UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

DESARROLLO DE DISPENSADOR AUTOMATICO DE ALIMENTO PARA TILAPIAS UTILIZANDO ESP32 CAM

ENRIQUEZ CORDOVA BRYAN ANDRES
INGENIERO DE SISTEMAS

ROMERO OCHOA MELANIE ROSARIO
INGENIERA DE SISTEMAS

MACHALA
2022

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

DESARROLLO DE DISPENSADOR AUTOMATICO DE ALIMENTO PARA TILAPIAS UTILIZANDO ESP32 CAM

ENRIQUEZ CORDOVA BRYAN ANDRES
INGENIERO DE SISTEMAS

ROMERO OCHOA MELANIE ROSARIO
INGENIERA DE SISTEMAS

MACHALA
2022

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

TRABAJO TITULACIÓN
PROPUESTAS TECNOLÓGICAS

DESARROLLO DE DISPENSADOR AUTOMATICO DE ALIMENTO PARA TILAPIAS UTILIZANDO ESP32 CAM

ENRIQUEZ CORDOVA BRYAN ANDRES
INGENIERO DE SISTEMAS

ROMERO OCHOA MELANIE ROSARIO
INGENIERA DE SISTEMAS

NOVILLO VICUÑA JOHNNY PAUL

MACHALA, 21 DE SEPTIEMBRE DE 2022

MACHALA
2022

TTI_Romero-Enriquez_2022-1

INFORME DE ORIGINALIDAD

8%	8%	1%	2%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	cdtechnologia.net Fuente de Internet	1%
2	remca.umet.edu.ec Fuente de Internet	1%
3	Submitted to Universidad de Costa Rica Trabajo del estudiante	1%
4	hdl.handle.net Fuente de Internet	<1%
5	www.sigmaelectronica.net Fuente de Internet	<1%
6	www.coursehero.com Fuente de Internet	<1%
7	www.industrialshields.com Fuente de Internet	<1%
8	1library.co Fuente de Internet	<1%
9	www.agricultura.gob.ec Fuente de Internet	<1%

CLÁUSULA DE CESIÓN DE DERECHO DE PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL

Los que suscriben, ENRIQUEZ CORDOVA BRYAN ANDRES y ROMERO OCHOA MELANIE ROSARIO, en calidad de autores del siguiente trabajo escrito titulado DESARROLLO DE DISPENSADOR AUTOMATICO DE ALIMENTO PARA TILAPIAS UTILIZANDO ESP32 CAM, otorgan a la Universidad Técnica de Machala, de forma gratuita y no exclusiva, los derechos de reproducción, distribución y comunicación pública de la obra, que constituye un trabajo de autoría propia, sobre la cual tienen potestad para otorgar los derechos contenidos en esta licencia.

Los autores declaran que el contenido que se publicará es de carácter académico y se enmarca en las dispociones definidas por la Universidad Técnica de Machala.

Se autoriza a transformar la obra, únicamente cuando sea necesario, y a realizar las adaptaciones pertinentes para permitir su preservación, distribución y publicación en el Repositorio Digital Institucional de la Universidad Técnica de Machala.

Los autores como garantes de la autoría de la obra y en relación a la misma, declaran que la universidad se encuentra libre de todo tipo de responsabilidad sobre el contenido de la obra y que asumen la responsabilidad frente a cualquier reclamo o demanda por parte de terceros de manera exclusiva.

Aceptando esta licencia, se cede a la Universidad Técnica de Machala el derecho exclusivo de archivar, reproducir, convertir, comunicar y/o distribuir la obra mundialmente en formato electrónico y digital a través de su Repositorio Digital Institucional, siempre y cuando no se lo haga para obtener beneficio económico.

Machala, 21 de septiembre de 2022

ENRIQUEZ CORDOVA BRYAN ANDRES
0705931624

ROMERO OCHOA MELANIE ROSARIO
0704988138

UNIVERSIDAD TECNICA DE MACHALA

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

TITULACIÓN

PROPUESTA TECNOLÓGICA:

DESARROLLO DE DISPENSADOR AUTOMÁTICO DE ALIMENTO PARA TILAPIAS UTILIZANDO ESP32-CAM

ESTUDIANTES:

BRYAN ANDRES ENRIQUEZ CORDOVA

MELANIE ROSARIO ROMERO OCHOA

TUTOR:

ING. JOHNNY PAUL NOVILLO VICUÑA

AÑO:

2022 – D2

DEDICATORIA

Este trabajo de titulación está dedicado principalmente a Dios, a mi Manuel Atiencie por haber sido mi compañía desde pequeña, aunque ya no lo tengo presencialmente sé que estaría muy orgulloso de mi.

A mi madrina Magdalena Serrano por su ayuda incondicional, consejos, y apoyo moral que me ha brindado para cumplir con todas las metas que me he propuesto.

A mis apreciados Bryan, Kevin, Diego y Ángel por acompañarme en toda la carrera universitaria, creer en mí, alentándome a seguir adelante, y jamás dejaré de agradecerles por tantas ayudas.

Srta. Melanie Rosario Romero Ochoa

Quiero dedicar este trabajo de tesis, tanto a mi abuelo Washington Enríquez, tío Marco Enríquez y prima Paula Pazmiño, aunque no están conmigo, pero sé que desde arriba deben estar apoyándome y orgullosos de mi.

También dedico mucho a mis seres queridos, mi mamá, papá y hermana y demás familiares que son muchos y siempre me han ayudado cuando lo he necesitado y me han apoyado en lo que he estudiado y están pendientes de mí.

