapas de un sistema de vision artificial

Las principales etapas que componen el sistema de visión artificial son:

1. **Captura de imagen:** Es un proceso fundamental para el análisis y procesamiento en visión artificial (figura 6, adquisición de imágenes para la creación de un dataset). Este procedimiento implica capturar imágenes mediante cámaras o sensores, seguida de un preprocesamiento destinado a mejorar la calidad y eliminar el ruido.

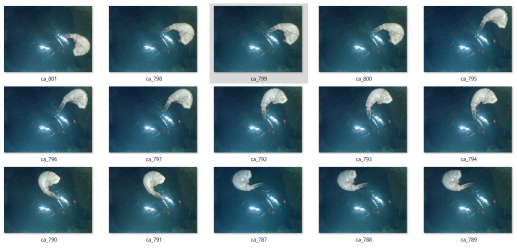


Figura 6: Contrucción del dataset de camarones.

2. **Segmentación:** El paso de segmentación de la imagen es fundamental para identificar y aislar objetos de imágenes o áreas de interés. Para ello, se utilizan técnicas y algoritmos avanzados que permiten dividirla en partes significativas en función de características como el color, la forma y la textura. Este proceso facilita la distinción entre los objetos y el fondo, lo que a su vez mejora el análisis posterior. La segmentación permite detectar contornos, resaltar áreas específicas y obtener información relevante para diversas aplicaciones, como la detección de rostros, entre otras. En la figura 7 se ilustra este proceso [31].

Figura 7: Segmentación de imagenes por clases.

3. **Reconocimiento o clasificación:** La etapa de reconocimiento o clasificación es muy importante para identificar y clasificar objetivos y clasificar objetivos en imágenes (ver Figura 8). Este proceso se puede realizar automáticamente mediante modelos de entrenamiento que puedan reconocoer los patrones y características de objetos específicos Estos modelos permiten asignar etiquetas a elementos identificados, facilitando la toma de decisiones visual. el reconocimiento y la clasificación se utilizan ampliamente en diversos campos, como el reconocimiento facial y el control de calidad[32].

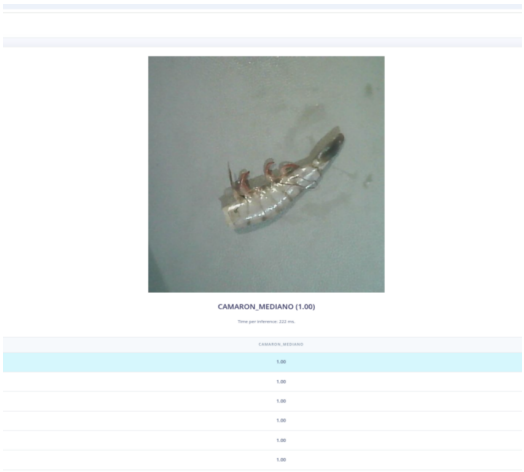


Figura 8: Clasificación en tiempo real.

1.4.8. Red neuronal

Una red neuronal artificial es un modelo matemático inspirado en el funcionamiento del cerebro humano. Se basa en la representación de una estructura interconectada de nodos, también llamados neuronas artificiales, que procesan información y realizan tareas específicas como clasificación, reconocimiento de voz o detección de objetos (véase la figura 9).Estas redes son altamente flexibles y se han consolidado como una poderosa herramienta para la resolución de problemas complejos en el campo de la visión artificial, entre muchas otras aplicaciones [33].

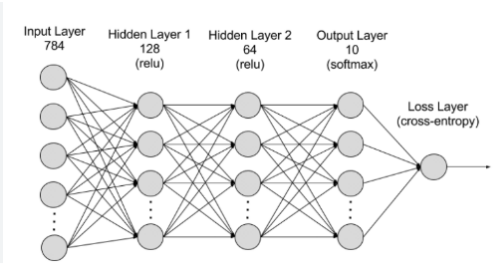


Figura 9: Estructura de una red Neuronal Artificial común[3].

1.4.9. Estructura de una CNN

Está diseñada para procesar datos en forma de imágenes, las cuales son representaciones digitales compuestas por un conjunto de píxeles. Estas redes tienen la capacidad de extraer automáticamente las características más

relevantes de las imágenes, tales como bordes, formas y objetos comunes, a través de sus capas convolucionales, donde dichas características se almacenan y procesan. El aprendizaje de una CNN se produce a medida que la red analiza registros individuales de datos, generando una predicción para cada uno. Cuando las ponderaciones iniciales son incorrectas, la red realiza ajustes mediante un proceso de retropropagación. Con cada iteración y al alimentarse con más datos, la red mejora progresivamente, logrando predicciones con mayores niveles de precisión. Las principales capas que componen una CNN son:

1. **Capa de entrada:**Las unidades de entrada representan los datos iniciales que recibe la red neuronal. Estas pueden estar organizadas en una o varias capas, y hacen referencia directamente a la información de entrada. En nuestro caso, corresponden a imágenes digitales, las cuales están compuestas por píxeles. Para procesarlas en la red, es necesario definir el ancho y el alto de la imagen, lo que permite que cada píxel sea interpretado como una unidad dentro de la capa de entrada.

2. **Bloque de procesamiento:**Esto hace referencia a lo que va a procesar la red, en este caso, las imágenes capturadas que posteriormente serán clasificadas o segmentadas. El propósito del procesamiento es mejorar la calidad de las imágenes y facilitar su análisis, lo que permite una búsqueda, detección y clasificación más eficiente.

3. **Bloque de aprendizaje:** Este aspecto es fundamental, ya que define el tipo de clase al que se orientará la red neuronal. En nuestro caso, se emplea la detección de objetos para distinguir los camarones entre sí, ya sea por tamaño o por sus características específicas. Además, se añaden las clases de objetos o atributos correspondientes a cada conjunto, de acuerdo con la categoría a la que pertenecen.

4. **Capa de salida:** Esta etapa representa la clasificación de salida de la red, es decir, cuántas salidas se van a obtener. También puede expresarse como el número de clases de detección u objetos que tendremos como resultado final del modelo.

A continuación, en la figura 10 se aprecia la arquitectura previamente mencionada que se utilizó para la creación de la CNN la cual consta en el modelo MobileNetv1 desarrollada con Edge Impulse.

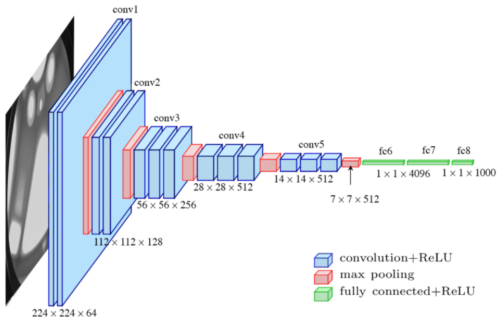


Figura 10: Estructura CNN MobileNetv1 [4].

1.4.10. Microprocesador

Es una circuito integrado que actúa como cerebro de un ordenador o dispositivo electrónico como se ve en la Figura 11. Su función es llevar a cabo órdenes, efectuar cálculos y gestionar todas las funciones del sistema. Se compone de unidades lógicas y aritméticas, registros y caché, lo que facilita un almacenamiento y procesamiento ágil de datos. Los microprocesadores son fundamentales en la tecnología moderna ya que permiten llevar a cabo aplicaciones informáticas complejas y una variedad de tareas de manera eficiente en tiempo real.[34].

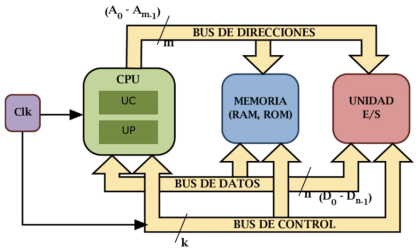


Figura 11: Estructura de un microprocesador [5].

1.4.11. Motor de corriente continua con engranajes reductor

En la figura 12 se ve que es un aparato electromecánico que transforma la electricidad en movimiento rotatorio, estos motores están compuestos por un rotor y un estator a través de los cuales circula la corriente a través de unas escobillas y un conmutador. La interacción entre el campo magnético creado por las bobinas del estator y la carga magnética del rotor genera el par motor, que impulsa el movimiento de rotación necesario para que la máquina automatizada realice sus funciones. Además, suelen venir acoplados a engranajes, los cuales reducen la velocidad del motor, aumentando así el torque en determinadas circunstancias donde sea requerido [35].

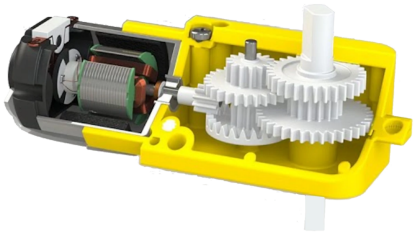


Figura 12: Motor corriente continua con engranajes reductor [6].

1.4.12. Servomotores

Como se ve en la figura 13 a pesar de su tamaño reducido, son capaces de generar una gran cantidad de potencia, además de destacar por su ahorro de energía. Se trata de un actuador giratorio que combina la potencia de un motor con la precisión de un reloj, en otras palabras, es un motor con una posición. Ofrece un control preciso de la posición angular, la aceleración y la velocidad, características que un motor común no posee [36].

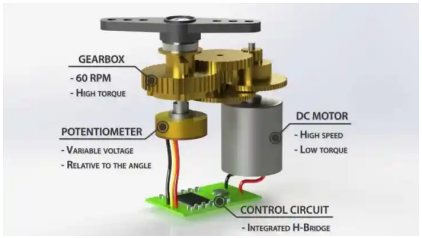


Figura 13: Servomotor [7].

1.4.13. Sensores de fotoeléctricos

En la Figura 14 se ve que el sensor emite luz visible o infrarroja desde su elemento emisor. Existe el tipo fotoeléctrico reflectivo, que detecta la luz reflejada desde el objeto. Estos sensores cuentan con emisor y receptor incorporados, de manera similar a los sensores de ultrasonido. Por su parte, el sensor de haz de barrera mide el cambio en la intensidad de luz cuando un objeto pasa y obstruye el eje óptico [37].

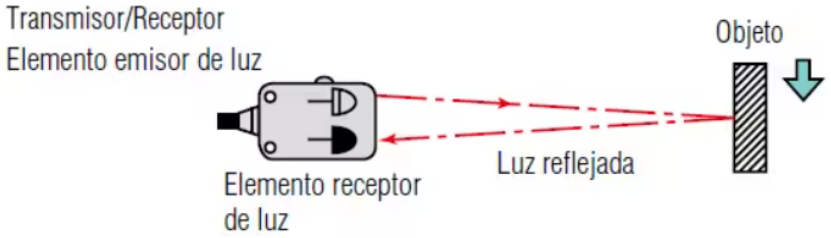


Figura 14: Sensor de proximidad infrarrojo [8].

1.4.14. Python

Ampliamente utilizado en una gran variedad de campos como el desarrollo de aplicaciones web, la automatización de procesos, la gestión y análisis de datos, la inteligencia artificial y el aprendizaje automático (machine learning). Su popularidad se debe principalmente a su simplicidad, legibilidad y versatilidad, como para desarrolladores experimentados. Una de las principales características de Python es su sintaxis clara y cercana al lenguaje humano, lo que facilita la comprensión del código y acelera el proceso de aprendizaje. Además, su estructura flexible permite escribir programas eficientes con menos líneas de código en comparación con otros lenguajes, favoreciendo la productividad y el mantenimiento del software.[38].

1.4.15. Firebase

Firebase es una plataforma integral de desarrollo de aplicaciones creada por Google que ofrece una amplia gama de servicios en la nube destinados a facilitar el diseño, la construcción y la administración de aplicaciones web y móviles. Su principal propósito es simplificar el proceso de desarrollo al proporcionar herramientas listas para usar, evitando que los programadores deban construir y mantener su propio servidor o infraestructura de backend. En esencia, Firebase funciona como un Backend-as-a-Service (BaaS), es decir, un servicio que ofrece todas las funciones del backend desde la nube,

permitiendo al desarrollador enfocarse más en la experiencia del usuario y en las características principales de la aplicación [39].

1.4.16. Edge Impulse

Es una plataforma innovadora diseñada para facilitar el desarrollo, entrenamiento y despliegue de modelos de inteligencia artificial (IA) y aprendizaje automático (Machine Learning) en dispositivos con recursos limitados, como microcontroladores, placas de desarrollo (Arduino, Raspberry Pi, ESP32, etc.) y sensores inteligentes. Su principal objetivo es llevar el poder de la IA al borde del sistema, es decir, al propio dispositivo donde se capturan los datos, sin depender constantemente de servicios en la nube o de una conexión a Internet[40].

1.5. Marco contextual

La industria acuícola, especialmente en países como México, Ecuador, India y Tailandia, ha experimentado un crecimiento significativo en las últimas decadas. Uno de los productos más destacados es camarón, cuya exportación constituye una importante fuente de ingresos económicos [41, 42].

Sin emabrgo, Uno de los procedimientos más sensitivos y que requiere intervención manual es el ordenamiento según dimensiones y calidad, lo cual afecta de manera directa el valor de mercado del artículo [43].

La clasificación de camarones hasta ahora se realiza manualmente, lo que se asocia con altos costos de mano de obra, resultados variables y eficiencia limitada. Estas condiciones, que complementan la creciente necesidad de automatización dentro de la industria 4.0 ha impulsado el desarrollo de soluciones tecnologicas orientadas a mejorar los procesos de la precisión y trazabilidad de este proceso [44, 45].

En este contexto, se propone un sistema de visión por computadora, capaz de capturar imágenes de los productos procesados, utilizando algoritmos de clasificación para determinar automáticamente el tamaño y tipo de camarones [46, 47]. El propósito de este sistema es aplicarse en nivel de empaquetado donde el tiempo de respuesta deber ser bajo y eficicencia alta [48].

Además, hay un sistema de almacenamiento en la nube incorporado que permite el registro automático de los reultados de la clasificación en tiempo real, generar informes accesibles desde cualquier lugar y ofrecer trazabilidad de los productos procesados [49, 50]. Este enfoque facilita la toma de decisiones, la supervisión del proceso, la optimización de la logística mejora de los estándares de exportación y calidad.

2. Métodos y diseño experimental

2.1. Alcance del proyecto

El alcance de este proyecto se enfoca en implementar un sistema automatizado para clasificar camarones en una cinta transportadora, con el objetivo de optimizar y reemplazar los procesos manuales utilizados en el sector de la elaboración de camarones. La meta principal es crear un sistema que satisfaga los siguientes puntos esenciales: eficiencia, precisión y automatización, logrando un aumento en la capacidad de clasificación actualmente disponible. Además, se busca reducir costos de operación y mejorar la calidad del camarón procesado como producto final.

La máquina clasificadora estará equipada con tecnología de visión artificial, implementada en una Raspberry Pi 4, para detectar el tamaño del camarón y realizar la clasificación correspondiente. Esto permitirá obtener un monitoreo en tiempo real de los datos recopilados por los sensores y servomotores integrados al sistema de clasificación.

Este sistema permite visualizar detalladamente el proceso de clasificación del camarón. Además, la solución incluirá una sección donde los camarones serán clasificados por tamaño: pequeño, mediano o grande— y también se podrán diferenciar camarones con cabeza o sin cabeza, independientemente del tamaño. Cabe recalcar que solo se clasificarán siete tipos de clases por proceso o modelo ya entrenado.

Se empleó la plataforma Edge Impulse para crear y capacitar la red neuronal por su notable capacidad de procesamiento y su eficacia en la elaboración de modelos de machine learning destinados a la clasificación de camarones. Además, el sistema permite la visualización de datos al instante, que incluye valores cambiantes de sensores, actuadores y cámaras, mediante una IoT en línea o una plataforma basada en la nube.

2.2. Metodología de investigación aplicada

La metodología de investigación aplicada consiste en utilizar tecnología nueva, o aplicarla a un proceso tradicional, para mejorar su desempeño en términos de eficiencia y eficacia. El principal objetivo de esta investigación es desarrollar una clasificación automatizada que implemente visión artificial sobre una banda transportadora, que permitirá el monitoreo en tiempo real mediante la recopilación de datos almacenados en la nube.

Para lograr este objetivo, se seguirán las siguientes fases:

1. **Investigación bibliográfica:** Se recopila información relevante acerca del tema, obtenida de textos, páginas web y tesis, incluyendo los orígenes y la evolución del sector camaronero del país. Además, se investigan nuevas tecnologías aplicables al diseño, haciendo énfasis en la inteligencia artificial utilizada en el procesamiento de alimentos, específicamente en mariscos, con un enfoque en el camarón.

2. **Selección de hardware:** Se realiza un análisis detallado para determinar los instrumentos y componentes electrónicos más adecuados para el proyecto. Se consideran aspectos como costo, potencia de procesamiento, conectividad a Internet, monitoreo remoto, compatibilidad con sensores y actuadores y fácil acceso a plataformas web para monitorear el proceso en tiempo real.

3. **Implementación del sistema eléctrico/electrónico:** Una vez seleccionado el hardware, se escogen los dispositivos electrónicos compatibles con la placa principal, como cámara, sensores, actuadores y acondicionadores de voltaje, asegurando que las señales de entrada y salida sean compatibles con los dispositivos y permitan la correcta activación de los actuadores.

4. **Adquisición de datos:** Se desarrolla un programa en Python para recolectar datos e imágenes, que serán posteriormente clasificadas y etiquetadas por clases. Esta información servirá para el entrenamiento de la red neuronal que realizará la clasificación automática de los camarones por tamaño.

5. **Formación de la red neuronal:** Una red neuronal se entrena utilizando imágenes que han sido etiquetadas. Cuando la red complete su entrenamiento y logre un grado satisfactorio de identificación, será capaz de identificar la especie de cada camarón que detecte en tiempo real a través de una cámara conectada a una Raspberry Pi 4. El proceso de formación se lleva a cabo con Edge Impulse, que optimiza su capacidad de procesamiento y permite ejecutar modelos ligeros sin afectar el rendimiento de la Raspberry Pi 4.

6. **Manejo de clasificación:** Un sistema automático de categorización se conecta con el sistema de detección instantánea y la red capacitada, donde los servomotores activan los mecanismos de separación de acuerdo con la clasificación establecida. Sensores de infrarrojos monitorean

el avance de cada camarón y facilitan una clasificación exacta conforme a su tipo.

7. **Monitoreo remoto a través de plataforma virtual o nube:** Finalmente, todos los estados de sensores, actuadores y cámaras se transmiten en tiempo real a través de la nube, permitiendo supervisar el proceso de clasificación de camarones desde cualquier lugar, facilitando la toma de decisiones y el control del sistema de manera remota.

2.2.1. Descripción del proyecto

El desarrollo de la construcción de una máquina clasificadora de camarón automatizada, equipada con visión artificial y monitoreo de datos en la nube, adaptada a una banda transportadora, es un proyecto innovador con el objetivo de aumentar la efectividad en el sector de la crianza de camarones. Este dispositivo ha sido creado para clasificar de manera automática los camarones empleando tecnología de visión artificial que permite reconocer y diferenciar los camarones según su tamaño y calidad, disminuyendo así el tiempo y el costo de producción.

El empleo de sistemas de visión artificial hace posible que la máquina identifique y separe los camarones de forma precisa, lo que incrementa tanto la calidad como la eficiencia del proceso. A esto se suma un sistema de monitoreo en la nube que permite seguir la producción al instante, ayudando a tomar decisiones rápidas y a manejar el rendimiento general del equipo.

La máquina está instalada sobre una banda transportadora, que hace avanzar los camarones para que, mediante una webcam conectada al microprocesador Raspberry Pi 4, sean clasificados según su tamaño: pequeño, mediano o grande. Una vez completada la clasificación por visión artificial, se activan los servomotores, que separan los camarones según la clase a la que pertenecen.

El sistema de clasificación separa los camarones según tamaño y calidad, según la norma FDA, lo que permite mantener un sistema de seguridad higiénico durante todo el proceso.

El sistema de vigilancia, la nube, es un componente crucial de la instalación, dado que facilita la observación de la producción al instante. La información recolectada se guarda en la nube, permitiendo a los usuarios consultarla desde cualquier lugar y en cualquier momento, siempre que dispongan de acceso a Internet.

La elaboración de este equipo demanda una inversión considerable en tecnología para garantizar que opere de manera adecuada. Asimismo, como el sistema produce y maneja datos de forma continua, resulta imprescindible contar con una conexión a internet confiable que permita registrar la información proveniente de los sensores, actuadores y cámaras durante todo el proceso de clasificación.

El propósito del proyecto es fabricar una máquina automatizada de clasificación de camarón, que incorpore un algoritmo de visión artificial para detectar y clasificar los camarones según su tamaño. Los actuadores, mediante las órdenes enviadas por el microprocesador, efectuarán la separación de manera automatizada, obteniendo así una clasificación precisa por tamaño.

Finalmente, se desarrollará una interfaz que integre todos los datos recopilados y almacenados por la máquina, permitiendo enviar la información a la nube (plataforma web) para su monitoreo y análisis en tiempo real.

2.3. Componentes de la propuesta

En los siguientes apartados se presentará una descripción detallada de los componentes de software y hardware que intervienen en el desarrollo de este proyecto. Estos elementos representan tanto la parte lógica como la física del sistema, siendo ambos esenciales para su correcto funcionamiento e integración.

Primero, se cubrirá el software, que incluye las herramientas, entornos de programación y plataformas digitales utilizadas para diseñar, codificar y ejecutar el sistema. En esta sección se explicarán las funciones que realiza cada programa, sus principales características, los motivos por los que fueron elegidos y la forma en que se interconectan para asegurar un adecuado procesamiento de datos y comunicación con los componentes físicos.

Posteriormente, se describirá el hardware, conformado por los dispositivos electrónicos y módulos que permiten la interacción directa con el entorno físico. Aquí se detallarán los sensores, actuadores, microcontroladores, cámaras, servomotores y demás elementos utilizados, especificando sus características técnicas, su conexión al sistema y su papel dentro de la arquitectura general del proyecto..

2.3.1. Componentes físicos

En este apartado se detallan todos los componentes de hardware utilizados para el desarrollo del presente trabajo de titulación, mediante los cuales fue posible implementar el circuito electrónico requerido para el proyecto.

2.3.1.1. Raspberry Pi 4

La Raspberry Pi 4 constituye el cerebro principal del sistema, ya que es la encargada de coordinar y controlar el funcionamiento general de todos los componentes involucrados. Este dispositivo actúa como una unidad central de procesamiento (CPU) capaz de ejecutar programas complejos y procesar información en tiempo real, lo que la convierte en el elemento esencial para la gestión de los servomotores, los actuadores y la detección automática de camarones mediante técnicas de inteligencia artificial. Además, la Raspberry Pi 4 cuenta con una amplia variedad de puertos y conexiones, entre ellos

puertos GPIO (General Purpose Input/Output), que posibilitan la comunicación directa con sensores, cámaras, y servomotores, permitiendo el control físico del sistema. También dispone de conectividad USB, HDMI, Ethernet y Wi-Fi, lo que facilita la interacción con otros dispositivos, la visualización de resultados en pantalla y la actualización del software de manera remota (figura 15 y cuadro 1).

Figura 15: Placa Rasperry pi [9].

Cuadro 1: Características técnicas del Raspberry Pi 4 [16].

DETALLES	**ESPECIFICACIONES**
Modelo	Raspberry Pi 4 - 4Gb
Procesador	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) SoC de 64 bits a 1,5 GHz
Memoria	4 GB LPDDR4-3200 SDRAM
Conectividad	LAN inalámbrica IEEE 802.11ac de 2,4 GHz y 5,0 Ghz, Bluetooth 5.0 BLE, Gigabit Ethernet
Video y sonido	2 puertos micro-HDMI (compatible con hasta 4K p60), Puerto de cámara CSI para conectar una cámara Raspberry Pi, Puerto de pantalla DSI para conectar una pantalla táctil Raspberry Pi
Puertos	2 puertos USB 3.0, 2 puertos USB 2.0
Alimentación	Entrada de alimentación de CC de 5 V/3 A (USB-C o GPIO)
Expansión	40 pines macho GPIO (27 E/S, UART, I2C, SPI)

2.3.1.2. Servomotor MG996R 180°

El servomotor MG996R es un actuador de precisión ampliamente utilizado en proyectos de robótica, automatización y control, gracias a su alto torque, resistencia mecánica y compatibilidad con múltiples plataformas electrónicas. Este modelo se distingue por ofrecer un par de torsión de aproximadamente 13 kg·cm a 6 V, lo que le permite mover cargas moderadas con gran estabilidad y precisión, siendo ideal para aplicaciones que requieren fuerza y exactitud en el posicionamiento. Su diseño incorpora engranajes metálicos de alta resistencia, los cuales brindan una estructura robusta y duradera, capaz de soportar esfuerzos mecánicos continuos sin degradar su rendimiento. Esta característica le otorga una mayor vida útil en comparación con servomotores que emplean engranajes plásticos, haciéndolo adecuado para entornos exigentes o de uso prolongado (figura 16 y cuadro 2).

Figura 16: Servomotor MG996R [10].

Cuadro 2: Características técnicas servomotor MG996R [10].

Modelo	MG996R
Torque	10.4 Kg/cm (4,8 V), 13Kg/cm(6v)
Voltaje de operación	4.8 V - 7.2 V
Velocidad de operación	(4.8 V sin carga): 0.2 seg/60 grados. (6 V con carga): 0.16 seg/60 grados.
Tamaño	40.6 x 19.8 x 42.9 mm
Plug	Jr, FUTABA general
Ángulo de rotación	180° máximo
Material engranajes	Metal40
Pulso ciclo	20 ms.
Ancho del pulso	600uS y 2400uS
Rango de temperatura	-30 a +60 °C

2.3.1.3. Tarjeta microSD

La tarjeta microSD es un dispositivo de almacenamiento portátil diseñado para guardar, transferir y proteger datos digitales de forma eficiente y segura. Se trata de una versión reducida de las tarjetas SD (Secure Digital), caracterizada por su tamaño compacto y alta capacidad de almacenamiento, lo que la convierte en una solución ideal para equipos electrónicos de dimensiones limitadas o con requerimientos de almacenamiento externo. Su funcionamiento se basa en el uso de memoria flash NAND, una tecnología que permite almacenar información sin necesidad de energía eléctrica, manteniendo los datos de manera permanente incluso cuando el dispositivo se apaga. Gracias a esta característica, las tarjetas microSD ofrecen una gran durabilidad y fiabilidad, con una vida útil que puede extenderse a miles de ciclos de lectura y escritura (figura 17 y cuadro 3).

Figura 17: Tarjeta microSD [11].

Cuadro 3: Características técnicas Tarjeta microSD [11].

Formato	Micro SDHC
Capacidad	64 GB
Velocidad de escritura	15 MB/s
Memoria flash NAND	TLC (triple level cell)
Dimensiones	14.99 x 1.02 mm
Temperatura de operación	$0°C$ a $70°C$

2.3.1.4. Sensor infrarrojo E18-D80NK

El sensor es un dispositivo infrarrojo E18-D80NK electrónico de detección de proximidad ampliamente utilizado en sistemas automatizados, robótica y control industrial. Su función principal es detectar la presencia u objetos dentro de un rango de distancia predeterminado, sin necesidad de contacto físico, ofreciendo mediciones estables, precisas y confiables. Este sensor opera mediante el principio de reflexión de luz infrarroja: emite un haz de luz invisible a través de un diodo emisor (LED IR) y mide la cantidad de luz reflejada por un objeto mediante un fototransistor receptor. Cuando un objeto entra en el rango de detección, la luz infrarroja se refleja hacia el receptor, activando la señal de salida digital. Este tipo de funcionamiento permite una detección rápida y sin desgaste mecánico, ideal para aplicaciones donde se requiere identificar presencia o movimiento sin contacto. El modelo E18-D80NK ofrece un rango de detección ajustable entre 3 y 80 centímetros, el cual puede modificarse mediante un potenciómetro incorporado en su parte posterior. Gracias a este ajuste, el sensor puede adaptarse a diferentes entornos y necesidades, desde detecciones muy cercanas hasta distancias medias. Además, su diseño es altamente resistente a interferencias causadas por la luz ambiental, lo que garantiza un desempeño estable incluso en condiciones de iluminación intensa o en exteriores. (figura 18 y cuadro 4).

Figura 18: Sensor infrarrojo E18-D80NK [12].

Cuadro 4: Características técnicas Sensor infrarrojo E18-D80NK [12].

Voltaje de operación	5V
Conexiones	Vcc, Trigger, Echo, GND
Rango de medición	2 cm - 400 cm
Precisión	±3 mm
Corriente de alimentación	15 mA
Frecuencia de pulsos	40 kHz
Ángulo efectivo de medición	$\frac{1}{15}^{\circ}$
Señal de disparo	10 µs
Dimensiones	45mm × 20mm × 15mm

2.3.1.5. WEB-CAM

La Web-cam es Utilizada en este proyecto desempeña un papel fundamental dentro del sistema de visión artificial, ya que se encarga de capturar imágenes en tiempo real con una calidad y resolución adecuadas para el análisis computacional. Este dispositivo cuenta con la capacidad de registrar imágenes en alta definición (Full HD 1080p), con una resolución de 1920 x 1080 píxeles, lo que garantiza una representación nítida y detallada de los objetos en escena, condición indispensable para el correcto funcionamiento de los algoritmos de detección y clasificación. El sensor óptico incorporado en la cámara permite obtener imágenes con una buena reproducción de color, contraste y luminosidad, incluso en condiciones de iluminación variable. Su conexión mediante cable USB 2.0 tipo-A ofrece una comunicación rápida y estable con la Raspberry Pi 4, la cual actúa como unidad central de procesamiento. (figura 19 y cuadro 5).

Figura 19: WEB-CAM 1920x1080 [13].

Cuadro 5: Características técnicas WEB-CAM [13].

Enfoque	Manual
Tamaño de video	1920 × 1080 (Full HD)
Velocidad de fotogramas	30 FPS
Sensibilidad	2.0 V/Lux
Distancia de enfoque	Mínimo 2 cm al infinito
Conexión	USB 2.0

2.3.1.6. Conversor TXS0108E

El modulo TXS0108E es una placa electrónica de adaptación de voltaje diseñada para permitir la comunicación segura y eficiente entre dispositivos que operan con diferentes niveles lógicos de tensión. Este módulo resulta esencial en proyectos donde intervienen microcontroladores, sensores o periféricos que no comparten el mismo rango de voltaje de operación, evitando incompatibilidades eléctricas o daños en los componentes sensibles del sistema. En su funcionamiento, el TXS0108E actúa como un traductor bidireccional automático de señales, capaz de convertir tensiones de 3.3V a 5V y viceversa sin necesidad de control externo adicional. Esto lo convierte en una herramienta sumamente versátil en entornos donde se requiere comunicación entre placas controladoras de 3.3V, como Raspberry Pi, ESP32 o ESP8266, y módulos o sensores de 5V, como algunos servomotores, pantallas LCD, o sensores de ultrasonido. Cada canal puede transmitir datos en ambas direcciones, de manera automática y dinámica, gracias a los transistores MOSFET de paso controlado integrados en su circuito interno, los cuales detectan el nivel lógico del dispositivo conectado y ajustan la tensión de salida en consecuencia.(figura 20 y cuadro 6).

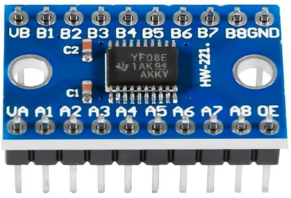


Figura 20: Modulo TXS0108E [14].

Cuadro 6: Características placa conversor TXS0108E [14].

Canales	8 entrada/salida bidireccionales
Voltaje	Lado A: 1.2V a 3.6V Lado B: 1.6V a 5.5V
Velocidad	60 Mbps (máxima)

2.3.1.7. Capacitores electrolíticos

Los capacitores electrolíticos son componentes electrónicos pasivos ampliamente utilizados en todo tipo de circuitos y placas electrónicas debido a su capacidad para almacenar carga eléctrica y filtrar señales no deseadas dentro de sistemas de corriente continua o alterna. Su principal función es estabilizar y suavizar el flujo de energía eléctrica, especialmente en líneas de alimentación, donde ayudan a eliminar picos de voltaje y reducir el ruido eléctrico generado por variaciones en la corriente o interferencias externas. Estos capacitores se distinguen por ofrecer altos valores de capacitancia (desde microfaradios hasta miles de microfaradios) en un tamaño físico compacto, lo cual los hace ideales para aplicaciones en fuentes de alimentación, amplificadores de audio, controladores de motores, microcontroladores y sistemas embebidos. Su estructura interna se compone de dos placas conductoras (ánodo y cátodo) separadas por un dieléctrico formado por una capa delgada de óxido de aluminio, que se obtiene mediante un proceso de electrólisis. Este material dieléctrico es el que confiere al componente su alta capacitancia y su denominación como "electrolítico". (figura 21 y cuadro 7).

Figura 21: Capacitores electroliticos [15].

Cuadro 7: Características técnicas de capacitores electrolíticos [15].

Filtrado de señales	Efectividad depende de la capacitancia y voltaje (mayor capacitancia = mejor filtrado para bajas frecuencias)
Polaridad	Poseen polaridad definida: ▪ Terminal negativo marcado con franja gris/negra ▪ Terminal positivo generalmente más largo
Rango térmico	-40°C a +105°C (dependiendo del modelo)
Características adicionales	▪ Alta capacitancia (1µF a 10000µF) ▪ Bajo costo ▪ Vida útil limitada por secado del electrolito

2.3.1.8. Banda transportadora

Es uno de los componentes mecánicos más importantes de este proyecto, desempeñando un papel central en el flujo y manipulación de los camarones dentro del sistema de clasificación automatizado. Su función principal consiste en desplazar los productos de manera continua y controlada a lo largo del recorrido de procesamiento, asegurando que cada camarón llegue a la zona de inspección y clasificación en una posición y velocidad adecuadas para la captura de imágenes y la posterior separación.Sobre la superficie de la banda se integra el sistema de visión artificial, compuesto por la cámara y el modelo de inteligencia artificial, el cual se encarga de capturar imágenes en tiempo real y analizar las características de cada camarón, como tamaño, forma y presencia de cabezas. Gracias a esta disposición, la banda transportadora actúa como un soporte dinámico y estable que garantiza la correcta alineación de los productos frente al sensor visual, optimizando la precisión del reconocimiento y clasificación automática. Una vez que el microprocesador analiza la información obtenida, envía señales de control a los actuadores, principalmente los servomotores, los cuales ejecutan movimientos precisos para desviar o clasificar los camarones según la categoría determinada por el algoritmo.(figura 22).

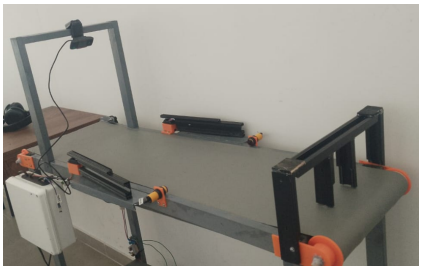


Figura 22: Banda transportadora.

2.3.2. Componetes Lógicos

Una vez descritos los componentes físicos utilizados en el desarrollo del sistema, en esta sección se detallan los componentes lógicos empleados. Estos corresponden al conjunto de software, plataformas y herramientas de programación necesarias para implementar la lógica de funcionamiento del proyecto y la integración de la visión artificial con la banda transportadora. Dentro de esta sección se incluyen:

- **Programación en Python:** Usado para la captura y preprocesamiento de imágenes, control de sensores y actuadores, implementación de la lógica de clasificación y conexión con servicios de nube e IoT.
- **Sistema de anotación de imágenes:** Una solución para generar y gestionar un conjunto de datos anotados que servirá para el entrenamiento de redes neuronales (anotación por categorías: pequeño, mediano, grande y etiquetas adicionales opcionales).
- **Plataforma para crear modelos:** Es el entorno donde se diseñan, entrenan y ponen a prueba las redes neuronales, ya sea de manera local, en la nube o mediante herramientas dedicadas para modelos más sencillos.
- **Repositorio de modelos y datos:** Sección destinada a almacenar los modelos, los conjuntos de datos y sus distintas versiones entrenads, lo que permite llevar un control ordenado y reproducción de todo el proceso de desarrollo.
- **Colección de modelos y datos:** Área donde se guardan los modelos, conjuntos de datos y sus versiones entrenadas, permitiendo mantener un control claro y reproducible del proceso de entrenamiento.

- **Plataformas de almacenamiento y visualización en la nube (IoT):** Servicios que permiten la subida remota de datos en tiempo real y la visualización de estados de sensores, actuadores y cámaras para supervisión y análisis.

2.3.2.1. Raspberry Pi Imager

Raspberry Pi Imager es una herramienta oficial desarrollada por la Fundación Raspberry Pi para simplificar la instalación del sistema operativo en placas Raspberry Pi. Esta plataforma permite a los usuarios seleccionar y escribir de forma rápida y segura el sistema operativo apropiado para su modelo Raspberry Pi directamente en la tarjeta microSD, que actúa como unidad de arranque de la tarjeta. El software ofrece una interfaz sencilla e intuitiva donde el usuario puede elegir entre varios sistemas operativos compatibles como Raspberry Pi OS, Ubuntu, LibreELEC, RetroPie, entre otros, dependiendo de las necesidades del proyecto y del modelo de Raspberry Pi utilizado. Además, Raspberry Pi Imager descarga automáticamente la última versión del sistema operativo elegido, lo que garantiza la compatibilidad del software y las actualizaciones desde el principio. Una vez instalado el sistema operativo en una tarjeta microSD usando Raspberry Pi Imager, esta tarjeta se inserta en la placa Raspberry Pi para que el sistema pueda funcionar normalmente. La configuración inicial del dispositivo se puede realizar mediante conexión directa a una pantalla o monitor compatible mediante HDMI o de forma inalámbrica mediante redes Wi-Fi o Ethernet, según el modelo de Raspberry Pi y el entorno de trabajo.

2.3.2.2. Advanced IP Scanner

Advanced IP Scanner es un software gratuito para Windows que está creado para examinar redes locales y proporcionar detalles específicos sobre cada dispositivo enlazado, como direcciones IP, direcciones MAC, nombres de usuario y, en algunos casos, la marca del equipo. Esta aplicación ayuda a usuarios y administradores de red a localizar rápidamente computadoras, impresoras, cámaras de vigilancia, enrutadores y otros instrumentos, lo que simplifica el inventario de la red, la identificación de dispositivos no autorizados y la administración eficaz de los recursos compartidos. Asimismo, ofrece características como Wake-on-LAN, acceso remoto a servicios, exploración de puertos básicos y la capacidad de generar informes, haciéndola valiosa para actividades de mantenimiento y supervisión, así como para auditorías de seguridad. Su diseño simple e intuitivo permite realizar análisis de forma rá-

pida y visualizar los resultados de manera clara, convirtiendo a Advanced IP Scanner en una herramienta confiable y útil para gestionar la red y asegurar la conexión de todos los dispositivos en línea.

2.3.2.3. REALVNC Viewer

REALVNC Viewer es una multiplataforma que permite acceder y controlar de forma remota computadoras y dispositivos como la Raspberry Pi mediante el protocolo VNC. Funciona mediante la diección IP del equipo o a través del servicio en la nube de RealVNC, permitiendo que sol usuarios autorizados puedan ingresar. Entre sus funciones principales están el control del escritorio en tiempo real, transferencia de archivos, uso compartido de portapapeles, impresión remota y ajuste automático de la resolución para mejorar el rendimiento. También incorpora medidas de seguridad como cifrado y autenticación por contraseña. Es posible con Windows, macOS, Android y iOS, lo que lo convierte en una opción práctica para mantenimiento, configuración y monitoreo de equipos a distancia, especialemente en laboratorios, entornos educativos o sistemas instalados en lugares de difícil acceso.

2.3.2.4. Python

Python es un lenguaje de programación de alto nivel y de uso general, conocido por su sintaxis simple, clara y fácil de entender, lo que lo hace perfecto tanto para nuevos usuarios como para programadores experimentados. Su capacidad de adaptación permite su aplicación en diversos sectores, que abarcan desde la ciencia de datos y la inteligencia artificial, hasta el desarrollo web, la automatización de tareas, la programación, el análisis estadístico y la creación de aplicaciones tanto móviles como de escritorio. Python cuenta con un amplio ecosistema de bibliotecas y marcos como NumPy, Pandas, Matplotlib, TensorFlow, PyTorch, Django y Flask, que simplifican la ejecución de proyectos complejos sin necesidad de codificar todo desde su inicio. Además, es un lenguaje que puede funcionar en múltiples plataformas y se opera de manera nativa en sistemas como Windows, Linux y macOS, lo que asegura su uso y adaptabilidad en diferentes entornos. Su comunidad internacional activa, junto con su flexibilidad y efectividad, han hecho de Python una herramienta vital en la investigación, el desarrollo tecnológico, la enseñanza y la automatización, posicionándose como uno de los lenguajes más solicitados y populares en la actualidad.

2.3.2.5. Edge Impulse

Edge Impulse es una plataforma especializada en el desarrollo, entrenamiento y despliegue de modelos de inteligencia artificial y redes neuronales convolucionales (CNN) para dispositivos embebidos. Su principal objetivo es facilitar la creación de modelos de machine learning capaces de ejecutarse directamente en placas de bajo consumo y capacidad limitada, como Arduino, Raspberry Pi, ESP32 y otros microcontroladores o sistemas embebidos. La plataforma permite entrenar redes neuronales con alta precisión, aprovechando datos recopilados de sensores, cámaras y otros dispositivos, para tareas como clasificación de imágenes, detección de objetos, reconocimiento de gestos o análisis de señales. Una de sus ventajas más destacadas es la interfaz intuitiva y amigable, que facilita la interacción con el usuario y agiliza el flujo de trabajo desde la captura de datos hasta la implementación del modelo. Además, Edge Impulse permite exportar los modelos entrenados en múltiples formatos, incluyendo TensorFlow Lite, C++, Arduino, COCO JSON, entre otros, garantizando compatibilidad y facilidad de integración en distintos entornos de hardware y software. Gracias a estas características, Edge Impulse se ha convertido en una herramienta fundamental para desarrollar aplicaciones de inteligencia artificial en tiempo real, optimizadas para funcionar en dispositivos con recursos limitados y aplicables en áreas como robótica, automatización industrial, visión artificial y sistemas IoT.

2.3.2.6. Tensorflow

Un marco de código abierto desarrollado por Google para construir, entrenar e implementar modelos de aprendizaje automático y redes neuronales profundas. Su arquitectura está diseñada para descubrir y descifrar patrones complejos en grandes volúmenes de datos, facilitando tareas como clasificación de imágenes, reconocimiento de voz, traducción automática, predicción estadística y análisis de datos en general. TensorFlow es altamente flexible y escalable, lo que permite la ejecución tanto en CPU como en GPU, optimizando el rendimiento para el entrenamiento de modelos complejos y el procesamiento en tiempo real. Además, es compatible con múltiples sistemas operativos, incluidos Windows, Linux y macOS, y ofrece interfaces en varios lenguajes de programación como Python, C y JavaScript, lo que facilita su integración en aplicaciones web, móviles e integradas. Gracias a su robustez, flexibilidad y extenso ecosistema de bibliotecas y herramientas adicionales, TensorFlow se ha convertido en una de las plataformas más utilizadas del mundo para desarrollar soluciones de inteligencia artificial y aprendizaje automático tanto en investigación académica como en aplicaciones industriales

y comerciales.

2.3.2.7. Tensorflow lite

TensorFlow Lite es una versión ligera y optimizada de TensorFlow, diseñada específicamente para ejecutar modelos de inteligencia artificial y aprendizaje automático en dispositivos con recursos limitados, como microcontroladores, teléfonos móviles, placas embebidas y sistemas IoT. A diferencia de TensorFlow completo, TensorFlow Lite está adaptado para minimizar el uso de memoria y consumo de energía, permitiendo que los modelos entrenados se implementen directamente en el dispositivo sin necesidad de depender de servidores externos o procesamiento en la nube. TensorFlow Lite soporta modelos de redes neuronales convolucionales (CNN), redes neuronales recurrentes (RNN) y otros tipos de arquitecturas de machine learning, ofreciendo además herramientas para optimización de modelos, como cuantización y fusión de operaciones, con el fin de reducir su tamaño y mejorar la eficiencia de ejecución. Los modelos convertidos a TensorFlow Lite pueden ser integrados en aplicaciones móviles, sistemas embebidos y dispositivos IoT, ejecutándose en CPUs de bajo consumo, GPUs móviles o aceleradores de hardware específicos como TPUs, garantizando un rendimiento adecuado incluso en entornos limitados.Gracias a estas características, TensorFlow Lite se ha convertido es una herramienta clave para el desarrollo de aplicaciones basadas en algoritmos avanzados en tiempo real, donde la eficiencia, velocidad y autonomía del dispositivo son esenciales, como en sistemas de visión artificial, reconocimiento de voz, detección de objetos y control automatizado.

2.3.2.8. OpenCV

OpenCV es una librería de código abierto que se utiliza extensamente en aplicaciones relacionadas con la visión por computadora, el procesamiento de imágenes y el aprendizaje automático. Inicialmente creada por Intel y actualmente mantenida de manera colaborativa en todo el mundo, OpenCV ofrece una amplia gama de herramientas y funciones que facilitan la captura, análisis, modificación y procesamiento de imágenes y videos en tiempo real. Sus aplicaciones abarcan la detección de objetos, el reconocimiento facial, el seguimiento de movimientos, la segmentación de imágenes, el análisis de patrones, la calibración de cámaras y la realidad aumentada, además de ofrecer soporte para algoritmos de aprendizaje automático en el análisis de datos visuales. OpenCV es compatible con diversos lenguajes de programación, tales como Python, C++, Java y más, y puede ser ejecutado en múltiples sistemas

operativos como Windows, Linux y macOS, lo que lo hace fácil de integrar en proyectos de robótica, inteligencia artificial, automatización industrial y aplicaciones móviles. Su adaptabilidad, eficacia y una activa comunidad la han establecido como una de las herramientas más relevantes para el desarrollo de sistemas avanzados de visión por computadora en tiempo real.