Sr. Bryan Andrés Enríquez Córdova

AGRADECIMIENTO

Dar gracias a Dios por las bendiciones recibidas e iluminar mi camino en cada paso, por regalarme a mi señora abuela materna quien me ha estado presente en cada momento.

Agradezco a mi abuela materna Carmen Flores, que con su sacrificio ha dado todo para convertirme en lo que soy actualmente, se convirtió en la madre que necesitaba, gracias por los consejos, por la ayuda económica y su apoyo moral.

Agradezco a mis tíos maternos Sandra, Silvia y Arriton Ochoa por todo el apoyo a la distancia, por confiar en mí y animarme a seguir adelante.

Finalmente quiero agradecer a cada uno de los docentes de la Facultad de Ingeniería de Sistemas, por brindarme sus conocimientos y guías durante todo el tiempo de universidad, en especial al Ing. Johnny Novillo por darme la oportunidad y la confianza de realizar esta propuesta tecnológica.

Srta. Melanie Rosario Romero Ochoa

Quiero agradecer en primer lugar a Dios, por darme las fuerzas y paciencia necesaria para seguir por el buen camino de mi vida personal y profesional.

Agradezco a mis padres por apoyarme con todo lo que he necesitado, darme la facilidad de continuar con mis estudios, gracias a ellos he llegado hasta este punto más de una meta.

También quiero agradecer a mi familia tanto del paterno como materna, por todo el apoyo y ánimos que me han dado, a pesar de duros momentos que he pasado, siempre han estado pendiente de mis estudios y alentarme.

Por último, quiero agradecer tanto a la carrera de Ingeniería de Sistemas y a todos los docentes por brindar sus enseñanzas y conocimientos que me ayudaron mucho en mi día a día de universidad y en especial agradecer a mi tutor el Ing. Johnny Novillo, por brindarme su asesoría y su tiempo para poder realizar este proyecto.

Sr. Bryan Andrés Enríquez Córdova

RESUMEN

Los dispensadores o comúnmente llamados comederos inteligentes, nacieron de la idea de tener mascotas más saludables a través de una alimentación adecuada, distribuida por raciones. Derivaron de las grandes industrias dedicadas a la acuicultura, que buscaban que peces y mariscos gocen de buena salud y un peso óptimo, para de esta forma maximizar ganancias y disminuir pérdidas. En el caso de las mascotas se busca que las mismas tengan una buena salud, para de esta manera mejorar su calidad de vida y reducir gastos veterinarios.

Actualmente en el mercado existen muchos alimentadores de peces, los cuales, al tener un tamaño desproporcionado, muchas veces desmotivan a las personas para adquirirlos, lo que conlleva a seguir manteniendo una alimentación manual.

La alimentación de forma manual o forma tradicional, puede llevar al comportamiento anormal de las mascotas, y en el caso de la producción de peces y mariscos, a un crecimiento lento y con población mayoritariamente enferma.

En la actualidad, la mayoría de productores dedicados a la crianza y comercialización de peces y mariscos, han dependido de la experiencia e intuición para la alimentación o predicción de las enfermedades. Pero esto puede ser automatizado, gracias a los avances en tecnología e inteligencia artificial, que permiten automatizar procesos, reconocer objetos, medir variables físicas, entre otros.; lo que conducirá a un mejor control de la producción.

Para el desarrollo del programa que controlará el alimentador, se utilizó el lenguaje de programación en el entorno de desarrollo integrado Arduino IDE, el cual utiliza el lenguaje C++, que contiene una gran variedad de ejemplos y versiones disponibles en su amplia librería de ESP32. El componente del prototipo, que se encarga de dispensar el alimento para los peces, está compuesto por un motor FD RS550, que será activado a través de un controlador de motor L298N. De esta forma se logra de manera fácil, controlar la velocidad del motor, y distribuir el voltaje necesario hacia el módulo ESP32-CAM.

Se ha considerado la autonomía energética para el desarrollo del prototipo, gracias a la utilización de un panel solar, que cumple la función de aprovechar la energía solar para recargar una batería, que servirá de fuente de alimentación para el equipo.

El presente trabajo de titulación detalla el desarrollo de un prototipo de dispensador de alimentos automático para tilapias utilizando el módulo ESP32-CAM y empleando como fuente de alimentación una batería, recargable a través de un panel solar. Recurriendo al Internet de las Cosas este dispositivo se conecta a través de una dirección IP estática

dentro una red local WiFi, para observar el comportamiento de las tilapias con la ayuda de su cámara integrada OV2640 y además poder alimentar a los ejemplares en el momento actual o dentro de un rango de tiempo programable.

También incluye un temporizador para el accionamiento del motor que permitirá dispensar la cantidad de alimento requerida desde los 5 hasta los 160 gramos. Además, se aplicó inteligencia artificial que posee el módulo ESP32-CAM, para detectar a las tilapias en tiempo real, dentro de la piscina de crianza.

Palabras clave: esp32-cam**,** dispensador, tilapia, inteligente, automático.

ABSTRACT

The dispensers or commonly called smart feeders, were born from the idea of having healthier pets through proper feeding, distributed by rations. They were derived from the large aquaculture industries, which were looking for fish and seafood to enjoy good health and optimum weight, in order to maximize profits and reduce losses. In the case of pets, the aim is for them to be in good health in order to improve their quality of life and reduce veterinary expenses.

Currently, there are many fish feeders in the market, which are disproportionate in size, often discouraging people to buy them, which leads to continue to maintain a manual feeding.

Manual or traditional feeding can lead to abnormal pet behavior, and in the case of fish and shellfish production, to slow growth and mostly diseased stock.