2.3.2.9. Firebase

Firebase es una plataforma de desarrollo de aplicaciones en la nube, creada por Google, que proporciona una amplia gama de servicios backend listos para usar, lo que permite a los desarrolladores centrarse en la funcionalidad de la aplicación sin preocuparse por la infraestructura del servidor. Entre sus principales características se incluyen bases de datos en tiempo real, almacenamiento de archivos, autenticación de usuarios, hosting, notificaciones push, análisis de rendimiento y monitorización de aplicaciones, así como herramientas de machine learning e integración con otros servicios de Google Cloud. Firebase permite que los datos de una aplicación se sincronicen en tiempo real entre múltiples dispositivos, facilitando la creación de aplicaciones colaborativas, chats, sistemas de monitoreo o cualquier proyecto que requiera actualizaciones instantáneas. Su compatibilidad con Android, iOS, Web y aplicaciones multiplataforma lo convierte en una solución versátil para distintos tipos de proyectos. Además, ofrece seguridad integrada mediante reglas de acceso y autenticación, lo que garantiza que los datos estén protegidos frente a accesos no autorizados. Gracias a su facilidad de integración, escalabilidad y servicios preconfigurados, Firebase se ha consolidado como una herramienta ideal tanto para desarrolladores independientes como para equipos profesionales, permitiendo acelerar el desarrollo de aplicaciones modernas, optimizar la experiencia de los usuarios y mantener la infraestructura de backend confiable y eficiente sin necesidad de administrar servidores propios.

2.4. Diseño experimental

Para el desarrollo de la clasificadora automática de camarones en la banda transportadora, se seleccionaron diversos componentes electrónicos, como sensores, que emiten señales de detección hacia la placa Raspberry Pi. Asimismo, se incorporó una cámara, que permite detectar en tiempo real el tamaño de los camarones, controlando automáticamente el estado de los servomotores. Cuando los camarones pasan por la banda, son clasificados de forma automática según la programación asignada a los servomotores.

Todo el desarrollo se implementó en python, utilizando las librerías necesarias para cubrir las distintas funciones del sistema. El código se escribió en Geany, un editor compatible con Raspberry Pi que facilita la detección de errores de sintaxis y mejora el flujo de trabajo. Además, se configuró un entorno virtual para gestionar dependencias y módulos, lo que permitió instalar los paquetes requeridos sin afectar el sistema operativo principal y evitar conflictos o fallos durante la ejecución y compilación del software.

Para entrenar el modelo de clasificación se establecieron tres categorías de camarón según su tamaño: pequeño, mediano y grande. Las imágenes se obtuvieron con una cámara web, recopiladoo alrededor de 800 fotografías por clase. En cada grupo se incluyeron camarones con y sin cabeza y adeás se añadieron otras 800 imágenes de fondo con distintas variaciones.

Las imágenes fueron cargadas a la plataforma Edge Impulse, donde se realizó el etiquetado manual de cada muestra. En este caso, se definieron siete clases distintas:

- `PEQUENO_SIN_CABEZA`
- `PEQUENO_CON_CABEZA`
- `MEDIANO_CON_CABEZA`
- `MEDIANO_SIN_CABEZA`
- `GRANDE_CON_CABEZA`
- `GRANDE_SIN_CABEZA`
- `FONDO`

Esta plataforma resultó óptima para el desarrollo del modelo de clasificación destinado a ejecutarse en la Raspberry Pi 4.

Finalmente, se desarrolló una interfaz de monitoreo de datos en tiempo real utilizando Firebase para ver el estado de las variables del sistema y monitorear el funcionamiento de las cintas transportadoras y los servomotores.

La sección que sigue exponene las características del proyecto, separadas en componentes de hardware y software, lo que facilita examinar con mayor precisión cada parte y el rol que cumple dentro del sistema.

2.4.1. Diseño Estructural

En la Figura 23 se presenta un diseño realizado en Fusion 360, con el fin de seguir un orden específico del ensamblaje final de la máquina clasificadora.

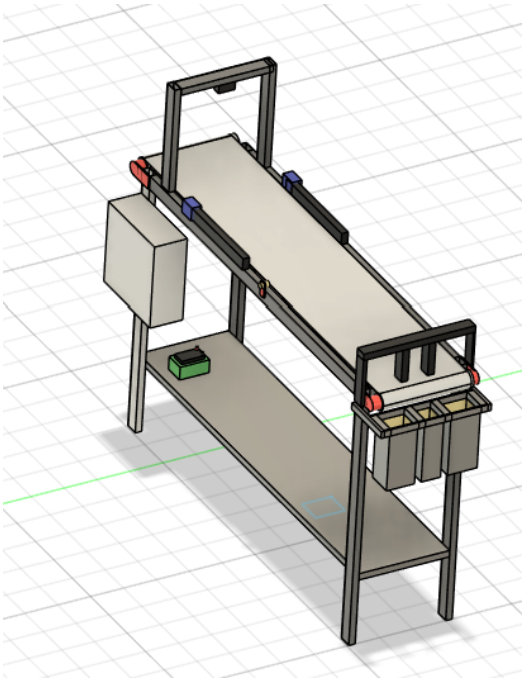


Figura 23: Diseño realizado en fusión 360 para la implementación del sistema
.

El diseño consiste en poder seguir un esquematico para realizar la estructutra de la máquina, esta cuenta con motor con engranejes reductores, caja para proteger el circuito eléctrico realizado, servomotores para desplazar los camarones segun la clase que detecte el modelo de red neuronal entrenado, web cam para que el modelo vea la clase que esta pasado y la detecte segun como se corresponda, la banda transportadora que esta se movera por medio del motor con engranajes recductores que sera alimentado por una fuente de 12V DC y finalmente unos envases donde los camarones que seran separados por clases llegaran a estar en esos contenedores.

2.4.2. Diseño hardware

El hardware constituye una parte fundamental del sistema, ya que se complementa con el software; sin él, no sería posible realizar programación ni el envío de datos. Se diseñó un esquema de conexiones utilizando el programa de simulación Proteus, cuyo objetivo principal fue integrar los sensores infrarrojos E18-D80NK con la placa TXS0108E, la cual trabaja exclusivamente con señales digitales. Esta placa permite que la señal de voltaje de los sensores, que es de 5V, se regule a 3.3V, voltaje de operación de los pines GPIO

de la Raspberry Pi, tanto para entradas como para salidas.

En este proyecto se emplearon dos placas TXS0108E: una dedicada a los sensores y otra específica para los actuadores, que en este caso son los servomotores. Para el funcionamiento de los sensores infrarrojos, se utilizaron capacitores de baja capacitancia, los cuales ayudan a reducir significativamente el ruido generado por los sensores. En momentos en que no se detecta ningún objeto, los sensores enviaban señales falsas; para corregir esto, se colocaron capacitores desde GND hasta la entrada de señal en la placa, eliminando así las falsas detecciones. Cabe destacar que en los Arduinos este problema no ocurre, ya que incluyen capacitores específicos para este tipo de errores.

Para poner en funcinamiento la placa TXS0108E, se conecta los pines VB y OE , lo que permite operar como interfaz de entrada hacia la Raspberry Pi. De esta forma, cuando el sensor genrra una señal de 5v, la placa la reduce a 3.3V para los pines de entrada de la Raspberry. En cuanto a los servomotores, no fue necesario realizar puentes adicionales, ya que la placa trabaja de forma directa: recibe la señal por el puerto VA y la entrega a 5v por el puente VB para alimentar correctamente a los actuadores.

El sistema también cuenta con una cámara web, utilizada tanto para la adquisición de imágenes (dataset) como para la detección en tiempo real mediante visión artificial. La cámara fue seleccionada por su fácil conexión mediante puerto USB. Además, se incorporó un ventilador específico para la Raspberry Pi, con el fin de evitar el sobrecalentamiento de la placa, dado que las tareas en ejecución en tiempo real generan calor, lo cual podría dañar algún componente.

Las fuentes de alimentación externas suministran 12V a 5V, ya que estos voltajes no son generados directamente por la Raspberry Pi, Aun así, se utilizó el mismo GND de la Raspberry pi como punto de referencia común, garantizando que la placa pueda interpretar adecuadamente las señales eléctricas.

2.4.2.1. Conexiones hacia la placa Raspberry pi 4

A continuacion se detallara la conexión de la placa Rasbperry con la cámara web-cam y con los modulo TXS0108E los cuales consisten en la conexiones de los pines GPIO de la Raspberry con los modulo, tambien con la alimentación de 3.3v y el punto de referencia común en La placa de expansion incluye puertos y pines adicional, mas detalles se presentaran en el cuadro 8 y la figura 24.

Cuadro 8: Dispositivos conectados a los pines GPIO y USB de Raspberry Pi 4

Pin Raspberry Pi	Componente	Función
3.3V	Módulos TXS0108E (VA)	Alimentación de 3.3V para los puertos VA de ambos módulos TXS0108E, permitiendo trabajar con los niveles de voltaje de la Raspberry Pi
GND	Módulos TXS0108E y fuente 5V	Conexión común de tierra para referencia de voltaje en todo el circuito
GPIO 5	Entrada A1 TXS0108E (Servomotores)	Control del Servomotor 1 mediante señales de 3.3V
GPIO 6	Entrada A2 TXS0108E (Servomotores)	Control del Servomotor 2 mediante señales de 3.3V
GPIO 22	Entrada A1 TXS0108E (Sensores)	Recepción de datos del Sensor 1 (3.3V)
GPIO 27	Entrada A2 TXS0108E (Sensores)	Recepción de datos del Sensor 2 (3.3V)
USB 3.0	Webcam	Adquisición de imágenes para construcción de dataset y detección en tiempo real

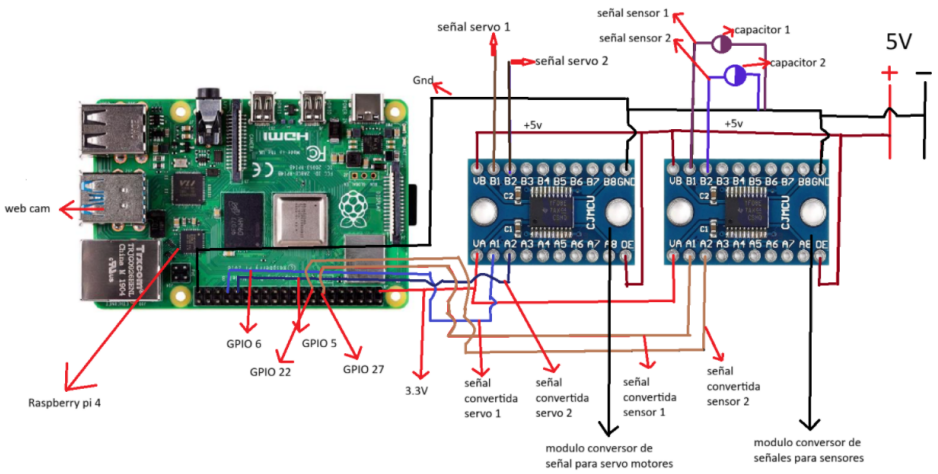


Figura 24: Diseño del circuito para conexiones hacia la Raspberry pi con los modulos.

2.4.2.2. Módulo TXS0108E con sensores

Se utilizó uno de los dos módulos TXS0108E para la adaptación de las señales provenientes de los sensores infrarrojos. Estos sensores, al operar, generan cierto nivel de ruido eléctrico que puede provocar falsas detecciones o interferencias en la señal. Para mitigar este efecto, se acoplaron capacitores de baja capacitancia entre la entrada de señal y el GND del módulo.La

función de estos capacitores es filtrar las fluctuaciones no deseadas y estabilizar la señal digital enviada a la Raspberry Pi, asegurando que únicamente se transmitan las señales reales de detección de objetos. De esta manera, el TXS0108E actúa no solo como un adaptador de niveles de voltaje reduciendo la señal de 5V de los sensores a 3.3V compatible con los pines GPIO, sino también como un filtro parcial del ruido eléctrico, mejorando la confiabilidad y precisión del sistema de detección. Este procedimiento es fundamental en sistemas de visión o control en tiempo real, ya que las señales limpias permiten que la lógica de programación interprete correctamente la información, evitando errores de lectura y posibles fallas en la actuación de los servomotores conectados al sistema comoc se detalla en el cuadro 9 y la figura 25.

Cuadro 9: Conexión de sensores a los pines del TXS0108E

Pin TXS0108E	**Componente**	**Función**
VA	3.3V Raspberry Pi	Alimentación del puerto VA (3.3V) para el correcto funcionamiento del módulo
A1	GPIO 22 (RPi)	Recepción del estado digital del Sensor 1 (3.3V)
A2	GPIO 27 (RPi)	Recepción del estado digital del Sensor 2 (3.3V)
VB	OE TXS0108E	Habilitación del módulo (conversión de 5V a 3.3V entre puertos B y A)
B1	Señal Sensor 1 + Capacitor 1µF	Conexión de la señal del Sensor 1 con filtro RC (1µF a GND) para eliminar ruido eléctrico
B2	Señal Sensor 2 + Capacitor 1µF	Conexión de la señal del Sensor 2 con filtro RC (1µF a GND) para eliminar ruido eléctrico

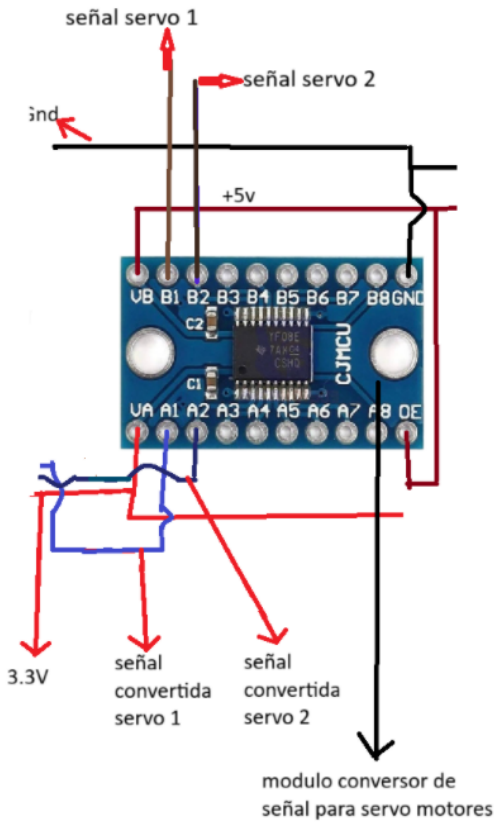


Figura 25: Diseño del circuito para conexiones de modulo con señal de sensores.

2.4.2.3. Módulo TXS0108E con actuadores

El segundo módulo se encarga de manejar las señales que controlan los servomotores, ajustando las salidas de 3.3V de la Raspberry pi a los 5V necesarios para que los actuadores funcionen de manera correcta y segura.

Además, se añadió una fuente externa de 5V para asegurar una alimentación estable, evitando que las variaciones producidas por los servomotores afecten a la Raspberry Pi y aumentado la confiabilidad del sistema.

El TXS0108E actúa como un puente bidireccional de nivel lógico, asegurando que las señales de control sean precisas y consistentes, y que los servomotores respondan de manera eficiente a las instrucciones generadas por el sistema de detección en tiempo real. De esta forma, se logra un control efectivo de los actuadores sin comprometer la integridad de la Raspberry Pi ni del circuito eléctrico, como se detalla en el cuadro 10 y la imagen 26.

Cuadro 10: Conexión de servomotores a los pines del TXS0108E

Pin TXS0108E	**Componente**	**Función**
VA	3.3V Raspberry Pi	Alimentación del puerto A (3.3V) para compatibilidad con niveles lógicos de la RPi
A1	GPIO 5 (RPi)	Envío de señal PWM (3.3V) para control del Servomotor 1 mediante visión artificial
A2	GPIO 6 (RPi)	Envío de señal PWM (3.3V) para control del Servomotor 2 mediante visión artificial
VB	Fuente externa 5V	Habilitación del módulo y alimentación del puerto B (5V) para los servomotores
B1	Señal Servomotor 1	Conexión al cable de control del Servomotor 1 (convertido de 3.3V a 5V)
B2	Señal Servomotor 2	Conexión al cable de control del Servomotor 2 (convertido de 3.3V a 5V)

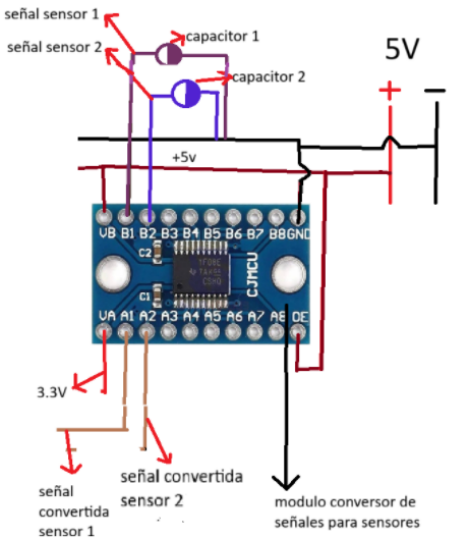


Figura 26: Diseño del circuito para conexiones del modulo con señal de servo motores .

2.4.2.4. Implementación del sistema directo

Se utilizó una fuente de alimentación externa de 5V para energizar de manera confiable los sensores, los actuadores y los módulos TXS0108E. Esto se debió a que, al alimentar todos los componentes directamente desde la Raspberry Pi, se presentaba una caída de voltaje que podía ocasionar el cierre inoportuno de los programas ejecutados en la placa, afectando la estabilidad del sistema.

La fuente de alimentación externa garantiza un suministro constante y suficiente corriente, evita caídas de tensión y protege tanto la Raspberry Pi como otros componentes del circuito. Esto garantiza que los sensores infrarrojos funcionen correctamente, que los servomotores obtengan la potencia que necesitan para funcionar sin problemas y que los módulos TXS0108E puedan ajustar eficazmente las señales de voltaje.

Asimismo, al mantener una referencia de GND común con la Raspberry Pi, la fuente externa permite que todas las señales digitales sean interpretadas correctamente por la placa controladora, asegurando la precisión en la comunicación entre sensores, actuadores y el sistema de control. Esta estrategia de alimentación contribuye significativamente a la estabilidad y confiabilidad del sistema en aplicaciones de detección y control en tiempo real, los detalles estaran en el cuadro 11 y la imagen 27.

Cuadro 11: Conexión de alimentación del sistema

Polaridad	**Componentes conectados**	**Función**
Positivo (+)	• Ventilador • Sensores 1, 2 • Servomotores 1, 2 • Pines VB de módulos TXS0108E	Alimentación principal del sistema (5V/12V según requerimiento)
GND (Negativo)	• Ventilador • Sensores 1, 2 • Servomotores 1, 2 • Pines GND de módulos TXS0108E • Raspberry Pi (tierra común)	• Referencia común de tierra • Completa el circuito eléctrico • Estabiliza señales de entrada/salida • Conexión única para evitar bucles.

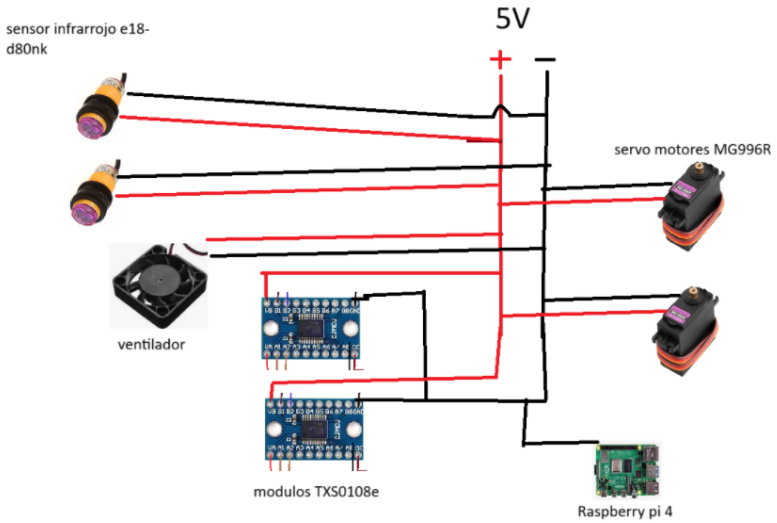


Figura 27: Diseño del circuito para conexiones de 5V .

Se utiliza una batería de 12V para alimentar el motor de la cinta transportadora, permitiendo regular su velocidad mediante un regulador llamado transistor LM412 y un potenciómetro ajustable. el motor empleado es un motorreductor, adecuado para mover cargas a baja velocidad con buen par, que lo hase ideal para el funcionamiento estable de la banda. la fuente de 12V asegura la potencia necesaria, mientras que el regulador permite adaptar la velocidad sin afectar el desempeño ni generar esfuezos innecesariso en el motor. Además, el ajusta mediante potenciómetros facilita modificar la velocidad en tiempo real y mantener un funcionamiento seguro y constante del sistema.

Se utiliza una fuente de 12V para alimentar el moto de la banda transportadora, lo que permite regular su velocidad de forma controlada. Para ello se incorporó un regulador basado en un LM412 y pontenciómetro ajustable, que permite modificar el voltaje de salida y así ajustando con precisión la velocidad del motor.

El motor empleado es un motorreductor, idela para trabajos que requieren alto par a bajas revoluiones, por lo que resulta adecuado para movr una cinta transportadora con estabilidad y durabilidad. La alimentación directa de 12V asegura el suministro energértico necesario, mientras que el regulador permite adaptar la velocidad a las condiciones del proceso sin afectar el par disponible.

Este sistema no solo brinda un control más exacto del movimiento de la cinta, sino que también protege el motor frente a sobrecargas y evita cambios bruscos que podrían interferir con sensores o actuadores, Además, el uso de potenciómetros facilita según las necesidades operativas.

2.4.3. Implementación del software

Esta sección describe el desarrollo del software necesario para operar el hardware introducido anteriormente. El diseño se divide en cuatro etapas principales:

Obtención de datos: Grabación de señales e imágenes de sensores mediante una cámara web, generando un conjunto de datos limpio y confiable para el entrenamiento.

Entrenamiento de la red neuronal: Desarrollo y optimización de un modelo capaz de clasificar los objetos detectados, ajustado para ejecución eficiente en la Raspberry Pi 4.

Programación en la Raspberry Pi 4: Integración del modelo con los sensores y contol de los servomotores, permitiendo que el sistema responda

de manera inmediata en tiempo real.

Envío de información a la nube: Registro y monitoreo remoto de los datos para su análisis y elabroación de reportes. Esta metodología garantiza una comunicación continua entre los dispositivos y las aplicaciones, manteniendo la precIsión, eficiencia y confiabilidad en el funcionamiento del sistema.

2.4.3.1. Obtención de datos

Para generar un dataset adecuado para el proyecto, se realizaron investigaciones en diversas fuentes de datos públicas, como Roboflow, Kaggle y Data.World. Sin embargo, no se encontró un conjunto de datos específico que cumpliera con los requerimientos del sistema, por lo que se decidió crear un dataset propio.

La creación del dataset se llevó a cabo utilizando varias librerías de Python para la captura y procesamiento de imágenes, así como la plataforma Edge Impulse, que facilita la recopilación, etiquetado y gestión de datos para el entrenamiento de modelos de aprendizaje automático. La estrategia incluyó la captura de imágenes con la cámara web conectada a la Raspberry Pi, asegurando que las condiciones de iluminación, ángulo y distancia fueran representativas del entorno real en el que operará el sistema. El dataset resultante garantiza que la red neuronal pueda aprender de manera efectiva, mejorando la precisión en la detección y clasificación durante la operación en tiempo real del sistema de visión artificial, las librerias con sus respectivas versiones s edetallaran en el cuadro 12.