Currently, most fish and shellfish farmers have relied on experience and intuition for feeding or predicting disease. But this can be automated, thanks to advances in technology and artificial intelligence, which allow to automate processes, recognize objects, measure physical variables, among others, which will allow a better control of production.

For the development of the program that will control the feeder, the programming language was used in the integrated development environment Arduino IDE, which uses the C++ language, which contains a wide variety of examples and versions available in its extensive ESP32 library. The prototype component, which is responsible for dispensing the fish feed, is composed of an FD RS550 motor, which will be activated through an L298N motor controller. In this way, it is easy to control the speed of the motor and to distribute the necessary voltage to the ESP32-CAM module.

Energy autonomy has been considered for the development of the prototype, thanks to the use of a solar panel, which fulfills the functions of harnessing solar energy and recharging the battery, which will serve as a power source for the equipment.

This degree work details the development of a prototype of an automatic food dispenser for tilapia using the ESP32-CAM module and using a battery as a power source, rechargeable through a solar panel. Using the Internet of Things, this device connects through a static IP address within a local WiFi network, to observe the behavior of the tilapia with the help of its integrated camera OV2640 and also to feed the specimens at the current time or within a programmable time range.

It also includes a timer for the motor drive that will allow dispensing the required amount of feed from 5 to 160 grams. In addition, the artificial intelligence of the ESP32-CAM module was applied to detect the tilapia in real time in the rearing pool.

Keywords: esp32-cam, dispenser, tilapia, intelligent, automatic, automatic

ÍNDICE

Portada 1
ABSTRACT 6
ÍNDICE DE ILUSTRACIONES 10
ÍNDICE DE TABLAS 12
ÍNDICE DE ANEXOS 13
INTRODUCCIÓN 15
CAPÍTULO I. DIAGNÓSTICO DE NECESIDADES Y REQUERIMIENTOS. 17
1.1. ÁMBITO DE APLICACIÓN: DESCRIPCIÓN DEL CONTEXTO Y HECHOS DE INTERÉS. 17
1.2. ESTABLECIMIENTO DE REQUERIMIENTOS. 19
1.3. JUSTIFICACIÓN DEL REQUERIMIENTO A SATISFACER. 20
CAPITULO II. DESARROLLO DEL PROTOTIPO. 21
2.1. DEFINICIÓN DEL PROTOTIPO TECNOLÓGICO. 21
2.2. FUNDAMENTACIÓN TEÓRICA DEL PROTOTIPO. 21
2.2.1. ESP32 21
2.2.1.1. ESP32-S2 22
2.2.2. ESP32-CAM 23
2.2.2.1. Características 24
2.2.2.2. Especificaciones 25
2.2.2.3. Diagrama Esquemático 25
2.2.2.4. Diagrama de pines 26
2.2.2.4.1. Descripción de pines 26
2.2.3. Arduino IDE 27
2.2.4. OPENCV 27
2.2.5. YOLOv3 28
2.2.5.1. Modelo Mobilenet 28
2.2.6. Modulo L298N 29
2.3. OBJETIVOS DEL PROTOTIPO. 31
2.3.1. Objetivo General 31

2.3.2. Objetivos Específicos 31

2.4. DISEÑO DEL PROTOTIPO. 31

2.4.1. Componentes Electrónicos 31

2.5. Alimentación 32

2.5.1. Periodo de alimentación 33

2.6. EJECUCIÓN Y/O ENSAMBLAJE DEL PROTOTIPO. 35

2.6.1. Configuración básica del ESP32-CAM en Arduino IDE 35

2.6.2. Configuración y Ejecución de Código 37

2.6.3. Descripción de Código 39

2.6.3.1. Arduino IDE 39

2.6.3.2. Python 42

2.6.4. Ensamblaje del Prototipo 44

CAPÍTULO III. EVALUACIÓN DEL PROTOTIPO. 52

3.1. PLAN DE EVALUACIÓN. 52

3.2. RESULTADOS DE LA EVALUACIÓN. 52

CONCLUSIONES 55

RECOMENDACIONES 56

BIBLIOGRAFÍA 57

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Modelos de módulos de ESP32 22
Ilustración 2: Módulo Espressif 22
Ilustración 3: Dimensiones del Módulo ESP32S 23
Ilustración 4: ESP32-CAM 23
Ilustración 5: ESP32-CAM con cámara OV2640 24
Ilustración 6: Diagrama Esquemático ESP32-CAM 25
Ilustración 7: Diagrama de pines ESP32-CAM 26
Ilustración 8: Almohadillas de conexión 3.3V o 5V 26
Ilustración 9: Diagrama de Modelo Mobilenet 28
Ilustración 10: Modulo Controlador de Motor L298N 29
Ilustración 11: Diagrama del circuito interno del L298N 30
Ilustración 12: Modelo en varias vistas 32
Ilustración 13: Conexión pasiva del ESP32-CAM 35
Ilustración 14: Preferencias en Arduino IDE 35
Ilustración 15: Menú para configurar el gestor de tarjetas en Arduino IDE 36
Ilustración 16: Gestor de Tarjetas 36
Ilustración 17: Selección de placa ESP32 Wrover Module 37
Ilustración 18: Configuración necesaria para cargar código 37
Ilustración 19: Subir código al ESP32-CAM 38
Ilustración 20: Proceso de subida de código en Arduino IDE 38
Ilustración 21: Monitor Serie 39
Ilustración 22: Diagrama Estructural 39
Ilustración 23: Configuración de red WiFi y Dirección IP estática 40
Ilustración 24: Configuración de red correcta 40
Ilustración 25: Procedimiento para declarar el ciclo de trabajo 40
Ilustración 26: Indicaciones de PWM 41
Ilustración 27: Procedimiento para correr alimentador 41
Ilustración 28: Version de Python para la IA 42
Ilustración 29: Librerías de Python utilizadas. 42
Ilustración 30: Configuración de IP y Ventana. 43
Ilustración 31: Creación del Modelo de Detección. 43
Ilustración 32: Ciclo repetitivo de detección. 43
Ilustración 33: Transportador de tornillo sin fin. 44
Ilustración 34: Medidas previas del modelo del dispensador. 45
Ilustración 35: Proceso de Moldeado del Transportador. 45