Cuadro 12: Paquetes y versiones utilizados

Paquete	**Versión**
numpy	2.2.4
opencv-contrib-python	4.11.0.86
pip	25.0.1
Paquete	**Versión**
Edge Impulse	pip3 install edge-impulse-linux

A continuación se describira la utilizacion de estas librerias en el cuadro 13.

Cuadro 13: Librerías instaladas en el sistema Raperry pi para obtencion de datos

Librería	**Descripción**
numpy	Utilizada para cálculo numérico avanzado en Python, incluyendo operaciones con matrices y arreglos multidimensionales.
opencv-contrib-python	Un paquete completo para procesamiento de imágenes y visión por computadora, que incluye módulos de inversión adicionales.
pip	El sistema de administración de paquetes predeterminado para Python, utilizado para instalar y administrar bibliotecas y dependencias.
pip3 install edge-impulse-linux	instalar el SDK de Edge Impulse en sistemas Linux, como la Raspberry Pi 4, usando Python 3, para ejecutar y cargar modelos ya entrenados de CNN.

Para crear este conjunto de datos se instalaron algunas bibliotecas específicas de Python y un compilador para este lenguaje, lo que permitió desarrollar scripts para la adquisición y procesamiento de imágenes. Además, se utilizó la plataforma Edge Impulse, que facilita la recopilación, el etiquetado y la gestión de conjuntos de datos para el entrenamiento de modelos de aprendizaje automático.

Después de que la plataforma fue instalada, se inició un nuevo proyecto enfocado en la categorización de objetos. Para conectar la Raspberry Pi 4 con Edge Impulse, se empleó el siguiente comando: sudo npm install -g edge-impulse-cli.

Este comando habilita el acceso desde la Raspberry Pi y permite usar la cámara incorporada para capturar imágenes directamente desde el dispositivo. Al iniciar el servicio de Edge Impulse en la Raspberry Pi, la cámara aparece de inmediato en la plataforma, lo que implica la supervición y el control del proceso de captura.

Durante la elaboración del conjunto de datos, se recolectaron 800 fotos de cada categoría de camarón en el siguiente orden:

Gamba pequeña, con y sin cabeza.

Gamba mediana, con y sin cabeza.

Gamba grande, con y sin cabeza.

Fotos de varios fondos para aumentar la variedad del conjunto de datos.

A lo largo de la filmación, los camarones fueron situados en varias configuraciones y zonas dentro del ángulo de la cámara, lo que generó una variedad

de datos para entrenar y comprobar de manera efectiva la red neuronal. Se definieron 7 clases o etiquetas para el proyecto:

pequeño_con_cabeza
mediano_con_cabeza
grande_con_cabeza
pequeño_sin_cabeza
mediano_sin_cabeza
grande_sin_cabeza
fondo

En total, se obtuvo un dataset de 5600 imágenes, el cual proporciona suficiente diversidad y cantidad de datos para entrenar un modelo de clasificación confiable y robusto para la detección de camarones en tiempo real.

2.4.3.2. Entrenamiento de la red neuronal

En este apartado se detalla el modelo de red neuronal entrenado para la detección y clasificación de camarones. El modelo utilizado fue desarrollado en la plataforma Edge Impulse, aprovechando sus herramientas de adquisición de datos, preprocesamiento y entrenamiento de modelos de aprendizaje automático.

Para la arquitectura de red neuronal se eligió MobileNetV1, una red liviana y eficiente diseñada específicamente para aplicaciones en dispositivos con recursos limitados como Raspberry Pi 4. MobileNetV1 utiliza convoluciones separables profundas, lo que reduce significativamente la cantidad de parámetros y pasos requeridos mientras mantiene un buen nivel de precisión de clasificación.

El entrenamiento se realizó utilizando nuestro propio conjunto de datos que consta de imágenes de camarones en diferentes tamaños y estados (con y sin cabeza) y diferentes fondos. Durante el proceso, se optimizaron hiperparámetros como la tasa de aprendizaje, el número de épocas y el tamaño del lote para obtener un modelo preciso y eficiente capaz de realizar inferencias en tiempo real.

Este diseño permite que la Raspberry Pi identifique camarones de manera eficiente en condiciones reales, logrando un equilibrio adecuado entre velocidad de procesamiento y precisión, aspectos esenciales para el buen funcionamiento de un sistema automatizado de clasificación.

MobileNetv1 La arquitectura empleada para el diseño del modelo de clasificación de camarones corresponde a MobileNetV1, una red neuronal convolucional (CNN) optimizada para dispositivos embebidos y entornos con

recursos computacionales limitados, como la Raspberry Pi 4. Para poder realizar el entrenamiento y validacion para la creacción de la red se divide los datos en un 80 % y 20 % el cual de las 10995 imagenes, 8796 imagenes son para validacion y 2199 son para entramiento, teniendo en cuenta que para cada clase se divide tambien en entrenamiento y validacion osea 1310 para entrenamiento y y 330 para validación como se ve en la figura 28.

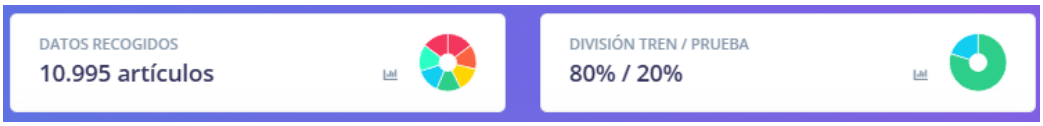


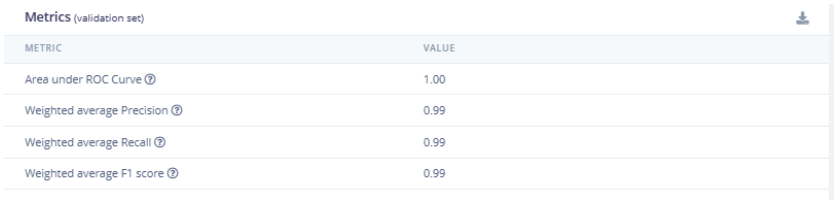
Figura 28: Recoleción de datos y porcentaje para validacion y entrenamiento.

El modelo fue diseñado y entrenado en la plataforma Edge Impulse, utilizando un conjunto de datos conformado por siete clases o etiquetas que ya se han mencionado anteriormente en los apartados. Cada etiqueta representa una variación visual significativa del camarón según su tamaño y presencia de cabeza, permitiendo al sistema realizar una clasificación precisa en tiempo real.

El entrenamiento se realizó con los siguientes parámetros: 20 ciclos de entrenamiento, una tasa de aprendizaje (learning rate) de 0.0005, tamaño de lote (batch size) de 32, y un conjunto de validación del 20 %. Se activó la opción de data augmentation para mejorar la generalización del modelo ante variaciones de iluminación, posición o rotación del camarón. Además, se generó un perfil cuantizado (int8) para reducir el tamaño del modelo y optimizar su inferencia en hardware de bajo consumo.

La arquitectura implementada presenta una capa de entrada de 96×96 píxeles con tres canales de color (RGB), seguida de una secuencia de bloques convolucionales basados en MobileNetV1 con un factor de ancho(α)de0,25.

Una capa densa final de 64 neuronas con una tasa de dropout de 0.5, la cual previene el sobreajuste del modelo. La capa de salida está compuesta por 7 neuronas correspondientes a las siete clases definidas, con activación Softmax, que proporciona las probabilidades de pertenencia de la imagen a cada categoría.

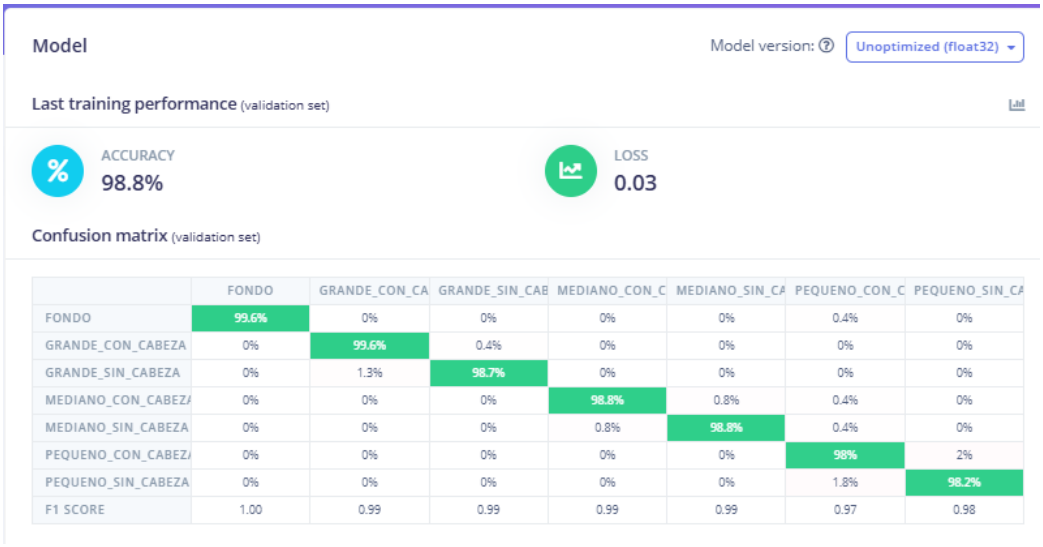


Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.99
Weighted average Recall	0.99
Weighted average F1 score	0.99

Figura 29: Matriz de validación.

Durante la fase de validación como en la figura 29, el modelo alcanzó un desempeño sobresaliente, obtenido un aprecisión del 98.8 % y una pérdida de 0.03. Además, las métricas de evaluación mostraron un Area bajo la curva ROC de 1.00, una precisión ponderada de 0.99, un recall ponderado de 0.99 y un F1 Score global de 0.99, evidenciando un buen equlibrio entre sensibilidad y exactitud en la clasificación.



Model

Model version: Unoptimized (float32)

Last training performance (validation set)

ACCURACY 98.8%

LOSS 0.03

Confusion matrix (validation set)

	FONDO	GRANDE_CON_CA	GRANDE_SIN_CAB	MEDIANO_CON_C	MEDIANO_SIN_CA	PEQUENO_CON_C	PEQUENO_SIN_CA
FONDO	99.6%	0%	0%	0%	0%	0.4%	0%
GRANDE_CON_CABEZA	0%	99.6%	0.4%	0%	0%	0%	0%
GRANDE_SIN_CABEZA	0%	1.3%	98.7%	0%	0%	0%	0%
MEDIANO_CON_CABEZA	0%	0%	0%	98.8%	0.8%	0.4%	0%
MEDIANO_SIN_CABEZA	0%	0%	0%	0.8%	98.8%	0.4%	0%
PEQUENO_CON_CABEZA	0%	0%	0%	0%	0%	98%	2%
PEQUENO_SIN_CABEZA	0%	0%	0%	0%	0%	1.8%	98.2%
F1 SCORE	1.00	0.99	0.99	0.99	0.99	0.97	0.98

Figura 30: Conjunto de validación.

La matriz de confusión mostró en la figura 30 una correcta clasificación en la mayoría de las categorías, destacando un 99.6 % de acierto en la clase FONDO y valores superiores al 97 % en las clases utilizadas y ya mencionadas anteriormente del modelo, demostrando la efectividad de la red en la identificación de las distintas morfologías de camarón. El modelo final se exportó en formatos TensorFlow Lite (tflite) y Edge Impulse (.eim) para su implementación en Raspberry Pi 4, donde se ejecuta a través de un entorno virtual Python que contiene todas las dependencias necesarias, conectado a una cámara de alta resolución que permite la clasificación de camarones en

tiempo real y con baja latencia. Gracias a su estructura optimizada, MobileNetV1 forma un núcleo eficiente y escalable del sistema de visión artificial propuesto, que permite automatizar el proceso de clasificación de camarones en función de su tamaño y características físicas con alta precisión y bajo costo computacional.

2.4.3.3. Programación en la Raspberry Pi 4

Para garantizar el funcionamiento integral del sistema de clasificación automática de camarones, se configuró un entorno virtual de Python dentro de la Raspberry Pi 4, el cual permitió aislar todas las dependencias y mantener un entorno controlado, estable y reproducible. Este entorno se creó con la finalidad de ejecutar el modelo neuronal entrenado en Edge Impulse, procesar imágenes en tiempo real, controlar actuadores electromecánicos y desplegar una interfaz de monitoreo (HMI).

En primera instancia, se implementaron las librerías que permiten la carga, interpretación y ejecución del modelo neuronal exportado desde Edge Impulse en formato .eim. Estas dependencias, mostradas en la Tabla 1, son fundamentales para realizar la inferencia local del modelo en la Raspberry Pi 4 sin necesidad de conexión a internet.

Cuadro 14: Paquetes y versiones utilizados

Librería / Paquete	**Descripción técnica**
edge Impulse Linux	Herramienta oficial de Edge Impulse para ejecutar modelos de ML en dispositivos Linux.
tensorflow	Framework de aprendizaje profundo desarrollado por Google.
tflite-runtime	Versión ligera de TensorFlow Lite optimizada para sistemas embebidos.
numpy	Biblioteca de cálculo numérico de alto rendimiento.
pandas	Biblioteca para el análisis de data estructurada.
matplotlib	Herramienta de visualización gráfica en Python.
scipy (opcional)	Funciones avanzadas para cálculos numéricos y estadísticas.

En el cuadro 14 se describen varias librerias pero la librería principal edge_impulse_linux proporciona las herramientas necesarias para ejecutar modelos Edge Impulse optimizados con TensorFlow Lite, permitiendo leer directamente el archivo .eim y obtener los resultados de clasificación en tiempo real. En conjunto con tensorflow y tflite-runtime, se habilita el entorno de ejecución de redes convolucionales MobileNetV1, optimizadas para hardware embebido.

Otras bibliotecas como numpy, pandas y matplotlib apoyan el desarrollo al proporcionar herramientas eficientes para gestionar datos, analizar métricas y visualizar resultados. Gracias a estas librerías es posible crear gráficos de precisión, pérdida y rendimiento del modelo durante las pruebas, lo que facilita evaluar de forma cuantitativa la red neuronal y comprobar su funcionamiento dentor del sistema embebido.

En conjunto, este primer grupo de librerías forma la base del componente de inteligencia artificial del proyecto, garantizando una ejecución liviana, eficiente y totalmente compatible con el hardware de la Raspberry Pi 4.

Cuadro 15: Librerías para control de servomotores, sensores y diseño de la plataforma (HMI)

Librería / Paquete	**Descripción técnica**	**Uso principal**
RPi.GPIO	Control de pines GPIO nativos de la Raspberry Pi.	Control directo de servos, sensores.
pigpio	Biblioteca avanzada para manejo de señales PWM con alta precisión.	Movimiento preciso de servomotores.
gpiozero (opcional)	API de alto nivel para controlar hardware conectado a la Raspberry.	Control simplificado de dispositivos electrónicos.
tkinter	Librería estándar de Python para crear interfaces gráficas (GUI).	Diseño básico de la pantalla HMI.
PyQt5 o customtkinter (opcional)	Herramientas modernas para el diseño de interfaces gráficas avanzadas.	Implementación visual del panel HMI táctil.
time	Módulo estándar para temporización y retardos.	Sincronización en los movimientos de actuadores.
os y sys	Librerías base del sistema operativo.	Control de archivos, procesos y comunicación interna.

El segundo conjunto de librerías, detallado en el cuadro 15, se utilizó para implementar la lógica de control físico y de supervisión visual del sistema automatizado. Estas dependencias permiten interactuar directamente con los componentes electrónicos implementados en el trabajo d etitulación, tales como servomotores y sensores, a través de los pines GPIO (General Purpose Input/Output) de la Raspberry Pi.

las bibliotecas RPI.GPIO y piggpio se emplean como herramientas principales para controlar los servomotores según su categoría. Estas permiten generar señales PWM (Modulación por ancho de pulso) con la precisión necesaria para ajustar el Angulo de giro de cada servo con la clasificación obtenida por el modelo neuronal.

Por otro lado, la implementación de librerías como gpiozero, time, y los módulos os y sys, facilita la comunicación entre los procesos de hardware y el sistema operativo, permitiendo realizar retardos, calibraciones y sincronizaciones durante la operación de la banda transportadora.

Asimismo, se incorporaron librerías para el desarrollo de interfaces de usuario que proporcionan una representación clara del sistema operativo. En esta ocasión, se emplearon tkinter y PyQt5 para crear una interfaz visual que exhibe las predicciones del modelo, el estado de los actuadores y posibilita el control manual del sistema cuando se requiera. Esta interfaz puede aparecer en una pantalla táctil vinculada a una Raspberry Pi y facilita la interacción inmediata entre el usuario y el sistema de visión artificial.

Gracias a esta serie de bibliotecas, el proyecto consigue combinar la inteligencia artificial con el hardware tangible, posibilitando no solo la categorización automática de los camarones, sino también su manipulación mecánica exacta en la plataforma de experimentación.

Cuadro 16: Librerías para visión artificial y manipulación de imágenes

Librería / Paquete	**Descripción técnica**
opencv-python	Biblioteca de visión artificial de código abierto (OpenCV).
picamera2 (según versión de OS)	Interfaz Python para cámaras CSI o USB en Raspberry Pi OS Bullseye o superior.
imutils	Conjunto de funciones útiles para operaciones de imagen con OpenCV.
Pillow (PIL)	Biblioteca para manipulación de imágenes en diversos formatos.
cvzone (opcional)	Extensión sobre OpenCV para visualización avanzada (FPS, bounding boxes, texto, etc.).
threading (opcional)	Control de procesos simultáneos (multithreading).

Finalmente, el tercer grupo de librerías, expuesto en el cuadro 16, corresponde a los componentes encargados de la adquisición y procesamiento de imágenes en tiempo real, etapa esencial para la clasificación visual de los camarones.

La biblioteca opencv-python (OpenCV) constituye el núcleo de este proceso y ofrece herramientas avanzadas de visión por computadora para la adquisición, el preprocesamiento y el análisis de imágenes en la cámara. Utilizando OpenCV, el sistema realiza una lectura continua de fotogramas, aplica transformaciones de color y tamaño y envía las imágenes procesadas a un modelo neuronal para su clasificación.

Para la comunicación directa con cámaras CSI o USB, se utilizó la librería picamera2, compatible con las versiones más recientes del sistema operativo Raspberry Pi OS (Bullseye y superiores). Esta librería ofrece un control completo de la resolución, balance de blancos, exposición y frecuencia de captura.

Tambien se utilizan las librerias PILLOW (PIL) e imutils para realizar tareas adicionales como cambios de formato, rotacion, recorte y ajuste de brillo o contraste, lo cual mejora la calidad de las imagenes que recibe el modelo. de igual manera la herramientas como cvzone y threards permite incorporar superposiciones graficas, elementos visuales y procesamiento en paralelo, contribuyendo a que e sistema funcione con mayor fluidez durante la interferencia.

En conjunto, este conjunto de bibliotecas asegura una obtención cons-

tante y confiable de imágenes, además de un procesamiento visual ágil que posibilita una clasificación precisa y de baja latencia de los camarones en tiempo real.

2.4.3.4. Envío de datos a la nube

Con el propósito de centralizar, almacenar y visualizar los datos generados por el sistema de clasificación automática de camarones, se implementó un módulo de comunicación en la Raspberry Pi 4 que permite enviar la información procesada hacia la plataforma en la nube Firebase, perteneciente al ecosistema de Google Cloud.

Esta capa de comunicación tiene un papel fundamental en la estructura general del sistema, funcionando como un vínculo entre la inteligencia artificial que opera localmente y la supervisión a distancia, garantizando que los resultados de la categorización, las métricas de rendimiento y los estados de los actuadores se registren instantáneamente y puedan ser consultados desde cualquier dispositivo con acceso a Internet.

Los datos estructurados se convierten a formato JSON y se envían mediante solicitudes HTTP seguras (HTTPS) utilizando las librerías de Python integradas en el entorno virtual. Firebase recibe esta información en su base de datos Realtime Database, la cual sincroniza los valores instantáneamente en la nube.

La información registrada puede visualizarse mediante una interfaz web creada con HTML, CSS y JavaScript, donde se muestren los datos más recientes, las clases detectadas y las métricas de desempeño del sistema. Esta plataforma puede abrirse desde cualquier navegador, ya sea en una computadora, una tablet o un teléfono móvil.

Este flujo de trabajo permite que el sistema opere de forma local e independiente en la Raspberry Pi, pero a la vez mantenga una sincronización constante con la nube, facilitando la supervisión, trazabilidad y análisis histórico de las clasificaciones realizadas.

Para establecer la comunicación entre la Raspberry Pi y Firebase, se configuró un entorno virtual que incluye las librerías listadas en el cuadro 17. Estas dependencias permiten realizar la conexión, autenticación, envío y actualización de datos de manera segura y eficiente.

Cuadro 17: Librerías para integración con Firebase y manejo de datos

Librería / Módulo	**Comando de instalación**	**Función técnica principal**
firebase-admin	pip install firebase-admin	SDK oficial de Firebase para Python. Permite inicializar la conexión con credenciales, autenticar el dispositivo y escribir/leer en la base de datos.
requests	pip install requests	Realiza peticiones HTTP/HTTPS a servicios web. Se usa como alternativa o complemento al SDK para enviar datos en formato JSON.
json (nativa)	–	Convierte estructuras de datos de Python a JSON para ser compatibles con Firebase Realtime Database.
datetime (nativa)	–	Añade una marca temporal (timestamp) a cada registro enviado, permitiendo un orden cronológico de los datos.
threading	pip install threading	Permite que el envío de datos se ejecute en segundo plano, evitando interferencias con el procesamiento principal del modelo.

La integración se realiza utilizando un archivo de credenciales privadas (credenciales.json), generado desde la consola de Firebase en el apartado "Cuentas de servicio", que otorga permisos de lectura y escritura a la Raspberry Pi.

Cada vez que el sistema identifíca un camrón, se crea un registro de información siguiendo un formato previamente establecido.

Estos datos son enviados a la referencia /clasificaciones dentro de Firebase, donde se almacenan de manera ordenada y pueden ser accedidos por la interfaz web. Este formato ligero y estandarizado facilita la compatibilidad con otros servicios de análisis o visualización.

La información almacenada en Firebase se visualiza a través de una interfaz HMI basada en tecnologías web, la cual se compone de los siguientes archivos:

index.html: Este archivo define la estructura visual de la página y la conexión inicial con Firebase:

main.js: Este script realiza la conexión a la base de datos y actualiza la interfaz en tiempo real conforme llegan nuevos registros:
sectionResultados

En la presente sección se presentarán los resultados obtenidos durante el desarrollo e implementación del sistema automatizado de clasificación de camarones por tamaño, utilizando visión artificial y el control de los servomotores para la clasificación mediante estos.

Los resultados se detallarán según las partes principales del proyecto, las cuales son:

Construcción e implementación del equipo.

Desempeño y validación del modelo de red neuronal.

Validación del sistema en tiempo real mediante pruebas experimentales.

2.5. Resultados de Hardware

En las siguientes seciones se tiene todo el desarrollo desde los materiales que se utilizaron para construir la máquina para clasificar, el envío de datos a la nube y como reacciona el modelo de detección en con los camarones ya realizando las pruebas experimentales respectivas.

2.5.1. Estructura de la maquina clasificadora

Para la construcción de la máquina clasificadora se utilizarón materiales con rigidez estructural, resistencia a la corrosión y fácil limpieza.

El sistema completo esta conformado por una Raspberry Pi 4, sensores infrarrojos E18-D80NK, modulo TXS0108E, servomotores, una cinta de transporte y una cámara web.

Los principales materiales y componentes utilizados son:

- Tubo de acero rectangular 20 x 40mm.
- Base construida en plywood 280 x 960 x 4 mm liviano y facil de sujetar.

- Banda transportadora tipo cuero sintentico gris 2300 x 300 mm.
- Pintura gris esmalte
- Rodillos de PVC 300 mm de 2 pulgadas
- Motor reducotr 12v 70 RPM
- Tornilos y turcas M3 y M5
- Cableado para los componentes.
- Cajas plasticas para alojamiento de conexiones
- Piezas y adaptadores obtenidos por impresión 3D
- Tubo de acero rectangular con dimensiones de 20 por 40 mm, que es ligero y se puede soldar o ensamblar fácilmente con tornillos.
- Base hecha de plywood con medidas de 280 por 960 por 4 mm, que resulta económica y sencilla de armar.
- Cinta transportadora fabricada en cuero sintético de color gris, con dimensiones de 2300 por 300 mm.
- Rulinas de tipo 608.

Figura 31: Implementación del sistema de clasificación de camarón automatizada con inteligencia artificial.

En la Figura 31 se puede apreciar la máquina clasificadora de camarón ya realizada e implementada, utilizando los materiales mencionados anteriormente.

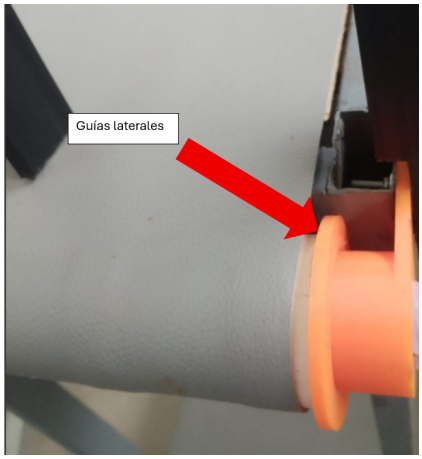


Figura 32: Implementación de guías laterales para evitar el deslizamiento lateral de la banda transportadora

Se presentó un inconveniente al poner en funcionamiento la banda transportadora, ya que esta se desplazaba lateralmente de manera constante. Este problema se solucionó adaptando unas guías laterales en las chumaceras, incluyendo la que sujeta el motor. Como resultado, se logró corregir el desplazamiento lateral, tal como se aprecia en la Figura 32.

2.5.2. Conexiones electricas

Figura 33, durante las pruebas se pudo evidenciar que la fuente de alimentación externa para los sensores, actuadores y módulos TXS0108E garantiza un funcionamiento estable del sistema, lo que evitó problemas de reinicio o pérdidas de comunicación en la Raspberry Pi. Cabe destacar que únicamente se presentan afectaciones debido a problemas de la red a la cual se conecta la Raspberry Pi, Figura 33.

Del mismo modo, la batería de 12 V conectada al motor de la banda transportadora permite regular la velocidad de desplazamiento de los productos. Esta regulación se logra gracias al transistor LM412 y al potenciómetro, obteniendo así los resultados esperados.

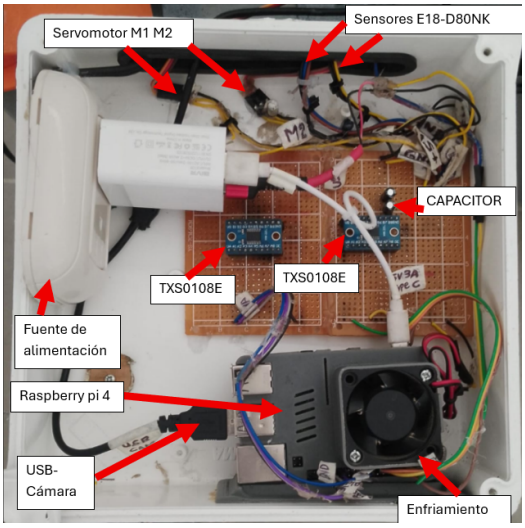


Figura 33: Circuitería interna: conexiones de servomotores y sensores mediante los módulos TXS0108E hacia la Raspberry Pi 4.

2.6. Resultados de Software

2.6.1. Interfaz HMI

Para facilitar la supervisión del sistema en funcionamiento, se elaboró una interfaz que permite visualizar en tiempo real las variables involucradas en el proceso, tales como los servomotores, los sensores infrarrojos, la cámara de visión artificial y el modelo de red neuronal utilizado. A través de la pantalla del computador, se establece la conexión mediante VNC dentro de la misma red, permitiendo conectarse a la Raspberry Pi y observar y controlar en tiempo real lo que ocurre.

Además, se realizó la programación en Geany, la cual refleja en la HMI el estado de los sensores, la acción de los servomotores y la respuesta o clase detectada por el sistema, permitiendo que este efectúe el control de los servomotores.

La interfaz desarrollada ofrece una verificación y validación del desempeño del sistema, permite registrar pruebas experimentales, analizar el comportamiento del modelo con las diferentes clases ya entrenadas y realizar los ajustes correspondientes en el software utilizado. La interfaz se puede apreciar en la Figura 34, como se mencionó previamente, mostrando su funcionamiento.

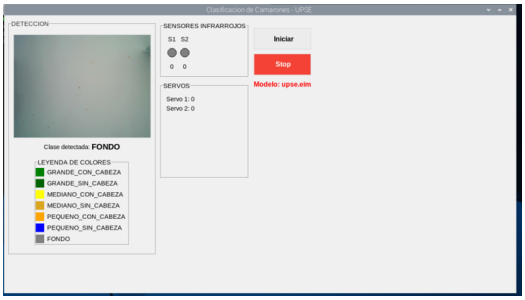


Figura 34: Interfaz HMI desarrollada en la Raspberry Pi para el monitoreo y control del sistema.

2.6.2. Resultados del modelo de visión artificial

El modelo utilizado para la red neuronal fue desarrollado en la plataforma Edge Impulse, empleando la arquitectura MobileNetV2, la cual es la más adecuada y precisa debido a su bajo consumo de recursos y eficiencia en dispositivos embebidos, siendo ideal para la Raspberry Pi 4.

Para realizar el entrenamiento y la validación de la red, se creó un dataset propio que consta de 5,600 imágenes, clasificadas en siete clases, las cuales son:

- `Camarón pequeño con cabeza.`
- `Camarón pequeño sin cabeza.`
- `Camarón mediano con cabeza.`
- `Camarón mediano sin cabeza.`
- `Camarón grande con cabeza.`
- `Camarón grande sin cabeza`
- `Fondo`

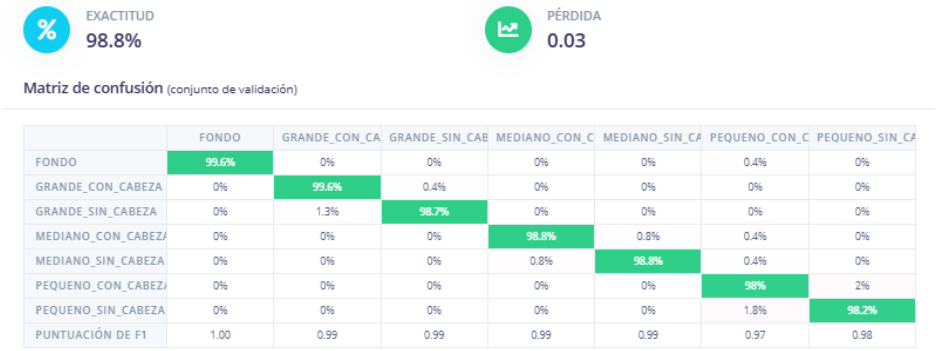
Cada imagen fue capturada con la cámara conectada a la Raspberry Pi 4 mediante la interfaz de Edge Impulse. Las fotografías fueron tomadas desde diferentes ángulos para mejorar la robustez del modelo de la red neuronal.

Para el entrenamiento, se utilizaron las siguientes configuraciones:

- 20 épocas.
- Capa de entrada (49.152 características).
- Tasa de aprendizaje del 0.0005.
- Tamaño de lote de 32.
- 20 % de los datos destinados para el conjunto de validación de la red.

La estructura de la red neuronal MobileNetv2 utilizada se compone con:

- Capa de entrada de 96x96 píxeles.
- Tres canales de colores (RGB).
- Factor de ancho(α)0,25.
- Capa densa de 64 neuronas con tasa de dropout del 50 % para evitar sobreajustes.
- Capa de salida de 7 neuronas con activación Softmax.



	FONDO	GRANDE_CON_CA	GRANDE_SIN_CAB	MEDIANO_CON_C	MEDIANO_SIN_CA	PEQUENO_CON_C	PEQUENO_SIN_CA
FONDO	99.6%	0%	0%	0%	0%	0.4%	0%
GRANDE_CON_CABEZA	0%	99.6%	0.4%	0%	0%	0%	0%
GRANDE_SIN_CABEZA	0%	1.3%	98.7%	0%	0%	0%	0%
MEDIANO_CON_CABEZA	0%	0%	0%	98.8%	0.8%	0.4%	0%
MEDIANO_SIN_CABEZA	0%	0%	0%	0.8%	98.8%	0.4%	0%
PEQUENO_CON_CABEZA	0%	0%	0%	0%	0%	98%	2%
PEQUENO_SIN_CABEZA	0%	0%	0%	0%	0%	1.8%	98.2%
PUNTUACIÓN DE F1	1.00	0.99	0.99	0.99	0.99	0.97	0.98

Figura 35: Matriz de confusión de entrenamiento del modelo MobileNetv1.

En la Figura 35 se puede observar la matriz de confusión del entrenamiento del modelo de red neuronal, en la cual se aprecia lo siguiente:

- Exactitud general: 98.8 % y una pérdida de 0.03, lo que indica buenos resultados.
- La clase “fondo” se distingue 99.6 % de las demás clases del modelo,con un 0.4 % de similitud con “pequeno sin cabeza”.
- La clase “grande con cabeza” se distingue en un 99.6 % de las demás clases, con un 0.4 % de similitud con “grande sin cabeza”.
- La clase “grande sin cabeza” es 98.7 % distinta de las demás clases, con un 1.3 % de similitud con “grande con cabeza”.
- La clase “mediano con cabeza” se distingue en un 98.8 % de las demás clases, presentando un 0.8 % de similitud tanto con “mediano sin cabeza” y 0.4 % con “pequeno con cabeza”.
- La clase “mediano sin cabeza” es 98.8 % distinta de las demás clases, con un 0.8 % de similitud con “mediano sin cabeza” y 0.4 % con "pequeno con cabeza".
- La clase “pequeño con cabeza” es 98 % distinta de las demás clases, con un 2 % de similitud con “pequeño sin cabeza”.
- La clase “pequeño sin cabeza” es 98.2 % distinta de las demás clases, con un 1.8 % de similitud con “pequeno con cabeza”.

Con base en los resultados obtenidos, se puede afirmar que el modelo es altamente apto para integrar la red neuronal en el sistema clasificación de camarón.

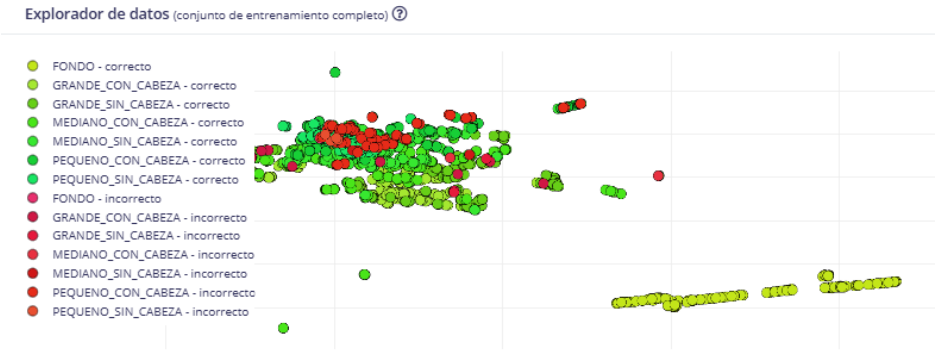
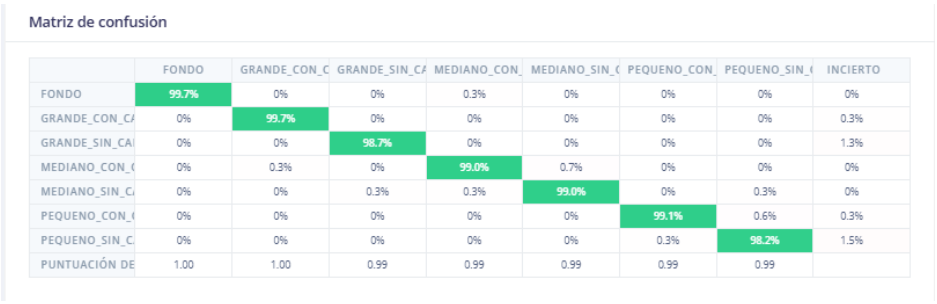


Figura 36: Conjunto de datos para entrenamiento.

En la Figura 36 se pueden observar las clases incorrectas, señaladas en color rojo, en todo el conjunto de datos. Las clases en color verde presentan un porcentaje muy bajo de similitud con otras clases, pero son adecuadas para el modelo. Por su parte, las clases en color amarillo son completamente distintas de las demás, como es el caso de la clase "fondo".



	FONDO	GRANDE_CON_C	GRANDE_SIN_CA	MEDIANO_CON_	MEDIANO_SIN_(	PEQUENO_CON_	PEQUENO_SIN_(	INCIERTO
FONDO	99.7%	0%	0%	0.3%	0%	0%	0%	0%
GRANDE_CON_CA	0%	99.7%	0%	0%	0%	0%	0%	0.3%
GRANDE_SIN_CA	0%	0%	98.7%	0%	0%	0%	0%	1.3%
MEDIANO_CON_(	0%	0.3%	0%	99.0%	0.7%	0%	0%	0%
MEDIANO_SIN_CA	0%	0%	0.3%	0.3%	99.0%	0%	0.3%	0%
PEQUENO_CON_(	0%	0%	0%	0%	0%	99.1%	0.6%	0.3%
PEQUENO_SIN_C	0%	0%	0%	0%	0%	0.3%	98.2%	1.5%
PUNTUACIÓN DE	1.00	1.00	0.99	0.99	0.99	0.99	0.99	

Figura 37: Matriz de validación.

En la Figura 37 se muestra la matriz de validación, la cual permite evaluar los datos ya entrenados, obteniéndose los siguientes resultados:

- La clase "fondo" se diferencia en un 99.7 % de las demás clases, con un 0.3 % de similitud con la clase "mediano con cabeza".
- La clase "grande con cabeza" se diferencia en un 99.7 % de las demás clases, con un 0.3 % para ser considerada incierta.
- La clase "grande sin cabeza" se diferencia en un 98.7 % de las demás clases, con un 1.3 % para ser considerada incierta.

- La clase “mediano con cabeza” se diferencia en un 99.0 % de las demás clases, con un 0.3 % de similitud con “grande sin cabeza” y 0.7 % para ser considerada "mediano sin cabeza".
- La clase “mediano sin cabeza” se distingue en un 99 % de las demás clases del modelo, con 0.3 % de similitud para cada clase que son: "grande sin cabeza", "mediano con cabezaz "pequeno sin cabeza"
- La clase “pequeño con cabeza” se diferencia en un 99.1 % de las demás clases, con un 0.6 % de similitud con “pequeño sin cabeza” y 0.3 % para ser ïncierto".
- La clase “pequeño sin cabeza” se diferencia en un 98.2 % de las demás clases, con un 1.5 % para ser considerada incierta.

Además, se empleó la variante cuantificada (FLOAT32), lo que facilita la disminución del tamaño del archivo final sin comprometer el rendimiento del modelo de red neuronal creado.

Una vez finalizado el modelo, este se exporta en formato .eim para ser utilizado en la Raspberry Pi y ejecutado en tiempo real desde la placa. Se pudo apreciar que el modelo es muy eficiente para la detección de los diferentes tamaños de camarones considerados en la red se obtiene lo siguiente:

- La clase “fondo” se distingue en un 100 % de las demás clases del modelo.
- La clase “grande con cabeza” se diferencia en un 100 % de las demás clases.
- La clase “grande sin cabeza” se diferencia en un 99.0 % de las demás clases.
- La clase “mediano con cabeza” se diferencia en un 99.0 % de las demás clases.
- La clase “mediano sin cabeza” se diferencia en un 99.0 % de las demás clases.
- La clase “pequeño con cabeza” se diferencia en un 99.0 % de las demás clases.
- La clase “pequeño sin cabeza” se diferencia en un 99.0 % de las demás clases.

2.6.3. Pruebas experimentales

2.6.3.1. Pruebas de los camarones realizadas con el modelo VA.

Para verificar el funcionamiento completo del sistema, se realizaron ensayos prácticos utilizando camarones de diferentes dimensiones y estados, incluyendo muestras con y sin cabeza.

Las pruebas consistieron en colocar un camarón sobre la banda transportadora, la cual es monitoreada por la cámara. El modelo previamente entrenado detecta si el camarón es pequeño, mediano o grande, con o sin cabeza, y ejecuta la acción correspondiente en los servomotores, según la clase detectada.

A continuación, se presentan las pruebas realizadas:

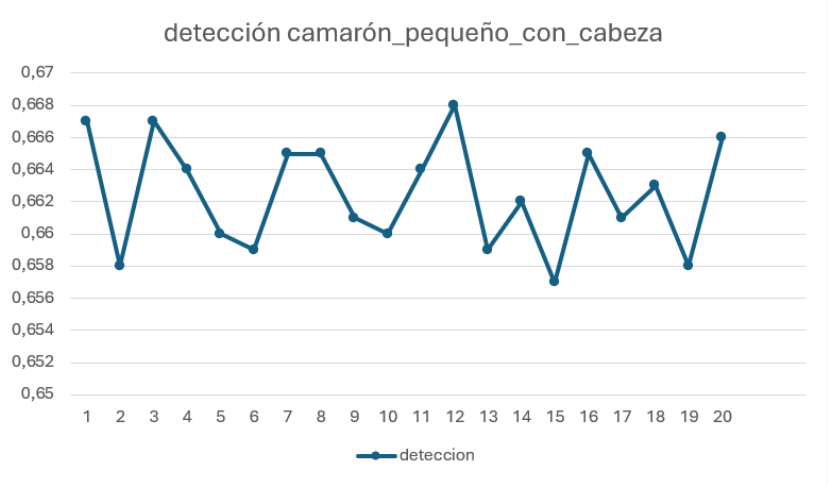


Figura 38: Detección camarón pequeño con cabeza.

En la figura 38 se puede apreciar en la grafica las 20 pruebas realizadas con los camorones de la clase pequeños con cabeza que el modelo esta detectando perfectamente con un valor promedio de detección de 66.4 %.

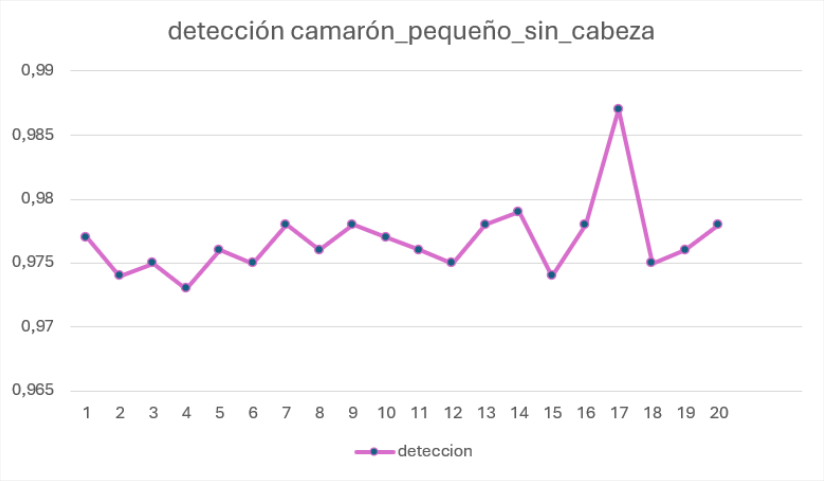


Figura 39: Detección camarón pequeño sin cabeza.

Figura 39, se puede apreciar en la grafica las 20 pruebas realizadas con los camorones de la clase pequeños sin cabeza que el modelo esta detectando perfectamente con un valor promedio de detección de 97.6 %.

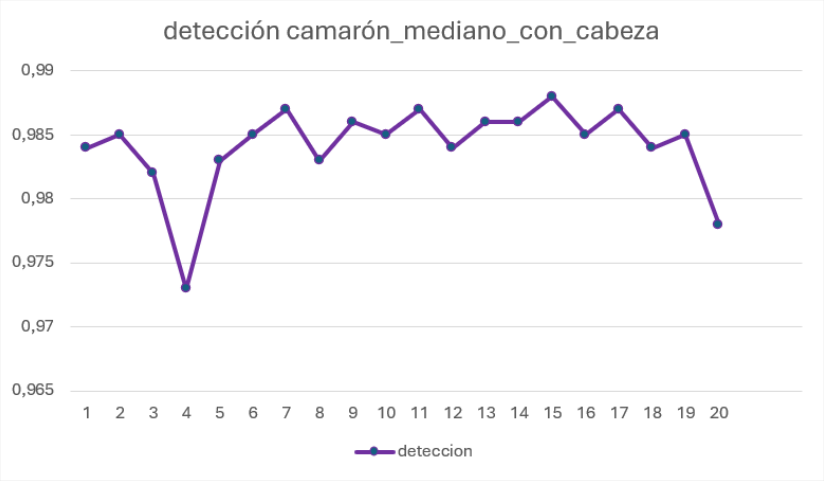


Figura 40: Detección camarón mediano con cabeza.

Figura 40, se puede apreciar en la grafica las 20 pruebas realizadas con los camarones de la clase mediano con cabeza que el modelo esta detectando perfectamente con un valor promedio de detección de 98.5 %.

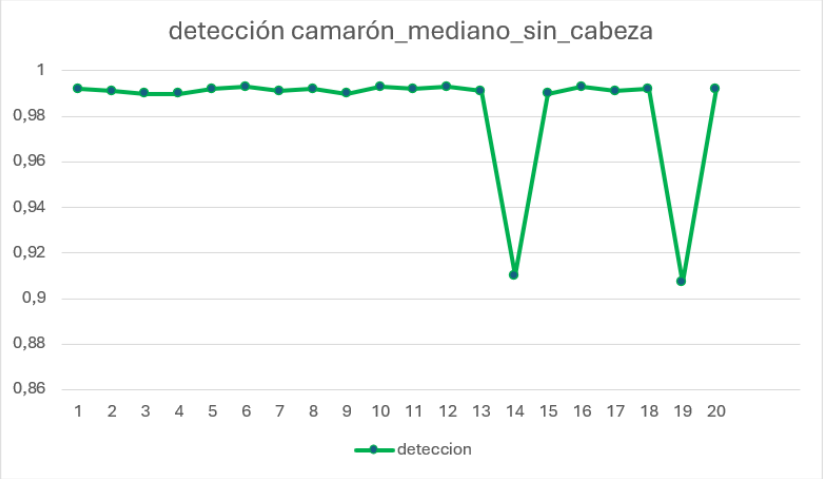


Figura 41: Detección camarón mediano con cabeza.