Ilustración 36: Proceso de cortes del transportador. 46
Ilustración 37: Proceso de Doblaje 46
Ilustración 38: Proceso de Lijado 46
Ilustración 39: Modelo del Tornillo sin fin 47
Ilustración 40: Proceso de diseño de la tolva con acrílico 47
Ilustración 41: Proceso de Ensamblaje final. 47
Ilustración 42: Proceso de Ensamblaje Final Vista Superior. 48
Ilustración 43: Soporte de dispensador. 48
Ilustración 44: Pruebas del panel solar. 49
Ilustración 45: Pruebas de Voltaje al Panel Solar. 49
Ilustración 46: Caja contenedora de los componentes electrónicos. 50
Ilustración 47: Caja contendora de componentes electrónicos. 50
Ilustración 48: Pruebas y Ensamble Final. 51
Ilustración 49: Ensamblaje terminado y funcional. 51

ÍNDICE DE TABLAS

Tabla 1: Configuración de Pines del L298N 30
Tabla 2: Componentes electrónicos utilizados en el prototipo 31
Tabla 3: Uso Recomendado de balanceado PRONACA. 34
Tabla 4: Cantidad de alimento dispensado con tolva llena 52
Tabla 5: Tabla de alimentación para tilapias en base a NICOVITA - VITAPRO 53
Tabla 6: Pruebas de funcionamiento del prototipo 54

ÍNDICE DE ANEXOS

Anexo 1: Cantidad de comida en proceso manual 62
Anexo 2: Tiempo de expulsión de comida del dispensador 62
Anexo 3: Tiempo de alimentación manual 63
Anexo 4: Cantidad de comida expulsada por dispensador a los 10s 63
Anexo 5: Cantidad de comida expulsada por dispensador a los 30s 64
Anexo 6: Proceso de instalación de dispensador 64
Anexo 7: Instalación y prueba del prototipo 65
Anexo 8: Código de AlimentadorESP32CAM (1) 65
Anexo 9: Código de AlimentadorESP32CAM (2) 66
Anexo 10: Código de AlimentadorESP32CAM (3) 67
Anexo 11: Código de AlimentdorESP32CAM (4) 67
Anexo 12: Código de app_httpd.cpp (1) 68
Anexo 13: Código de app_httpd.cpp (2) 68
Anexo 14: Código de app_httpd.cpp (3) 69
Anexo 15: Código de app_httpd.cpp (4) 69
Anexo 16: Código de app_httpd.cpp (5) 70
Anexo 17: Código de app_httpd.cpp (6) 70
Anexo 18: Código de app_httpd.cpp (7) 71
Anexo 19: Código de app_httpd.cpp (8) 71
Anexo 20: Código de app_httpd.cpp (9) 72
Anexo 21: Código de app_httpd.cpp (10) 72
Anexo 22: Código de app_httpd.cpp (11) 73
Anexo 23: Código de app_httpd.cpp (12) 73
Anexo 24: Código de app_httpd.cpp (13) 74
Anexo 25: Código de app_httpd.cpp (14) 74
Anexo 26: Código de app_httpd.cpp (15) 75
Anexo 27: Código de app_httpd.cpp (16) 75
Anexo 28: Código de app_httpd.cpp (17) 76
Anexo 29: Código de app_httpd.cpp (18) 76
Anexo 30: Código de app_httpd.cpp (19) 77
Anexo 31: Código de app_httpd.cpp (20) 77
Anexo 32: Código de app_httpd.cpp (21) 78
Anexo 33: Código de app_httpd.cpp (22) 78
Anexo 34: Código del app_httpd.cpp (23) 79
Anexo 35: Código del app_httpd.cpp (24) 79

Anexo 36: Código del app_httpd.cpp (25) 80
Anexo 37: Código de camera_index.h (1) 80
Anexo 38: Código de camera_index.h (2) 81
Anexo 39: Código de camera_index.h (3) 81
Anexo 40: Código de camera_index.h (4) 82
Anexo 41: Código de camera_index.h (5) 82
Anexo 42:Código de camera_index.h (6) 83
Anexo 43: Código de camera_index.h (7) 83
Anexo 44: Código de camera_index.h (8) 84
Anexo 45: Código de camera_index.h (9) 84
Anexo 46: Código de camera_index.h (10) 85
Anexo 47: Código de camera_index.h (11) 85
Anexo 48: Código de camera_index.h (12) 86
Anexo 49: Código de camera_index.h (13) 86
Anexo 50: Código de camera_index.h (14) 87
Anexo 51: Código de camera_index.h (15) 87
Anexo 52: Código de camera_index.h (16) 88
Anexo 53: Código de camera_index.h (17) 88
Anexo 54: Código de camera_index.h (18) 89
Anexo 55: Código de camera_index.h (19) 89
Anexo 56: Código de camera_pins.h (1) 90
Anexo 57: Código de camera_pins.h (2) 90
Anexo 58: Código de camera_pins.h (3) 91
Anexo 59: Código de camera_pins.h (4) 91
Anexo 60: Código IPaddressClassification.py (1) 92
Anexo 61: Código IPaddressClassification.py (2) 92

INTRODUCCIÓN

El avance tecnológico acoge diversas áreas de negocios, como son la agricultura y la crianza de seres vivos para el consumo humano, de los cuales la tilapia en la acuicultura está considerada como el producto no petrolero de exportación rentable, esta representa una de las labores primordiales de índole económico del país, agregando su progresión cada vez se intensifica, junto con la crianza del camarón son los sectores acuícolas con más prosperidad en América latina.[1], [2]

La actividad de crianza de tilapia se impulsó en el año 1999 a causa de una caída en el sector camaronero, que fue responsabilidad de una enfermedad ocasionado por un virus denominado síndrome de la mancha blanca, de esta manera alcanzando máximos históricos en venta de exportación en 2007 de 27.315.395 libras. [3], [4]

Sin embargo, con todo avance surgen problemas, en este caso, esta actividad conlleva un control necesario sobre diversas variables en el entorno de desarrollo de la tilapia, debido a que, de no cumplir, la tasa de mortalidad de los mismo es muy alta.4 Una de las condiciones para aprovechar el crecimiento y la salud de la misma, se obtiene en la alimentación; su forma y raciones durante el día.

Con la implementación del internet de las cosas en el campo de la agricultura, llamándose agricultura inteligente [5], es necesario que se expanda el mismo concepto sobre las demás áreas de crianza [6], por eso el presente documento plantea el diseño y la implementación de un prototipo de bajo costo para la alimentación inteligente y automática de tilapias, utilizando un módulo ESP32-CAM, un tornillo sin fin, motor l298n y un programador FTDI FT232, los cuales monitorearan el comportamiento de las tilapias. Este documento se encuentra particionado en 3 capítulos que son descritos a continuación:

El **capítulo 1** se expresa el ámbito de la propuesta tecnológica, que conceptualiza el dictamen de las necesidades y requerimientos, se especifica la factibilidad de desarrollo del prototipo, también se determinan los tiempos y su debida justificación.

El **capítulo 2** explica las normas de avance del prototipo. Se concreta la propuesta tecnológica, con su fundamentación teórica de los componentes electrónicos usados, luego se determinan los objetivos, y, por último, partiendo del diseño del prototipo con todos los avances y configuraciones.

En el **capítulo 3** se ejecuta un plan de valoración del prototipo, con el uso de dispositivos de grabación para comparar el comportamiento del mismo con su propósito. Como paso final,

se explican las conclusiones y recomendaciones que emergieron en el desarrollo de la propuesta tecnológica.

CAPÍTULO I. DIAGNÓSTICO DE NECESIDADES Y REQUERIMIENTOS.

1.1. ÁMBITO DE APLICACIÓN: DESCRIPCIÓN DEL CONTEXTO Y HECHOS DE INTERÉS.

La acuicultura como actividad interdisciplinaria representa una empresa productiva que aprovecha sus conocimientos de biología, ingeniería y ecología. Para solucionar los problemas nutricionales, existen varios tipos de organismos acuícolas, uno de los más desarrollados es la acuicultura o cultivo de peces, de los cuales el pez más utilizado en el mundo es la tilapia.

Originario y endémico del África y cercano Oriente, la tilapia es el nombre común en el cual se conocen a las diversas especies del género Oreochromis, peces de agua dulce que posee características importantes como su rápido crecimiento y reproducción, además de la gran capacidad de adaptabilidad que poseen, por consiguiente a comienzo del siglo XIX en el Congo Belga, se inician a investigarlas para incluirlas en la piscicultura rural, debido a sus excelentes resultados se inicia el progresivo cultivo en 1924 a diferentes partes del mundo. [7]

Según [8] El 19 de octubre de 1965 fue introducida desde Colombia al Ecuador la tilapia mossambica (*Oreochromis mossambicus*) para la zona de la ahora llamada Santo Domingo de los Tsáchilas, pero por una ruptura del muro perimetral del estanque la mayoría de los ejemplares escaparon. Los pocos peces recapturados, se transfirieron a la provincia de Imbabura, al lago Yaguarcocha situado a 2.253 metros sobre el nivel del mar.

Otros piscicultores particulares introducen desde Brasil en el año 1974 la tilapia nilótica (*Oreochromis niloticus*), pero la especie que predomina en los cultivos comerciales es el híbrido rojo de tilapia que fue introducida al país a inicios de los años 80.

Jácome menciona *Oreochromis sp*. fue introducida para su uso en prácticas de acuicultura en la provincia de Manabí en el río Chone, donde invade estructuras artesanales conocidas como "chameras" donde se lleva a cabo la pesca tradicional y la cría de especies nativas de "chame". (*Dormitator latifrons).* [4]

El cultivo de tilapia en el Ecuador es considerado como cultivos no tradicionales, es por eso que el Ministerio de Agricultura y Ganadería en 2018, aportaba con pequeños intensivos a los pequeños productores, que podían generar un promedio de 100 mil

tilapias con un peso por unidad de una libra en el programa de Innovación Tecnológica, Participativa y Productividad Agrícola. [9]

La producción de tilapia roja (Oreochromis spp. O) mantiene una importante participación en el mundo, con una producción superior al millón de toneladas. En Ecuador, la tilapia roja es el segundo cultivo con mayor participación económica, con la actividad económica más desarrollada en las provincias de Guayas y El Oro.