Figura 41, se puede apreciar en la grafica las 20 pruebas realizadas con los camarones de la clase mediano sin cabeza que el modelo esta detectando perfectamente con un valor promedio de detección de 98.7 %.

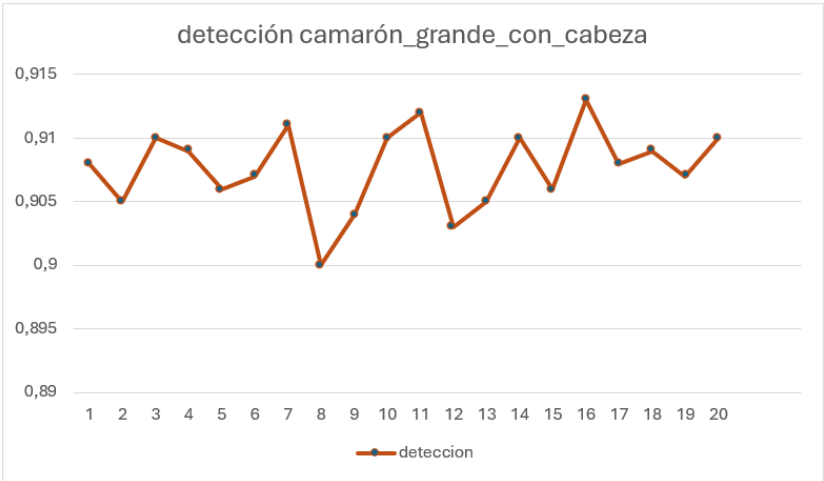


Figura 42: Detección camarón grande con cabeza.

Figura 42, se puede apreciar en la grafica las 20 pruebas realizadas con los camarones de la clase grande con cabeza que el modelo esta detectando perfectamente con un valor promedio de detección de 90.7 %.

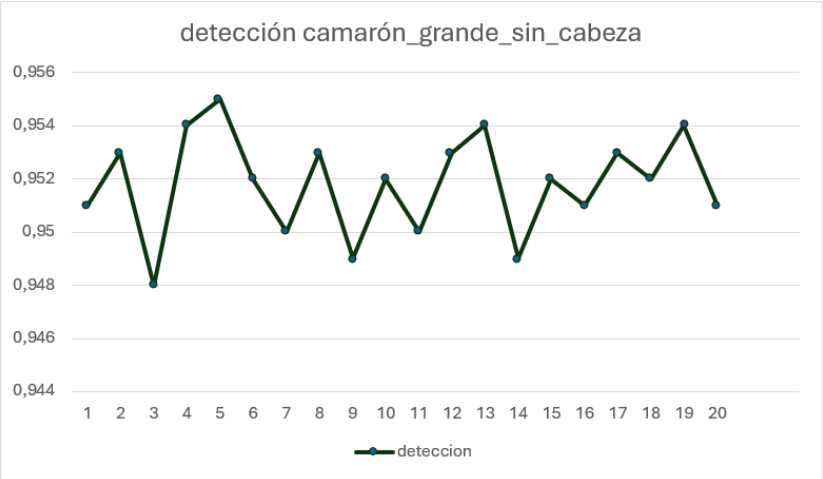


Figura 43: Detección camarón grande sin cabeza.

Figura 43, se puede apreciar en la grafica las 20 pruebas realizadas con los camarones de la clase grande sin cabeza que el modelo esta detectando perfectamente con un valor promedio de detección de 95.2 %.

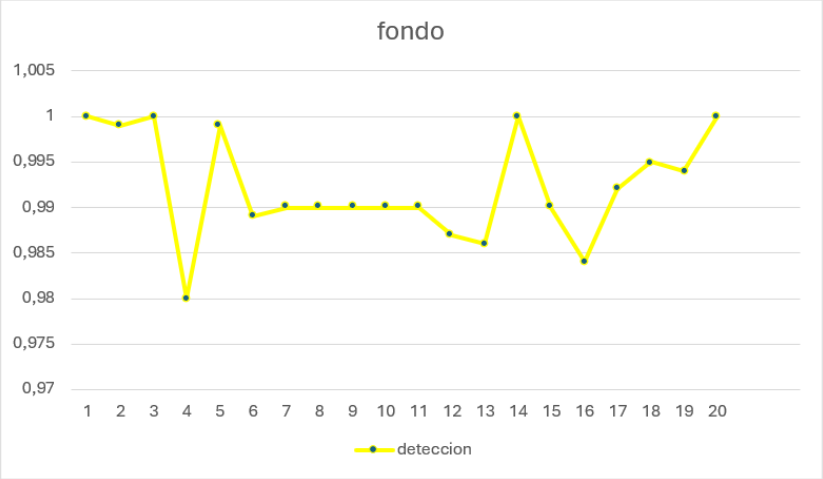


Figura 44: Detección fondo.

Figura 44, se puede apreciar en la grafica las 20 pruebas realizadas con los camarones de la clase fondo que el modelo esta detectando perfectamente con un valor promedio de detección de 99.9 %.

2.6.3.2. Pruebas de los servomotores con al detección de los camarones.

Figura 45: Funcionamiento de servomotores con camarones pequeños con cabeza.

Figura 45, se puede apreciar en la grafica los servomotores se mantiene en el estado deseado servomotor 1 = 0° y servomotor 2 = -55°, clase detectada camarón pequeño con cabeza.

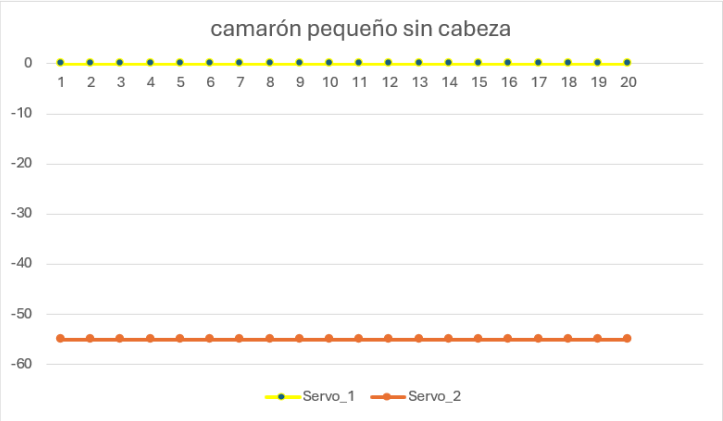


Figura 46: Funcionamiento de servomotores con camarones pequeños sin cabeza.

Figura 46, se puede apreciar en la grafica los servomotores se mantiene en el estado deseado servomotor 1 = 0° y servomotor 2 = -55°, clase detectada camarón pequeño sin cabeza.

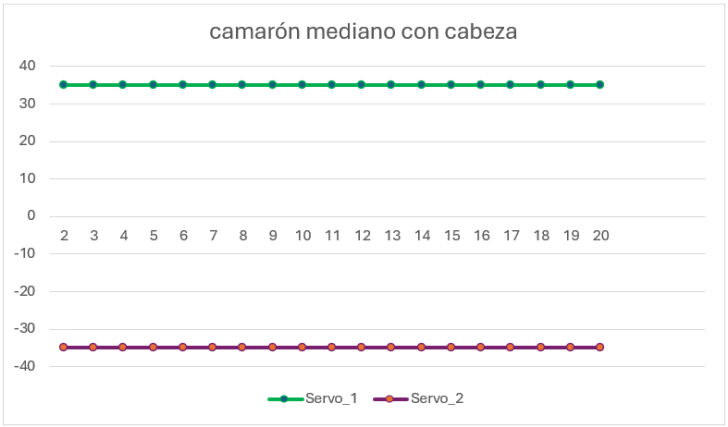


Figura 47: Funcionamiento de servomotores con camarones mediano con cabeza.

Figura 47, se puede apreciar en la grafica los servomotores se mantiene en el estado deseado servomotor 1 = -35° y servomotor 2 = -35°, clase detectada camarón mediano con cabeza.

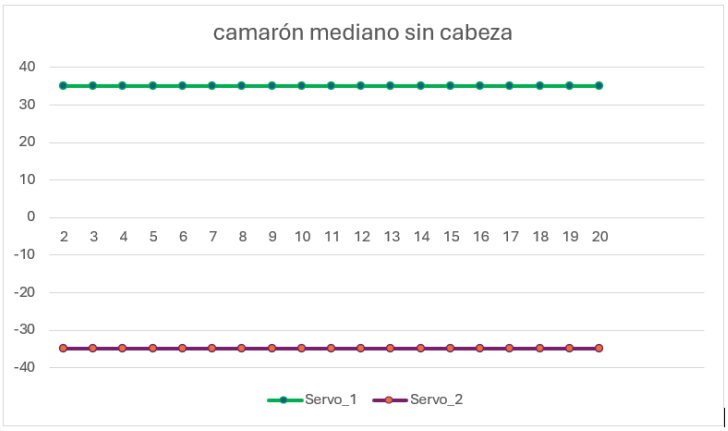


Figura 48: Funcionamiento de servomotores con camarones mediano sin cabeza.

Figura 48, se puede apreciar en la grafica los servomotores se mantiene en el estado deseado servomotor 1 = -35° y servomotor 2 = -35°, clase detectada camarón mediano sin cabeza.

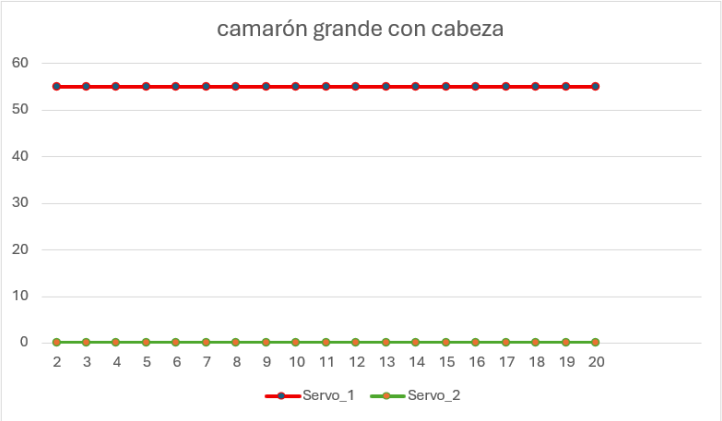


Figura 49: Funcionamiento de servomotores con camarones grande con cabeza.

Figura 49, se puede apreciar en la grafica los servomotores se mantiene en el estado deseado servomotor 1 = -55° y servomotor 2 = 0°, clase detectada camarón grande con cabeza.

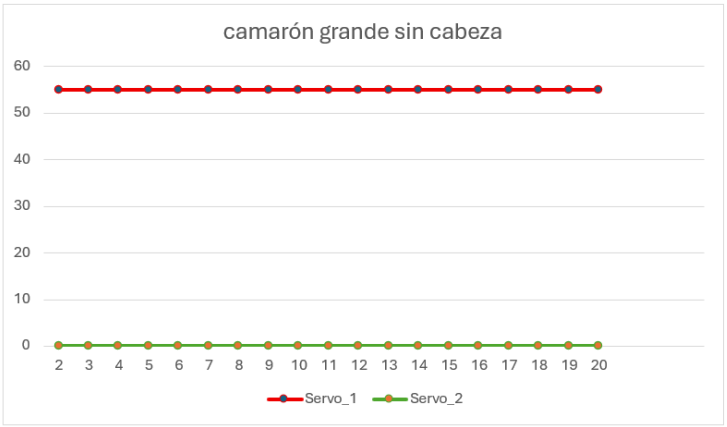


Figura 50: Funcionamiento de servomotores con camarones grande sin cabeza.

Figura 50, se puede apreciar en la grafica los servomotores se mantiene en el estado deseado servomotor 1 = -55° y servomotor 2 = 0°, clase detectada camarón grande sin cabeza.

Cuadro 18: Resultados de Detección

Clase detectada	**Total de pruebas**	**Aciertos**	**Precisión (%)**	**No Precisión (%)**
Pequeño con cabeza	20	16	80 %	20 %
Pequeño sin cabeza	20	18	90 %	10 %
Mediano con cabeza	20	19	95 %	5 %
Mediano sin cabeza	20	17	85 %	15 %
Grande con cabeza	20	19	95 %	5 %
Grande sin cabeza	20	18	90 %	10 %
Fondo	20	20	100 %	0 %
Promedio general	**140**	**127**	**90.71 %**	**9.29 %**

Tabla 18 ,esta presenta los resultados de detección de diferentes clases de camarones del modelo realizado. Se evaluaron siete categorías de camarones (pequeños, medianos y grandes, con y sin cabeza, además del fondo), con 20 pruebas por clase, para un total de 140 pruebas.

Los resultados muestran una precisión promedio general del 90,71 %, lo que indica una alta tasa de éxito en la detección, mientras que la tasa de error o falta de precisión fue del 9,29 %. El mejor rendimiento fue el de fondo (100 %) de precisión), y la clase más baja fue "pequeño con cabeza"(80 %).

2.6.4. Resultados de envio de datos a la nube

Figura 51, se logró cumplir satisfactoriamente el objetivo específico de enviar los datos en tiempo real a la nube, implementando un dashboard web diseñado con la plataforma Firebase. Esta herramienta permitió visualizar de forma virtual y dinámica todo el proceso de clasificación del camarón, mostrando la información actualizada en cada instante. Además, el sistema se puede supervisar de manera remota desde cualquier lugar del mundo, siempre que tanto la placa Raspberry Pi como el dispositivo del usuario cuenten con conexión a Internet. De esta forma, se garantiza un monitoreo continuo, práctico y eficiente mediante una simple dirección web personalizada.

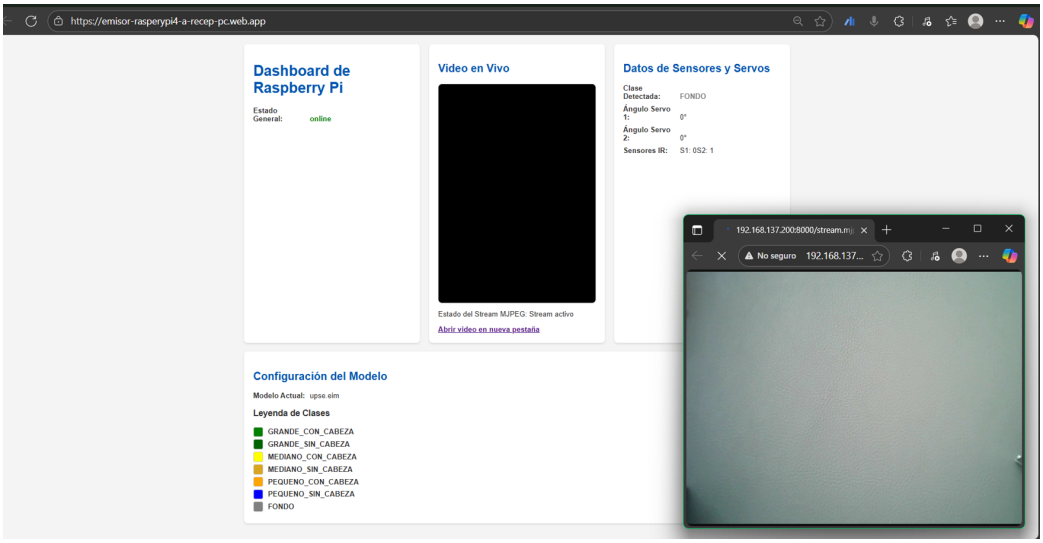


Figura 51: Dashboard Web realizado con firebase para en envió de datos en tiempo real a la plataforma.

3. Conclusiones y Recomendaciones

3.1. Conclusiones

Con el presente trabajo se puede concluir que se realizó lo que verdaderamente se requería al construir la clasificadora de camarón para realizar todo el sistema necesario que se emplea, esta incluye la banda transportadora que pasa a los camarones hacia los contenedores respectivos, también se le adapto los servomotores que se encargan de separar los camarones uno de los otros, y la web cam que permite al modelo ya entrenado clasificar los camarones que está captando en tiempo real, y finalmente se aisló todo la parte eléctrica en una caja especial para el sistema eléctrico.

El modelo MobileNetv2 es muy bueno para la detección se puede comprobar y obtener muy buenos resultados con la clasificación de los camarones a diferencia de los otro modelos que se estaban utilizando anteriormente tenía muchas fallas estas incluyen ralentizamiento de los datos obtenidos, modelo muy pesado para la placa Raspberry lo que hacía que se calentara más de lo normal la placa Raspberry pero con la utilización del Modelo MobileNetV1 de Edge Impulse se tiene los resultados que se esperaba.

La clasificación por los servomotore es muy aceptable ya que hay cierto porcentaje de error al mover el servo 1 este es un poco brusco, pero siempre cumple con los grados necesarios para mandar los camarones a su respectiva sección designada hacia el contenedor.

Para finalizar, se puede afirmar que los datos hacia la nube de Firebase se envía en tiempo real, lo que hace que esta sea una ventaja para el monitoreo en tiempo real de lo que ocurre en tiempo real, además esta conecta con una interfaz gráfica que visualiza todo el proceso la clase detectada, servomotores y los sensores infrarrojos y la deteccion o el video en vivo se necesita d ela ip de la Rasperry para verlo en tiempo local.

3.2. Recomendaciones

A partir de los resultados obtenidos en la construcción y validación de la máquina clasificadora de camarón, se recomienda continuar con la optimización de los algoritmos de visión artificial, especialmente en lo referente al tratamiento de imágenes bajo diferentes condiciones de iluminación, contraste y movimiento, con el fin de incrementar la precisión del reconocimiento y la clasificación.

Se sugiere investigar arquitecturas de redes neuronales más eficientes, diseñadas para dispositivos que consumen menos energía, o bien combinar modelos de aprendizaje profundo con técnicas de transferencia de conocimiento para potenciar la habilidad del sistema en la generalización. En lo que respecta a la infraestructura de hardware, se recomienda la incorporación de sensores extras que permitan la medición de variables como el peso, la temperatura o la humedad, lo que enriquecería la información visual y facilitaría un análisis más completo del producto.

En el apartado de control, es posible utilizar PLC o microcontroladores dedicados para mejorar la compatiblidad entre los servomotores y la unidad principal del sistema. En cuanto a la gestión y supervisión en la nube, se recomienda fortalecer la seguridad de los datos mediante protocolos de cifrado y autenticación, además de crear interfaces de visualización más intuitivas que facilitan el monitoreo remoto y apoyen la toma de decisiones en tiempo real.

Es recomendable llevar un registro detallado de cada actualización del sistema, aplicar un plan de mantenimiento preventivo y considerar el uso de energías renovables para asegurar un funcionamiento sostenible. A futuro, este tipo de maquinas puede ajustarse a distintas clases versátil para procesos de clasificación automatizada, aportando valor a la cadena productiva y favoreciendo la digitalización en la industria pesquera.

4. Bibliografía

Referencias

[1] Electronica Lam, "Cable hdmi plug to hdmi," 2025. Accedido: 29 de agosto de 2025.

[2] Jukino.

[3] G. G. C., "Red neuronal inteligencia artificial," tech. rep., Naps Tecnología y Educación, 2020. Publicado el 22 de septiembre de 2020, consultado en https://naps.com.mx/blog/introduccion-a-la-inteligencia-artificial/3-2/.

[4] H. N. Dang, "Mobilenet architecture." https://danghoangnhan.github.io/MobileNetArchitecture/, Apr. 2023. Accessed: 21 de octubre de 2025.

[5] J. R. Osio, W. J. Aróztegui, and J. Rapallini, *Sistemas digitales basados en microcontroladores: Descripción, funcionalidades y aplicaciones de los microcontroladores basadas en el CPU HCS08*. 2020.

[6] Rubén, "Cómo usar un motor de corriente continua dc con crumble," 6 2024.

[7] TecnoSalva, "Control de un servomotor con arduino." https://www.tecnosalva.com/control-de-un-servomotor-con-arduino/, 2020. Fecha de acceso: 9 de septiembre de 2025.

[8] K. México, "Sensores fotoeléctricos: Principio y tipos principales," 2025. Accedido: 09 septiembre 2025.

[9] RoboticsEC, "Raspberry pi 4 modelo b 8gb." https://roboticsec.com/producto/raspberry-pi-4-modelo-b-8gb/, 2025. Accedido el 17 de noviembre de 2025.

[10] TowerPro, "Mg946r digital metal servo 13kg," 2025. Accessed: Nov. 17, 2025.

[11] SanDisk, "Sandisk extreme® tarjeta microsdxc™ uhs-i - 32gb," 2025. Accessed: Nov. 17, 2025.

[12] Sigma Electrónica, "Sensor de proximidad infrarrojo e18-d80nk," 2025. Accessed: Nov. 17, 2025.

[13] Lamtech, "Cámara web usb de alta definición lamtech 720p," 2025. Accessed: Nov. 17, 2025.

[14] Texas Instruments, *TXS0108E: 8-Bit Bidirectional Voltage-Level Translator for Open-Drain and Push-Pull Applications*, Nov 2021. Datasheet (Rev. L).

[15] TESLA ELECTRONIC, "Condensador Electrolítico 1000uf, 25V," 2025.

[16] Raspberry Pi Foundation, "Raspberry pi 4 model b specifications," 2023. Accessed: Nov. 17, 2025.

[17] P. González, "Camarón supera al petróleo en exportaciones del primer semestre en ecuador," 2025. Consultado el 8 de septiembre de 2025.

[18] X. I. Toala Peña and D. J. Zambrano Yagual, "Sistema de reconocimiento y clasificación de camarón utilizando visión artificial," 2023.

[19] D. F. Sánchez Huilcapi and A. N. Manrique Torres, "Diseño y simulación de un proceso de control de calidad en camaroneras usando tia portal y plcsim," 2025.

[20] D. Angulo Acosta, V. A. Arce Domínguez, *et al.*, *Diseño de proyecto automatización de una máquina clasificadora de camarón para empresa manufacturera*. PhD thesis, ESPOL. FIEC, 2025.

[21] X. I. Toala Peña and D. J. Zambrano Yagual, "Sistema de reconocimiento y clasificación de camarón utilizando visión artificial," B.S. thesis, 2023.

[22] FNB Tech, "Raising shrimp - guía completa para el cultivo de camarones," 2023. Consultado el [Fecha de consulta].

[23] C. A. Commission, "Standard for canned shrimps or prawns (codex stan 37-1991)," tech. rep., Food and Agriculture Organization of the United Nations and World Health Organization, 2013. Amended 2013.

[24] Servicio de Acreditación Ecuatoriano, "Én qué consiste la ISO 22000?." https://www.acreditacion.gob.ec/en-que-consiste-la-iso-22000/, Desconocido.

[25] Instituto Ecuatoriano de Normalización (INEN), "NORMA TÉCNICA ECUATORIANA NTE INEN 456:2013: CAMARONES O LANGOSTINOS CONGELADOS. REQUISITOS. Primera revisión," Norma Técnica NTE INEN 456:2013, INEN, Quito, Ecuador, 2013.