Según el estudio de Arboleda y otros: La tilapia tiene una cualidad a la hora de la compra que es su frescura; es por eso que es el segundo tipo de pescado más predilecto por los habitantes de Puerto Jelí, perteneciente al cantón Santa Rosa, de la provincia de El Oro. [10]

La acuicultura ha tomado gran importancia en el Ecuador, debido a la implementación tecnológica que le ha permitido desarrollarse exitosamente en especies de piscina como la Tilapia Roja. Grandes empresas del país han logrado posicionar al Ecuador como referente productor y exportador de filetes de tilapia siendo socios comerciales los siguientes países: Estados Unidos, España, Francia, Italia, China, Corea del Sur, Japón, Vietnam, Singapur, entre otros.

1.2. ESTABLECIMIENTO DE REQUERIMIENTOS.

El prototipo permitirá alimentar de manera automática y continua a tilapias a través de una conexión WiFi que se conecta a una red IP estática al ESP32-CAM el cual es el encargado de alimentar en el momento actual y/o intervalos de tiempo de alimentación.

Este prototipo puede ser controlado de manera remota mediante un navegador de internet, puede ser PC o smartphone ingresando la dirección IP proporcionada por el ESP32-CAM, facilitando la evaluación del funcionamiento del dispositivo en conjunto con la cámara integrada para un mayor monitoreo.

1.3. JUSTIFICACIÓN DEL REQUERIMIENTO A SATISFACER.

El modelado del prototipo de un dispensador de alimento para tilapias, surge de la necesidad para brindar ayuda a los propietarios y/o encargados de la acuicultura de tilapias al momento de alimentar, debido a situaciones de diario vivir se imposibilite la correcta distribución de la comida, mediante la utilización de un sitio web con una dirección IP que proporciona el microcomponente ESP32-CAM, donde permitirá administrar la alimentación de forma automática por intervalo de horas, la alimentación en el momento actual, además de poder visualizar mediante su cámara en tiempo real.

Con la visualización de la cámara en tiempo real contribuye un mejor control al momento de la alimentación de la tilapia, ya que podrá observar si todos los ejemplares que tiene a cargo están alimentándose de manera adecuada.

Los resultados de la evaluación demuestran que la medición de alimentación manual difiere, a causa de la proporción de la mano, tamaño, grosor, etc. El proceso de la alimentación manual es el siguiente: Una persona recorre el estanque distribuyendo la alimentación de manera no uniforme, puesto que, dependiendo de los factores climáticos, la comida puede no llegar hacia la tilapia y se quede en los muros, en la vegetación, entre otros, y por consiguiente exista un desperdicio del mismo.

CAPITULO II. DESARROLLO DEL PROTOTIPO.

1.1. DEFINICIÓN DEL PROTOTIPO TECNOLÓGICO.

La arquitectura del prototipo está compuesta de tres secciones: subida de código empleando el modo pasivo del ESP32-CAM, elaboración física del prototipo implementando el modo activo del microcontrolador y desarrollo e implementación de la detección de imagen en tiempo real a través del algoritmo YOLOv3.

1.2. FUNDAMENTACIÓN TEÓRICA DEL PROTOTIPO.

El prototipo está constituido teóricamente por los siguientes componentes electrónicos:

1.2.1. ESP32

Como lo detallan Romero y Elustondo [11] en los últimos años han surgido investigaciones, casos de estudios y productos que emplean la placa ESP32 como eje principal, ha servido para que evolucionen a modelos más complejos los sistemas ciber-físicos.

Según [12] el ESP32 cuenta con un chip de doble núcleo de Espressif que combina Bluetooth de 2,4 GHz y WiFi que son capacidades inalámbricas. Espressif ofrece una amplia gama de módulos totalmente certificados con sus propios System on a Chip (**SoC**) avanzados.

Diseñado con una tecnología de ultra bajo consumo de 40 nm de TSMC, logra el mejor rendimiento de potencia y radio frecuencia (RF), que ante una gran y amplia variedad de aplicaciones y escenarios de potencia muestra una robustez, versatilidad y fiabilidad.

ESP32 utiliza CMOS (Complementary Meta Oxide Semiconductor Memory) para la radio y la banda de base que están totalmente integradas en un solo chip, que a su vez integra circuitos de calibración avanzados que permiten que la solución elimine las imperfecciones del circuito externo o se ajuste a los cambios en las condiciones externas.

En el mercado existen una gran variedad de ESP32, pero estas difieren en algunos aspectos como, por ejemplo, número de GPIOs accesibles, interfaz USB a UART, botones BOOT y RESET, conector de batería, además de otras características como: pantalla OLED integrada chips LoRa, tarjeta SD, etc.

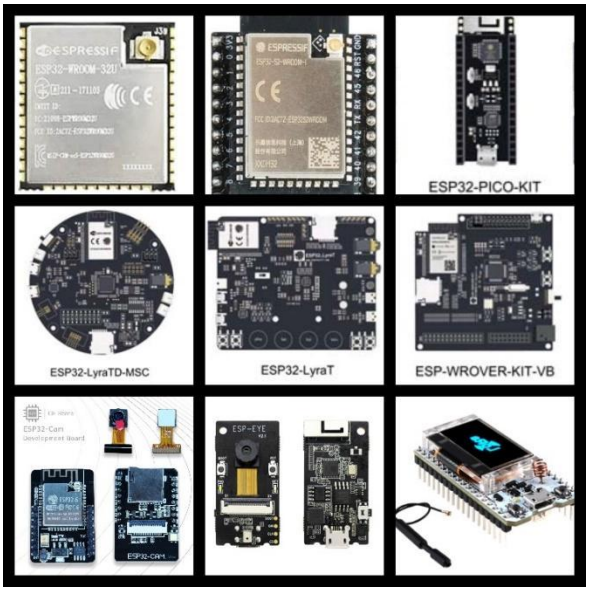


Ilustración 1: Modelos de módulos ESP32
Fuente: [13]

1.2.1.1. ESP32-S2

Como lo señala Espressif [13]:La serie ESP32-S2 es una solución Wi-Fi de 2,4 GHz de bajo consumo en un chip (SoC). Con un rendimiento de energía y radiofrecuencia líder en la industria, el SoC es una opción ideal para una amplia gama de aplicaciones de Internet de las cosas (IoT), computación y hogar inteligente.