[26] R. E. A. GARCÍA, *DETERMINACIÓN DE RESIDUOS DE CLORANFENICOL EN CAMARÓN DE ACUACULTURA (Litopenaeus vannamei) MEDIANTE TÉCNICA ELISA EXPENDIDOS EN LOS MERCADOS MUNICIPALES DE LA ZONA SURESTE DE LA CIUDAD DE GUAYAQUIL*. PhD thesis, UNIVERSIDAD AGRARIA DEL ECUADOR, 2022.

[27] S. Equihua, "Hablemos sobre Big Data." Blogtecnarios, Sept. 2013. Consultado el 24 de noviembe de 2025.

[28] LibreTexts, "Monitoreo, analítica y control de proyectos." LibreTexts Español, n.d. Proceso continuo que implica recopilar datos, analizarlos y tomar acciones correctivas para asegurar el cumplimiento de los objetivos.

[29] HDMI Licensing Administrator, Inc., "Hdmi technology overview," 2023. Accedido el 8 de septiembre de 2025.

[30] R. T. Fielding, M. Nottingham, and J. Reschke, "Http semantics," RFC 9110, Internet Engineering Task Force (IETF), 2022. Accedido el 8 de septiembre de 2025.

[31] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2 ed., 2022. Accedido el 8 de septiembre de 2025.

[32] Shaip, "Ia para reconocimiento de imágenes: qué es, cómo funciona y ejemplos," julio 2025. Accedido: 19 agosto 2025.

[33] Wikipedia contributors, "Red neuronal artificial — Wikipedia, la enciclopedia libre," 2025. Accedido: 19 agosto 2025.

[34] Wikipedia, "Microprocesador." https://es.wikipedia.org/wiki/Microprocesador, 2024. Accedido el 24 de noviembre de 2025.

[35] Fundación Red de Energía BUN-CA, "Manual técnico de motores eléctricos," tech. rep., Fundación Red de Energía BUN-CA, 2011. Consultado en https://www.bun-ca.org/wp-content/uploads/2019/02/Motores.pdf.

[36] Varios, "Guía básica sobre servomotores: Características y aplicaciones," 2025.

[37] Wikipedia, "Sensor fotoeléctrico," 2024. Accedido: 9 septiembre 2025.

[38] Python Software Foundation, "What is python? executive summary." https://www.python.org/doc/essays/blurb/, 2025. Accedido el 9 de septiembre de 2025.

[39] P. Chougale, V. Yadav, A. Gaikwad, and B. Vidyapeeth, "Firebase-overview and usage," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 3, no. 12, pp. 1178–1183, 2021.

[40] S. Studio, "Edge impulse en reterminal: Ia en el borde," 2025. Accedido: 09 septiembre 2025.

[41] L. A. Barrera, "Exportación del camarón y su impacto a la economía," *Dialnet*, 2024.

[42] F. A. T. Ponce, "Eficiencia productiva en la industria camaronera del ecuador," *Dialnet*, 2025.

[43] F. S. M. Arteaga, "Análisis económico de la producción ecuatoriana de camarón," *Zamorano*, 2009.

[44] A. N. Manrique Torres, "Diseño y simulación de un proceso de control de calidad en la industria camaronera," *Repositorio UPS*, 2025.

[45] X. I. Toala Peña, "Sistema de reconocimiento y clasificación de camarón utilizando redes neuronales," *Repositorio UPS*, 2023.

[46] A. N. Manrique Torres, "Ensacado del camarón: proceso y automatización," *Repositorio UPS*, 2025.

[47] M. R. Intellect, "Máquina de clasificación de camarones automática," 2024.

[48] G. G. Insights, "Mercado de equipos de procesamiento de camarones," 2025.

[49] Primicias, "Camarón ecuatoriano: los desafíos del sector que define la economía," 2025.

[50] G. Seafood, "La industria de cultivo de camarón en ecuador, parte 1," 2018.

5. Anexos

A continuación se muestran algunas imágenes del prototipo construido:

En anexo 1 se puede evidenciar todo el diagrama realizado en proteus para la posterior implmentación que se realizó.

En anexos 2 el modelo esta detectando un camarón grande con cabeza.

En anexos 3 el modelo esta detectando un camarón grande sin cabeza.

En anexos 4 el modelo esta detectando un camarón mediano con cabeza.

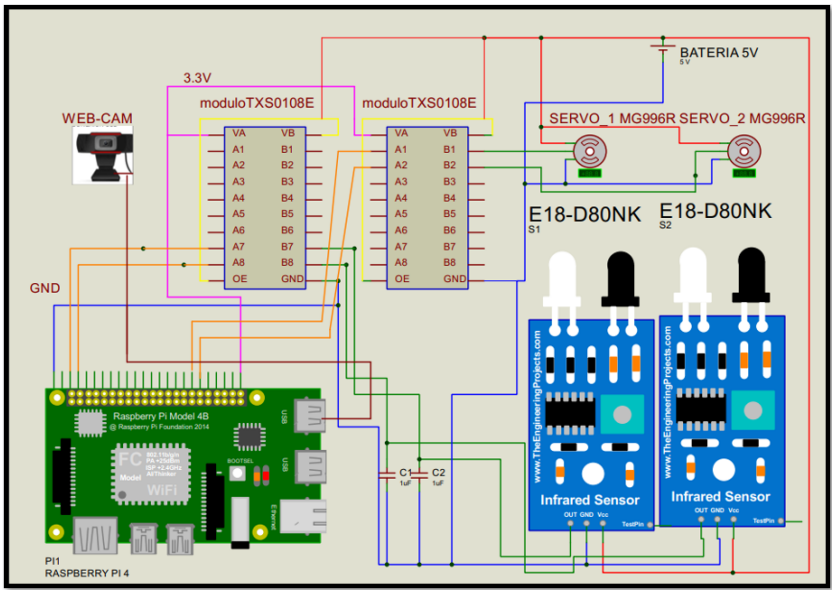
En anexos 5 el modelo esta detectando un camarón mediano sin cabeza.

En anexos 6 el modelo esta detectando un camarón pequeño con cabeza.

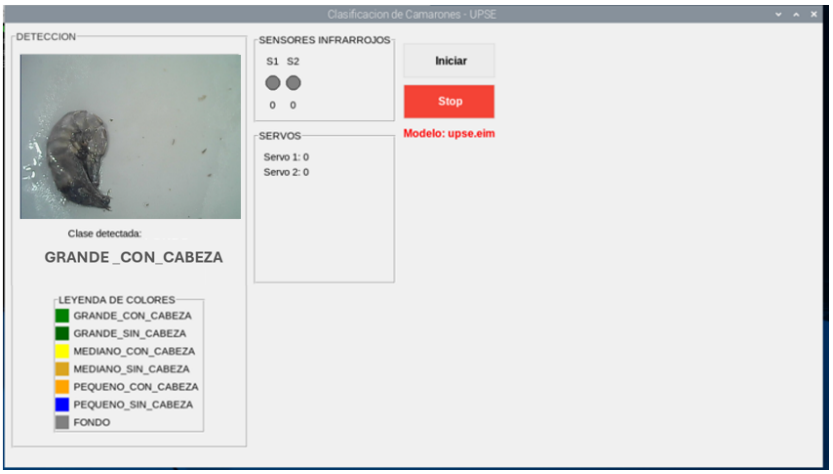
En anexos 7 el modelo esta detectando un camarón pequeño sin cabeza.

En anexos 8 el modelo esta detectando un guante y este lo hace pasar como fondo ya que solo detecta los camarones y si este detecta cualquier otra cosa lo ase pasar como la clase de fondo.

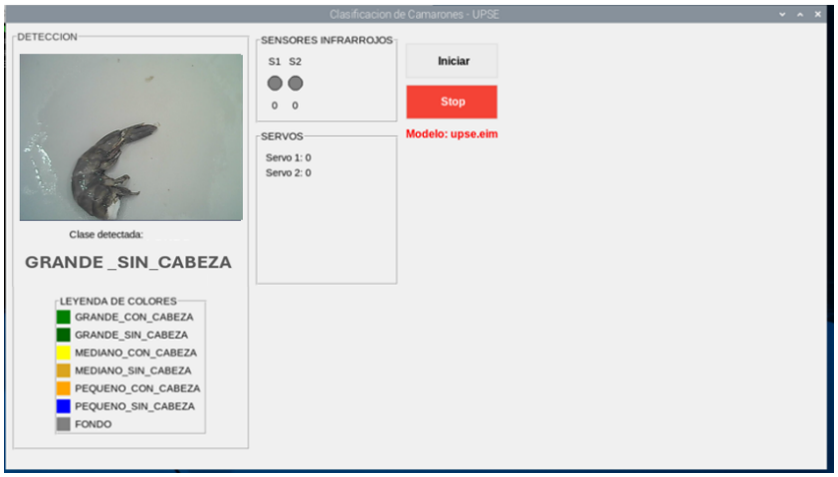
En anexo 9 ya seria la construcción ya finalizada de la maquina realizada para el proyecto.



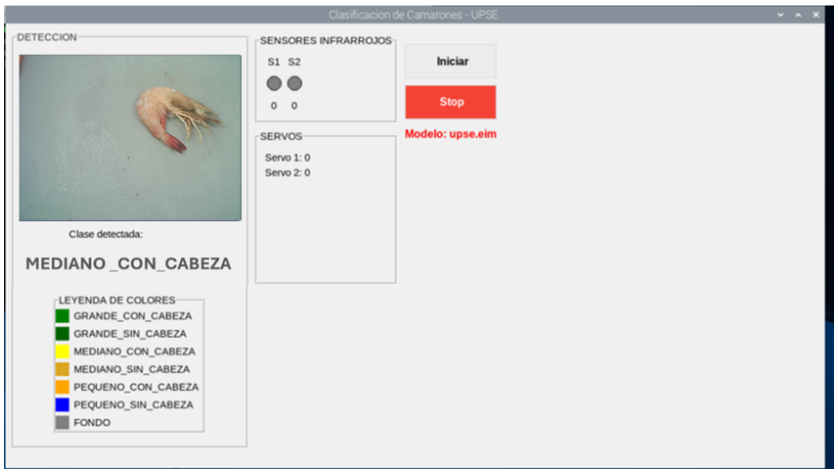
Anexo 1. Diagrama de conexión de componentes y placa Raspberry Pi 4.



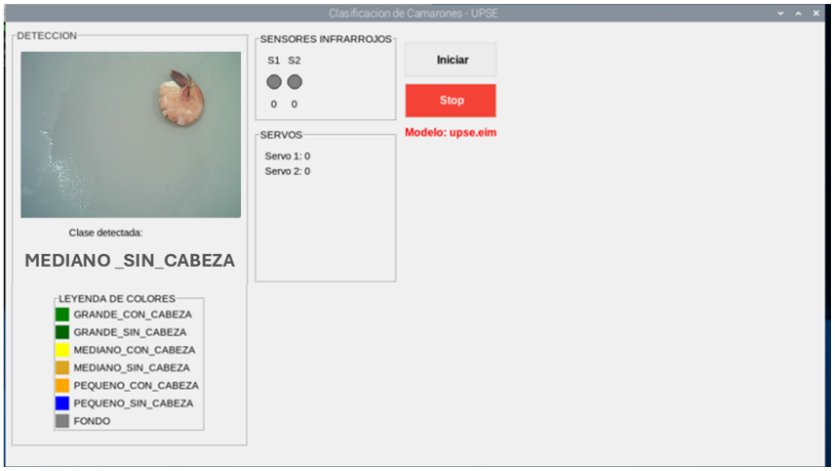
Anexo 2. Detección camarón grande con cabeza.



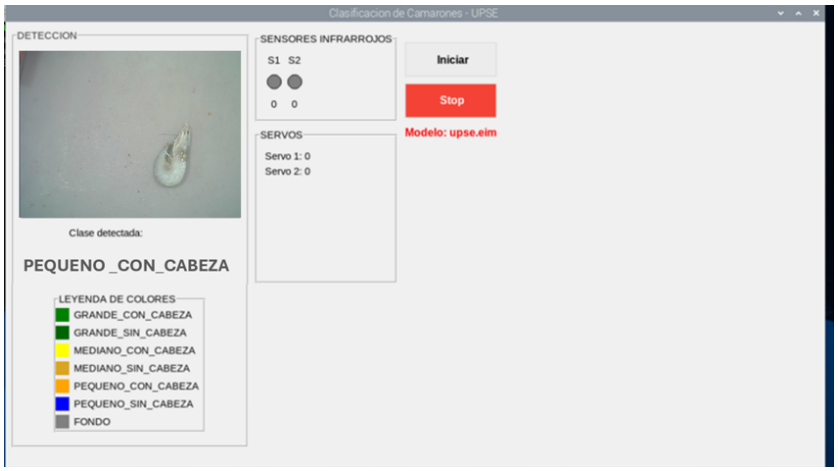
Anexo 3. Detección camarón grande sin cabeza.



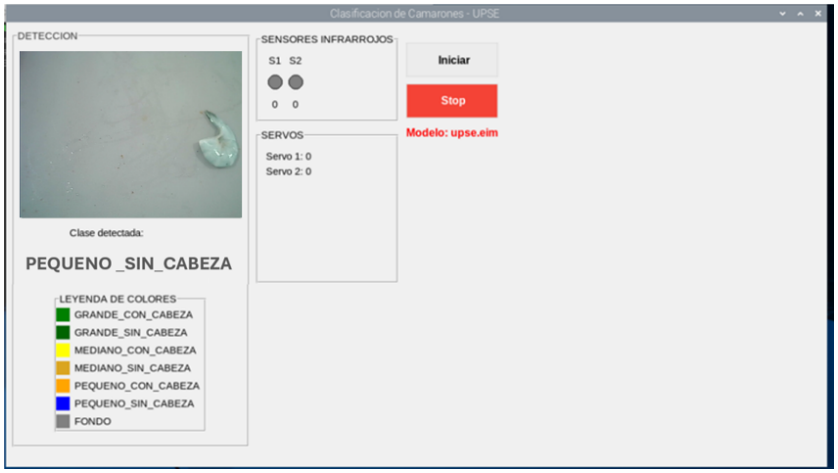
Anexo 4. Detección camarón mediano con cabeza.



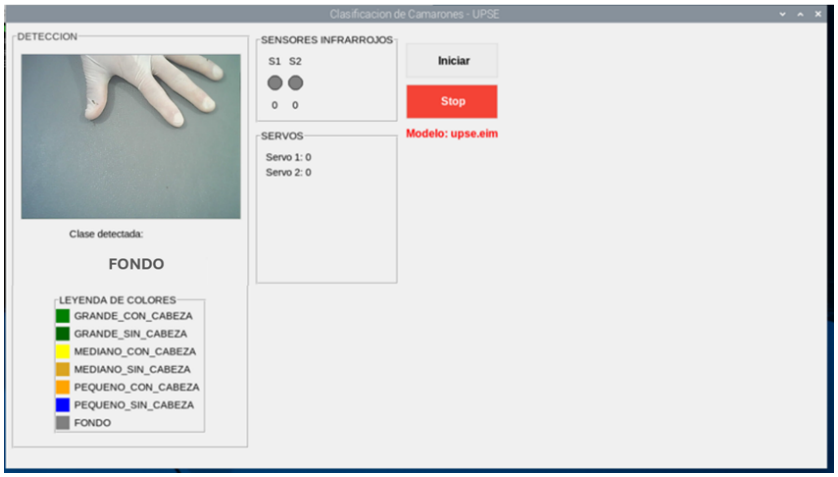
Anexo 5. Detección camarón mediano sin cabeza.



Anexo 6. Detección camarón pequeño con cabeza.



Anexo 7. Detección camarón pequeño sin cabeza.



Anexo 8. Detección Fondo.

Anexo 9. Maquina de clasificación de camarón completa.

Ejemplo de código Python

A continuación se muestra un fragmento de código que importa librerías y ejecuta el programa importante en la Raspberry

Listing 1: Código Python del programa

```
1
2
3  # -*- coding: utf-8 -*-
4  import RPi.GPIO as GPIO
5  import subprocess
6  import time
7  import re
8  import threading
9  import tkinter as tk
10 from tkinter import ttk, messagebox
11 import cv2
12 from PIL import Image, ImageTk
13 import os
14 from http.server import BaseHTTPRequestHandler,
       HTTPServer
15 from socketserver import ThreadingMixIn
16 import socket
17 import firebase_admin
18 from firebase_admin import credentials, db
19
20 # === INICIALIZACION DE FIREBASE ===
21 cred = credentials.Certificate("serviceAccountKey.json")
22 firebase_admin.initialize_app(cred, {
23     'databaseURL': 'https://emisor-rasperypi4-a-recep-pc-
          default-rtdb.firebaseio.com/'
24 })
25 ref = db.reference('/raspberry_data')
26
27 # === Configuracion GPIO ===
28 GPIO.setmode(GPIO.BCM)
29 GPIO.setwarnings(False)
30
31 servo1_pin = 6
32 servo2_pin = 5
33 ir_pins = [22, 27]
34 GPIO.setup(servo1_pin, GPIO.OUT)
35 GPIO.setup(servo2_pin, GPIO.OUT)
36 for pin in ir_pins:
```

```
37      GPIO.setup(pin, GPIO.IN)
38
39  servo1 = GPIO.PWM(servo1_pin, 50)
40  servo2 = GPIO.PWM(servo2_pin, 50)
41  servo1.start(0)
42  servo2.start(0)
43
44  # === Variables globales ===
45  estado_clase = "FONDO"
46  angulo_servo1 = 0
47  angulo_servo2 = 0
48  estado_ir = [0, 0]
49  en_ejecucion = False
50  proceso_modelo = None
51  hilo_ir = None
52  ultimo_angulo_servo1 = 0
53  ultimo_angulo_servo2 = 0
54  NEUTRO = 90
55  btn_iniciar = None  # <-- CORREGIDO: Inicializar globales
56  btn_stop = None     # <-- CORREGIDO: Inicializar globales
57
58  # === MODELO UNICO ===
59  MODEL_FILENAME = "upse.eim"
60  MODEL_PATH = f"/home/saulo/vision/{MODEL_FILENAME}"
61  modelo_actual_firebase = MODEL_FILENAME
62
63  # === Clases del modelo ===
64  clases_por_modelo_unico = [
65      "GRANDE_CON_CABEZA", "GRANDE_SIN_CABEZA", "
        MEDIANO_CON_CABEZA",
66      "MEDIANO_SIN_CABEZA", "PEQUENO_CON_CABEZA", "
        PEQUENO_SIN_CABEZA", "FONDO"
67  ]
68
69  colores = {
70      "GRANDE_CON_CABEZA": "green",
71      "GRANDE_SIN_CABEZA": "darkgreen",
72      "MEDIANO_CON_CABEZA": "yellow",
73      "MEDIANO_SIN_CABEZA": "goldenrod",
74      "PEQUENO_CON_CABEZA": "orange",
75      "PEQUENO_SIN_CABEZA": "blue",
76      "FONDO": "gray"
77  }
```

```
78
79
80  # === MJPEG GLOBALES ===
81  camera_capture = None
82  mjpeg_server = None
83  MJPEG_PORT = 8000
84
85  # === MJPEG Stream ===
86  class MJPEGStreamHandler(BaseHTTPRequestHandler):
87      def do_GET(self):
88          global camera_capture
89          if self.path == '/stream.mjpeg':
90              self.send_response(200)
91              self.send_header('Content-type', 'multipart/x
                    -mixed-replace; boundary=--jpgboundary')
92              self.end_headers()
93              print("MJPEG: Cliente conectado al stream.")
94              try:
95                  while True:
96                      if camera_capture and camera_capture.
                            isOpened():
97                          ret, frame = camera_capture.read
                                ()
98                          if not ret:
99                              time.sleep(0.1)
100                             continue
101                         frame = cv2.resize(frame, (640,
                                480))
102                         ret, jpeg = cv2.imencode('.jpg',
                                frame)
103                         if not ret:
104                             time.sleep(0.1)
105                             continue
106                         self.wfile.write(b'--jpgboundary\
                                r\n')
107                         self.wfile.write(b'Content-type:
                                image/jpeg\r\n')
108                         self.wfile.write(b'Content-length
                                : %d\r\n\r\n' % jpeg.nbytes)
109                         self.wfile.write(jpeg.tobytes())
110                         self.wfile.write(b'\r\n')
111                         time.sleep(0.03)
112                     else:
```

```
113                         self.wfile.write(b'--jpgboundary\
                               r\n')
114                         self.wfile.write(b'Content-type:
                               text/plain\r\n\r\n')
115                         self.wfile.write(b'Camera not
                               available\r\n')
116                         time.sleep(1)
117             except Exception as e:
118                 print(f"MJPEG: Error en el stream: {e}")
119         else:
120             self.send_error(404)
121
122 class ThreadedHTTPServer(ThreadingMixIn, HTTPServer):
123     daemon_threads = True
124
125 def run_mjpeg_server():
126     global mjpeg_server
127     try:
128         s = socket.socket(socket.AF_INET, socket.
                SOCK_DGRAM)
129         s.connect(("8.8.8.8", 80))
130         pi_ip = s.getsockname()[0]
131         s.close()
132         print(f"MJPEG: Iniciando servidor en http://{
                pi_ip}:{MJPEG_PORT}/stream.mjpeg")
133         mjpeg_server = ThreadedHTTPServer(("", MJPEG_PORT
                ), MJPEGStreamHandler)
134         mjpeg_server.serve_forever()
135     except Exception as e:
136         print(f"MJPEG: Error al iniciar el servidor MJPEG
                : {e}")
137
138 # === Funciones de control ===
139 def mover_servo(servo, angulo):
140     global angulo_servo1, angulo_servo2
141     duty = 2 + ((NEUTRO + angulo) / 18)
142     duty = max(2, min(12, duty))
143     servo.ChangeDutyCycle(duty)
144     time.sleep(0.5)
145     servo.ChangeDutyCycle(0)
146     if servo == servo1:
147         angulo_servo1 = angulo
148     else:
```

```
149             angulo_servo2 = angulo
150         ref.update({'servo1_angle': angulo_servo1, '
                servo2_angle': angulo_servo2})
151
152 def leer_sensores_ir():
153     global estado_ir, en_ejecucion
154     while en_ejecucion:
155         nuevos_estado_ir = [GPIO.input(pin) for pin in
                ir_pins]
156         nuevos_estado_ir_norm = [1 if val else 0 for val
                in nuevos_estado_ir]
157         if nuevos_estado_ir_norm != estado_ir:
158             estado_ir = nuevos_estado_ir_norm
159             ref.update({'ir_sensors': estado_ir})
160         time.sleep(0.1)
161
162 def ejecutar_modelo():
163     global estado_clase, proceso_modelo, en_ejecucion
164     if proceso_modelo:
165         proceso_modelo.terminate()
166         proceso_modelo = None
167
168     proceso_modelo = subprocess.Popen(
169         ['/usr/bin/edge-impulse-linux-runner', '--model',
                MODEL_FILENAME,
170          '--video', '/dev/video0', '--raw', '--silent'],
171         stdout=subprocess.PIPE,
172         stderr=subprocess.STDOUT,
173         universal_newlines=True,
174         bufsize=1
175     )
176
177     ultima_clase = "FONDO"
178     predicciones = {}
179     class_line_re = re.compile(r'\s*(\w+):\s*([0-9.]+),?'
            )
180     for line in proceso_modelo.stdout:
181         if not en_ejecucion:
182             break
183         line = line.strip()
184         match = class_line_re.match(line)
185         if match:
186             clase = match.group(1)
```

```
187                 valor = float(match.group(2))
188                 predicciones[clase] = valor
189                 continue
190             if line == '}' and predicciones:
191                 clase_detectada = max(predicciones, key=
                        predicciones.get)
192                 if clase_detectada != ultima_clase:
193                     ultima_clase = clase_detectada
194                     estado_clase = clase_detectada
195                     print(f"? Clase: {clase_detectada}")
196                     ref.update({'detected_class':
                            estado_clase})
197                     if clase_detectada in ["GRANDE_SIN_CABEZA
                            ", "GRANDE_CON_CABEZA"]:
198                         mover_servo(servo1, 0)
199                         mover_servo(servo2, 50)
200                     elif clase_detectada in ["
                            MEDIANO_SIN_CABEZA", "
                            MEDIANO_CON_CABEZA"]:
201                         mover_servo(servo1, -35)
202                         mover_servo(servo2, 35)
203                     elif clase_detectada in ["
                            PEQUENO_SIN_CABEZA", "
                            PEQUENO_CON_CABEZA"]:
204                         mover_servo(servo1, -50)
205                         mover_servo(servo2, 0)
206                 predicciones = {}
207     if proceso_modelo:
208         proceso_modelo.terminate()
209         proceso_modelo = None
210
211 def iniciar_procesos():
212     global en_ejecucion, hilo_ir, btn_iniciar, btn_stop #
            Asegurarse de que se accede a los globales
213     if not en_ejecucion:
214         en_ejecucion = True
215         hilo_ir = threading.Thread(target=
                leer_sensores_ir, daemon=True)
216         hilo_ir.start()
217         threading.Thread(target=ejecutar_modelo, daemon=
                True).start()
218         threading.Thread(target=run_mjpeg_server, daemon=
                True).start()
```

```
219
220         if btn_iniciar: # <-- CORREGIDO: Comprobar si el
                boton existe
221             btn_iniciar.config(bg="gray")
222         if btn_stop:    # <-- CORREGIDO: Comprobar si el
                boton existe
223             btn_stop.config(bg="#f44336")
224
225         s = socket.socket(socket.AF_INET, socket.
                SOCK_DGRAM)
226         s.connect(("8.8.8.8", 80))
227         pi_ip = s.getsockname()[0]
228         s.close()
229         ref.update({
230             'status': 'online',
231             'detected_class': estado_clase,
232             'servo1_angle': angulo_servo1,
233             'servo2_angle': angulo_servo2,
234             'ir_sensors': estado_ir,
235             'model_used': modelo_actual_firebase,
236             'model_classes': clases_por_modelo_unico,
237             'model_colors': {cls: colores.get(cls, "black
                    ") for cls in clases_por_modelo_unico},
238             'mjpeg_stream_url': f"http://{pi_ip}:{
                    MJPEG_PORT}/stream.mjpeg"
239         })
240
241 def detener_procesos():
242     global en_ejecucion, proceso_modelo, mjpeg_server,
            btn_iniciar, btn_stop # Asegurarse de que se
            accede a los globales
243     en_ejecucion = False
244     mover_servo(servo1, 0)
245     mover_servo(servo2, 0)
246     if proceso_modelo:
247         proceso_modelo.terminate()
248         proceso_modelo = None
249     if mjpeg_server:
250         mjpeg_server.shutdown()
251         mjpeg_server.server_close()
252         mjpeg_server = None
253
254     if btn_iniciar: # <-- CORREGIDO: Comprobar si el
```

```
        boton existe
255         btn_iniciar.config(bg="#4caf50")
256     if btn_stop:    # <-- CORREGIDO: Comprobar si el
        boton existe
257         btn_stop.config(bg="gray")
258 
259     ref.update({
260         'status': 'offline',
261         'detected_class': 'FONDO',
262         'servo1_angle': 0,
263         'servo2_angle': 0,
264         'ir_sensors': [0, 0],
265         'mjpeg_stream_url': ''
266     })
267 
268 # === GUI ===
269 def iniciar_gui():
270     global btn_iniciar, btn_stop, camera_capture,
        estado_clase, angulo_servo1, angulo_servo2,
        estado_ir # Anadir estado_clase, etc.
271     root = tk.Tk()
272     root.title("Clasificacion de Camarones - UPSE")
273     root.geometry("1200x650")
274     root.configure(bg="#f0f0f0")
275     estilo_general = {"font": ("Arial", 11), "bg": "#
        f0f0f0"}
276 
277     frame_principal = tk.Frame(root, bg="#f0f0f0")
278     frame_principal.pack(fill="both", expand=True, padx
        =10, pady=10)
279 
280     frame_det = tk.LabelFrame(frame_principal, text="
        DETECCION", padx=10, pady=10, **estilo_general, fg
        ="#333")
281     frame_det.grid(row=0, column=0, padx=(0, 10), sticky=
        "n")
282 
283     video_label = tk.Label(frame_det)
284     video_label.grid(row=0, column=0, columnspan=2, pady
        =5)
285 
286     camera_capture = cv2.VideoCapture(0)
287     if not camera_capture.isOpened():
```

```
288         messagebox.showerror("Error de Camara", "No se
                pudo abrir la camara.")
289         root.destroy()
290         return
291 
292     def actualizar_camara_tkinter():
293         # en_ejecucion solo controla los hilos de
                procesos, no la GUI en s
294         if camera_capture and camera_capture.isOpened():
295             ret, frame = camera_capture.read()
296             if ret:
297                 frame_display = cv2.resize(frame, (320,
                        240))
298                 frame_display = cv2.cvtColor(
                        frame_display, cv2.COLOR_BGR2RGB)
299                 img = Image.fromarray(frame_display)
300                 imgtk = ImageTk.PhotoImage(image=img)
301                 video_label.imgtk = imgtk
302                 video_label.config(image=imgtk)
303         root.after(30, actualizar_camara_tkinter) #
                Siempre programar el proximo update de GUI
304 
305     actualizar_camara_tkinter()
306 
307     tk.Label(frame_det, text="Clase detectada:", **
            estilo_general).grid(row=1, column=0, sticky="e",
            pady=5)
308     var_clase = tk.StringVar(value="FONDO")
309     tk.Label(frame_det, textvariable=var_clase, font=("
            Arial", 14, "bold"),
310              bg="#f0f0f0", fg="#000").grid(row=1, column
                    =1, sticky="w")
311 
312     leyenda = tk.LabelFrame(frame_det, text="LEYENDA DE
            COLORES", **estilo_general)
313     leyenda.grid(row=2, column=0, columnspan=2, pady=10)
314     for nombre in clases_por_modelo_unico:
315         fila = tk.Frame(leyenda, bg="#f0f0f0")
316         fila.pack(anchor='w', pady=2)
317         c = tk.Canvas(fila, width=20, height=20, bg="#
                f0f0f0", highlightthickness=0)
318         c.create_rectangle(0, 0, 20, 20, fill=colores.get
                (nombre, "black"), outline=colores.get(nombre,
```

```
            "black"))
319         c.pack(side="left")
320         tk.Label(fila, text=nombre, **estilo_general).
               pack(side="left", padx=5)
321
322     frame_sensores_servos = tk.Frame(frame_principal, bg=
           "#f0f0f0")
323     frame_sensores_servos.grid(row=0, column=1, padx=(0,
           10), sticky="n")
324
325     frame_ir = tk.LabelFrame(frame_sensores_servos, text=
           "SENSORES INFRARROJOS", padx=10, pady=10, **
           estilo_general)
326     frame_ir.pack(pady=5, fill="x")
327     ir_valores, ir_circulos = [], []
328     for i in range(2):
329         tk.Label(frame_ir, text=f"S{i+1}", **
               estilo_general).grid(row=0, column=i)
330         canvas_ir = tk.Canvas(frame_ir, width=30, height
               =30, bg="#f0f0f0", highlightthickness=0)
331         canvas_ir.grid(row=1, column=i, pady=5)
332         circ = canvas_ir.create_oval(5, 5, 25, 25, fill="
               gray")
333         ir_circulos.append((canvas_ir, circ))
334         val = tk.StringVar(value="0")
335         tk.Label(frame_ir, textvariable=val, **
               estilo_general).grid(row=2, column=i, pady=2)
336         ir_valores.append(val)
337
338     frame_servos = tk.LabelFrame(frame_sensores_servos,
           text="SERVOS", padx=10, pady=10, **estilo_general)
339     frame_servos.pack(pady=5, fill="x")
340     var_s1 = tk.StringVar(value="Servo 1: 0")
341     var_s2 = tk.StringVar(value="Servo 2: 0")
342     tk.Label(frame_servos, textvariable=var_s1, **
           estilo_general).pack(anchor="w")
343     tk.Label(frame_servos, textvariable=var_s2, **
           estilo_general).pack(anchor="w")
344
345     canvas_s1 = tk.Canvas(frame_servos, width=100, height
           =60, bg="#f0f0f0", highlightthickness=0)
346     canvas_s1.pack(pady=5)
347     arco_s1 = canvas_s1.create_arc(10, 10, 90, 90, start
```

```
        =180, extent=0, style="arc", width=5, outline="
        blue")
348
349     canvas_s2 = tk.Canvas(frame_servos, width=100, height
        =60, bg="#f0f0f0", highlightthickness=0)
350     canvas_s2.pack(pady=5)
351     arco_s2 = canvas_s2.create_arc(10, 10, 90, 90, start
        =180, extent=0, style="arc", width=5, outline="red
        ")
352
353     frame_ctrl = tk.Frame(frame_principal, bg="#f0f0f0")
354     frame_ctrl.grid(row=0, column=2, sticky="n")
355
356     btn_iniciar = tk.Button(frame_ctrl, text=" Iniciar",
        command=iniciar_procesos,
357                             font=("Arial", 12, "bold"),
                                bg="#4caf50", fg="white",
                                width=12, height=2, bd=0)
358     btn_iniciar.pack(pady=(20, 10))
359
360     btn_stop = tk.Button(frame_ctrl, text=" Stop",
        command=detener_procesos,
361                          font=("Arial", 12, "bold"), bg="
                            #f44336", fg="white", width
                            =12, height=2, bd=0)
362     btn_stop.pack()
363
364     tk.Label(frame_ctrl, text=f"Modelo: {MODEL_FILENAME}"
        , font=("Arial", 12, "bold"),
365              fg="red", bg="#f0f0f0").pack(pady=10)
366
367     # Inicializaciin de Firebase con el modelo actual al
        inicio
368     s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
369     s.connect(("8.8.8.8", 80))
370     pi_ip = s.getsockname()[0]
371     s.close()
372     initial_mjpeg_stream_url = f"http://{pi_ip}:{
        MJPEG_PORT}/stream.mjpeg"
373
374     ref.update({
375         'model_used': modelo_actual_firebase,
376         'status': 'offline',
```

```
        'detected_class': 'FONDO',
        'servo1_angle': 0,
        'servo2_angle': 0,
        'ir_sensors': [0,0],
        'model_classes': clases_por_modelo_unico,
        'model_colors': {cls: colores.get(cls, "black")
            for cls in clases_por_modelo_unico},
        'mjpeg_stream_url': initial_mjpeg_stream_url
    })


    def actualizar_gui():
        var_clase.set(estado_clase)
        var_s1.set(f"Servo 1: {angulo_servo1} ") # <--
            CORREGIDO: Quitar asteriscos duplicados
        var_s2.set(f"Servo 2: {angulo_servo2} ") # <--
            CORREGIDO: Quitar asteriscos duplicados
        canvas_s1.itemconfig(arco_s1, extent=-(
            angulo_servo1))
        canvas_s2.itemconfig(arco_s2, extent=-(
            angulo_servo2))
        for i in range(2):
            color = "green" if estado_ir[i] == 1 else "
                gray"
            canvas_ir, circ = ir_circulos[i]
            canvas_ir.itemconfig(circ, fill=color)
            ir_valores[i].set(str(estado_ir[i]))
        root.after(300, actualizar_gui)

    actualizar_gui()
    root.mainloop()
    if camera_capture:
        camera_capture.release()
        print("Camara liberada al cerrar la GUI.")

if __name__ == '__main__':
    try:
        iniciar_gui()
    except KeyboardInterrupt:
        print("Detenido por el usuario.")
    finally:
        detener_procesos()
        servo1.stop()
```

```
414         servo2.stop()
415         GPIO.cleanup()
416         print("GPIO limpiado.")
```

Ejemplo de código Json

A continuación se muestra el código que Json para el envió de datos en tiempo real.

Listing 2: Código Json de ejemplo

```
1
2  <!DOCTYPE html>
3  <html lang="es">
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width,
           initial-scale=1.0">
7      <title>Dashboard en Vivo de Raspberry Pi</title>
8      <style>
9          body { font-family: Arial, sans-serif; margin: 20
               px; background-color: #f4f4f4; color: #333; }
10         .container { max-width: 1200px; margin: 0 auto;
               display: flex; flex-wrap: wrap; gap: 20px; }
11         .panel { background-color: #fff; border-radius: 8
               px; box-shadow: 0 2px 4px rgba(0,0,0,0.1);
               padding: 20px; flex: 1; min-width: 300px; }
12         h1, h2 { color: #0056b3; }
13         /* AHORA ESTILIZAMOS EL IFRAME */
14         #mjpegStreamFrame {
15             width: 100%;
16             max-width: 640px;
17             height: 480px; /* IMPORTANTE: IFRAME necesita
                   una altura definida */
18             background-color: black;
19             border-radius: 8px;
20             display: block;
21             margin-top: 10px;
22             border: none; /* Quitar borde por defecto del
                   iframe */
23         }
24         .data-item { margin-bottom: 10px; }
25         .data-item strong { display: inline-block; width:
               120px; }
```

```
26          .status-online { color: green; font-weight: bold; }
27          .status-offline { color: red; font-weight: bold; }
28          #ir_sensors span { margin-right: 10px; font-weight: bold; }
29          .legend-item { display: flex; align-items: center; margin-bottom: 5px; }
30          .legend-color-box { width: 20px; height: 20px; border-radius: 4px; margin-right: 10px; border: 1px solid #ccc; }
31          .legend-text { font-weight: bold; }
32          .detected-class-display { font-weight: bold; }
33          #videoStreamLink { margin-top: 10px; display: block; font-weight: bold; }
34      </style>
35  </head>
36  <body>
37      <div id="dashboard-content">
38          <div class="container">
39              <div class="panel">
40                  <h1>Dashboard de Raspberry Pi</h1>
41                  <div class="data-item"><strong>Estado General:</strong> <span id="status">Cargando...</span></div>
42              </div>
43
44              <div class="panel">
45                  <h2>Video en Vivo</h2>
46                  <!-- CAMBIO DE IMG A IFRAME -->
47                  <iframe id="mjpegStreamFrame" src="about:blank" style="display:none;" allow="camera;microphone" referrerpolicy="no-referrer"></iframe>
48                  <p>Estado del Stream MJPEG: <span id="mjpegStatus">Desconectado</span></p>
49                  <a id="videoStreamLink" href="#" target="_blank" style="display:none;">Abrir video en nueva pesta a</a>
50              </div>
51
52              <div class="panel">
53                  <h2>Datos de Sensores y Servos</h2>
```

```
54                  <div class="data-item"><strong>Clase
                        Detectada:</strong> <span id="
                        detectedClass" class="detected-class-
                        display">Cargando...</span></div>
55                  <div class="data-item"><strong> ngulo
                        Servo 1:</strong> <span id="
                        servo1Angle">Cargando...</span></div>
56                  <div class="data-item"><strong> ngulo
                        Servo 2:</strong> <span id="
                        servo2Angle">Cargando...</span></div>
57                  <div class="data-item"><strong>Sensores
                        IR:</strong> <span id="irSensors">
                        Cargando...</span></div>
58              </div>
59
60              <div class="panel">
61                  <h2>Configuraci n del Modelo</h2>
62                  <div class="data-item"><strong>Modelo
                        Actual:</strong> <span id="modelName">
                        Cargando...</span></div>
63                  <h3>Leyenda de Clases</h3>
64                  <div id="classLegend">
65                      <!-- La leyenda se generar
                            din micamente aqu  con
                            JavaScript -->
66                  </div>
67              </div>
68          </div>
69      </div>
70
71      <!-- Firebase SDKs -->
72      <script src="https://www.gstatic.com/firebasejs
            /9.6.0/firebase-app-compat.js"></script>
73      <script src="https://www.gstatic.com/firebasejs
            /9.6.0/firebase-database-compat.js"></script>
74      <script src="main.js"></script>
75
76  </body>
77  </html>
```

Ejemplo de código HTML

A continuación se muestra un fragmento e codigo realizado en HTML para poder realizar la interfaz grafica de la paguina web o tambien conocido como Website.

Listing 3: Código HTML de ejemplo

```

// Tu objeto de configuraci n de Firebase
const firebaseConfig = {
  apiKey: "AIzaSyD9xgdhxljCI2sS4UKUubY8tcRVgnS1pxc",
  authDomain: "emisor-rasperypi4-a-recep-pc.firebaseapp.com",
  databaseURL: "https://emisor-rasperypi4-a-recep-pc-default-rtdb.firebaseio.com",
  projectId: "emisor-rasperypi4-a-recep-pc",
  storageBucket: "emisor-rasperypi4-a-recep-pc.firebasestorage.app",
  messagingSenderId: "630095340200",
  appId: "1:630095340200:web:611d68bda35bc976cfd89d",
  measurementId: "G-HDN2S58S1Q"
};

// Inicializa Firebase
firebase.initializeApp(firebaseConfig);
console.log('Firebase initialized!');

const db = firebase.database();
const raspberryDataRef = db.ref('raspberry_data');
// Eliminamos webrtcSignalingRef

// Elementos del DOM para actualizar
const statusSpan = document.getElementById('status');
const detectedClassSpan = document.getElementById('detectedClass');
const servo1AngleSpan = document.getElementById('servo1Angle');
const servo2AngleSpan = document.getElementById('servo2Angle');
const irSensorsSpan = document.getElementById('irSensors');
// Elementos para el stream MJPEG
const mjpegStreamImg = document.getElementById('
```

```
    mjpegStream');
const mjpegStatusSpan = document.getElementById('
    mjpegStatus');
const videoStreamLink = document.getElementById('
    videoStreamLink');

const modelNameSpan = document.getElementById('modelName'
    );
const classLegendDiv = document.getElementById('
    classLegend');

let currentModelClasses = [];
let currentModelColors = {};

function generateClassLegend(classes, colors) {
    classLegendDiv.innerHTML = '';
    if (classes && classes.length > 0 && colors) {
        classes.forEach(className => {
            const color = colors[className] || '#333';
            const legendItem = document.createElement('
                div');
            legendItem.className = 'legend-item';
            legendItem.innerHTML = `
                <div class="legend-color-box" style="
                    background-color: ${color};"></div>
                <span class="legend-text">${className}</
                    span>
            `;
            classLegendDiv.appendChild(legendItem);
        });
    } else {
        classLegendDiv.innerHTML = '<p>No hay clases
            disponibles para el modelo seleccionado.</p>';
    }
}

// --- PARTE DE LECTURA DE DATOS DE RASPBERRY PI (RTDB)
    ---
raspberryDataRef.on('value', (snapshot) => {
    const data = snapshot.val();
    console.log('Datos de Firebase recibidos en el
        dashboard:', data);

```

```
63      if (data) {
64          statusSpan.textContent = data.status || '
                Desconocido';
65          statusSpan.className = data.status === 'online' ?
                'status-online' : 'status-offline';
66
67          modelNameSpan.textContent = data.model_used || '
                Desconocido';
68
69          if (data.model_classes && JSON.stringify(data.
                model_classes) !== JSON.stringify(
                currentModelClasses)) {
70              currentModelClasses = data.model_classes;
71              currentModelColors = data.model_colors || {};
72              generateClassLegend(currentModelClasses,
                    currentModelColors);
73          } else if (!data.model_classes &&
                currentModelClasses.length > 0) {
74              currentModelClasses = [];
75              currentModelColors = {};
76              generateClassLegend(currentModelClasses,
                    currentModelColors);
77          }
78
79          const detectedClass = data.detected_class || 'N/A
                ';
80          detectedClassSpan.textContent = detectedClass;
81          detectedClassSpan.style.color =
                currentModelColors[detectedClass] || '#333';
82
83          servo1AngleSpan.textContent = data.servo1_angle
                !== undefined ? data.servo1_angle + '  ' : 'N/
                A';
84          servo2AngleSpan.textContent = data.servo2_angle
                !== undefined ? data.servo2_angle + '  ' : 'N/
                A';
85
86          if (data.ir_sensors !== undefined && data.
                ir_sensors !== null) {
87              let irArray;
88              if (Array.isArray(data.ir_sensors)) {
89                  irArray = data.ir_sensors;
90              } else if (typeof data.ir_sensors === 'object
```

```
                ') {
91                  irArray = Object.keys(data.ir_sensors)
92                                  .sort((a, b) => parseInt(
                                        a) - parseInt(b))
93                                  .map(key => data.
                                        ir_sensors[key]);
94              } else {
95                  irArray = [];
96              }
97
98              if (irArray.length > 0) {
99                  irSensorsSpan.innerHTML = irArray.map((s,
                        i) => `<span>S${i+1}: ${s}</span>`).
                        join('');
100             } else {
101                 irSensorsSpan.textContent = 'N/A';
102             }
103         } else {
104             irSensorsSpan.textContent = 'N/A';
105         }
106
107         // --- MANEJO DEL STREAM MJPEG ---
108         const mjpegUrl = data.mjpeg_stream_url;
109         if (mjpegUrl) {
110             if (mjpegStreamImg.src !== mjpegUrl) { //
                    Solo actualizar si la URL ha cambiado
111                 mjpegStreamImg.src = mjpegUrl;
112                 mjpegStreamImg.style.display = 'block';
                        // Mostrar la imagen
113                 mjpegStatusSpan.textContent = 'Stream
                        activo';
114                 videoStreamLink.href = mjpegUrl; //
                        Asignar URL al link
115                 videoStreamLink.style.display = 'block';
                        // Mostrar el link
116                 console.log('MJPEG: URL de stream
                        actualizada:', mjpegUrl);
117             }
118         } else {
119             mjpegStreamImg.src = '';
120             mjpegStreamImg.style.display = 'none';
121             mjpegStatusSpan.textContent = 'Stream no
                    disponible';
```

```
122             videoStreamLink.href = '#';
123             videoStreamLink.style.display = 'none';
124         }
125 
126     } else {
127         // Valores por defecto si no hay datos (Pi
              desconectada o nodo vac o)
128         statusSpan.textContent = 'Desconectado';
129         statusSpan.className = 'status-offline';
130         detectedClassSpan.textContent = 'N/A';
131         detectedClassSpan.style.color = '#333';
132         servo1AngleSpan.textContent = 'N/A';
133         servo2AngleSpan.textContent = 'N/A';
134         irSensorsSpan.textContent = 'N/A';
135         modelNameSpan.textContent = 'Desconocido';
136 
137         currentModelClasses = [];
138         currentModelColors = {};
139         generateClassLegend(currentModelClasses,
              currentModelColors);
140 
141         // Limpiar el stream MJPEG
142         mjpegStreamImg.src = '';
143         mjpegStreamImg.style.display = 'none';
144         mjpegStatusSpan.textContent = 'Stream no
              disponible';
145         videoStreamLink.href = '#';
146         videoStreamLink.style.display = 'none';
147     }
148 });
```