La serie ESP32-S2 incluye un subsistema Wi-Fi que integra Wi-Fi MAC, radio Wi-Fi, banda base, conmutador RF, amplificador de potencia, amplificador

Ilustración 2: Módulo Espressif
Fuente: [14]

de bajo ruido (LNA), etc. Este chip es totalmente compatible con IEEE 802.11b/g/n y proporciona una solución Wi-Fi completa.

Su reloj de grano fino, el voltaje dinámico y el escalado de y frecuencia, y la potencia de salida del amplificador de potencia contribuyen a un equilibrio óptimo entre el rango de comunicación, la velocidad de datos y el consumo de energía.

El módulo ESP32-S genérico, el módulo tiene el tamaño más competitivo del mercado con las siguientes dimensiones:

- Altura: 25.5 mm
- Ancho: 18 mm

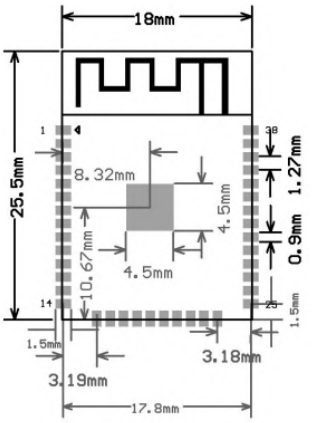


Ilustración 3: Dimensiones del Módulo ESP32S
Fuente: [15]

1.2.2. ESP32-CAM

Es una placa de desarrollo de bajo costo con un chip ESP32-S, cámara OV2640, varios GPIOs para conectar periféricos y cuenta con una ranura para tarjetas microSD para almacenar imágenes tomadas con la cámara.

Ilustración 4: ESP32-CAM
Fuente: [16]

Está diseñado para aplicaciones de muy bajo consumo, aplicaciones móviles, de electrónica portátil y de Internet de las Cosas (IoT). Como el trabajo de Martín Osuna [17], que diseña y programa un sistema de bajo coste para el control de un brazo robótico, con el objetivo de almacenar y clasificar piezas, utilizando la cámara del ESP32-CAM programada mediante la biblioteca openCV en lenguaje Python para procesar la imagen. Algunas otras aplicaciones se detallan a continuación:

- Cámaras para la transmisión de video.
- Reconocimiento de imágenes.
- Agricultura inteligente
    - Invernaderos inteligentes

  - Riego inteligente
  - Robótica agrícola
- Electrónica portátil
  - Relojes inteligentes
  - Pulseras inteligentes
- Registradores de datos IoT genéricos de bajo consumo
- Hub genérico de sensores IoT de bajo consumo [18]

1.2.2.1. Características

La placa integra WiFi, Bluetooth tradicional y Bluetooth Low Energy o Bluetooth Smart (BLE), con dos CPU LX6 de 32 bits de alto rendimiento. Adopta una arquitectura de tubería de 7 fases, sensor en chip, sensor de pasillo, sensor de temperatura, etc. Su frecuencia de sintonización principal oscila entre 80 MHz y 240 MHz.

Totalmente compatible con los estándares WiFi 802.11b/g/n/e/i y Bluetooth 4.2, se puede usar como modo maestro para crear un controlador de red independiente o como esclavo de otras MCU anfitrionas para agregar funcionalidad de red a los dispositivos existentes.[19]

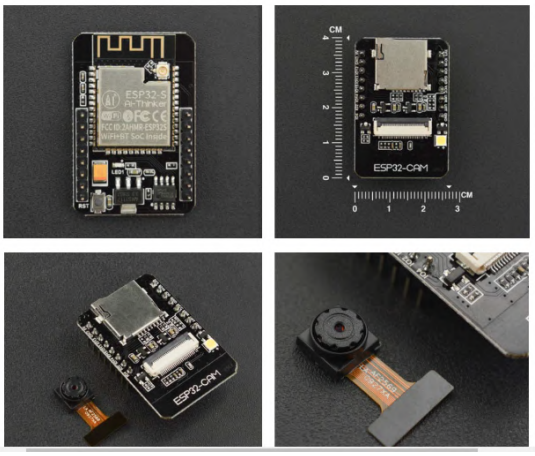


Ilustración 5: ESP32-CAM con cámara OV2640
Fuente:[19]

1.2.2.2. Especificaciones

- Voltaje de alimentación: 5VDC
- Voltaje entradas/salidas (GPIO): 3.3VDC
- SoM: ESP-32S (Ai-Thinker)
- SoC: ESP32 (ESP32-D0WDQ6)
- CPU: Dual core Tensilica Xtensa LX6 (32 bit)
- WiFi 802.11b/g/n, Bluetooth 4.2
- Antena PCB, también disponible conexión a antena externa

- 520KB SRAM interna, 4MB SRAM externa
- Soporta UART/SPI/I2C/PWM/ADC/DAC
- Incluye socket para TF card micro-SD
- Cámara OV2640
- Resolución fotos: 1600x1200 pixeles
- Resolución vídeo: 1080p30, 720p60 y 640x480p90
- Incluye LED de flash en placa
- Óptica de 1/4 pulgada
- Dimensiones: 27x40.5x6 mm
- Peso: 20 gramos

1.2.2.3. Diagrama Esquemático

El siguiente gráfico muestra el diagrama esquemático del ESP32-CAM. Donde se puede observar cada uno de los componentes del microcontrolador detallados anteriormente; los elementos más destacados serían el módulo ESP-32S, las conexiones de la cámara que trae por defecto (OV2640), led y/o flash integrado, PSRAM y el Socket Micro SD.

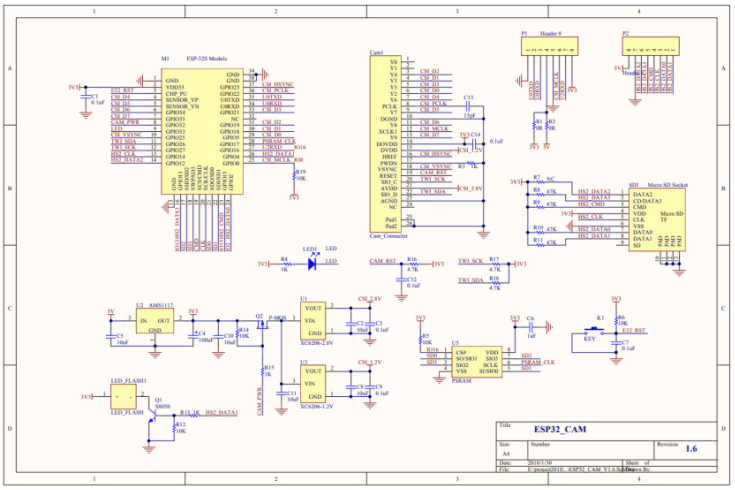


Ilustración 6: Diagrama Esquemático ESP32-CAM
Fuente: [20]

1.2.2.4. Diagrama de pines

En la ilustración a continuación se detalla los diferentes pines que posee el ESP32-CAM.

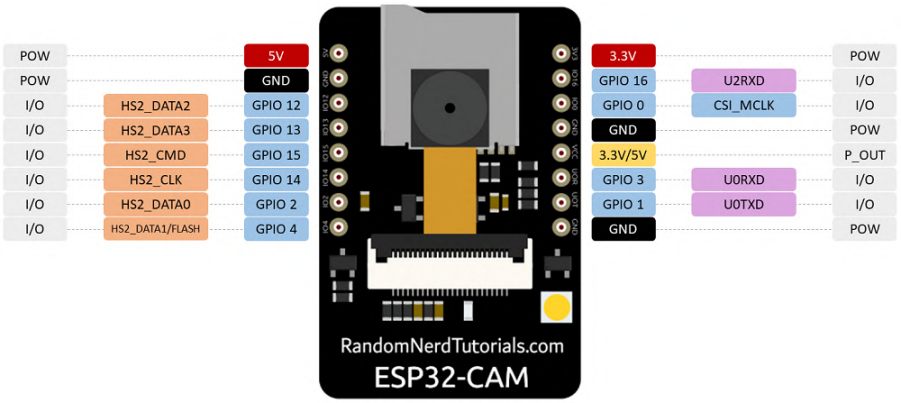


Ilustración 7: Diagrama de pines ESP32-CAM

Fuente: [21]

1.2.2.4.1. Descripción de pines

Como se puede visualizar en la Ilustración 7, el ESP32-CAM cuenta con tres pines GNDs (color negro) y dos pines de alimentación: 3,3 V y 5V (color rojo).

También se encuentra el pin VCC (color amarillo) que es un pin de potencia de salida, que puede emitir 5V o 3.3V, en este caso por defecto la ESP32-CAM emitirá 3.3V, si se desea tener una salida de 5V, se debe desoldar la conexión y soldar las de 5V.

Ilustración 8: Almohadillas de conexión 3.3V o 5V
Fuente: [22]

Los pines de serie TX y RX, son GPIO1 y GPIO3 respectivamente, la ESP32-CAM no dispone de programador incorporado, debe usar para comunicarse con la placa y cargar el código, después de eso puede utilizar para conectar otros periféricos, pero no podrá visualizar Serial Monitor y comprobar que la configuración sea exitosa.

El GPIO0 establece si el ESP32-CAM se encuentra en modo pasivo o no. Si GPIO0 está conectado a GND puede cargar el código a la placa. Para que el microcontrolador funcione necesita desconectar GPIO0 de GND.

La ESP32-CAM tiene un LED incorporado muy brillante que se puede usar como flash durante la captura de foto y/o grabación de video. Este LED está conectado internamente a GPIO 4. [22]

1.2.3. Arduino IDE

Según Claudio Peña [23] El entorno de desarrollo integrado de Arduino, o IDE, es un software multiplataforma que se puede utilizar para programar y subir programas en Arduino y placas compatibles. Pero no solo eso, se puede utilizar para cargar programas en placas de otros fabricantes, gracias a kernel generados por terceros.

Un modelo de utilización de diferentes placas es la que evidencia Tomás Domínguez en el manual para desarrollar aplicaciones IoT en Arduino y ESP8266 junto con el Asistente Virtual de Amazon: Alexa, empleando el entorno de Arduino IDE.[24]

1.2.4. OPENCV

De acuerdo con Hernández, López y Castellanos, OpenCV (Open Source Computer Visión) es una librería software open-source de visión artificial y Machine Learning. Proporcionada por Intel [25] .

OpenCV como biblioteca libre brinda varias ventajas en el procesamiento de imágenes, y podemos usar sus herramientas para identificar objetos y aplicarlos a la visión artificial al diseñar objetos de investigación.

Como mencionan los autores [26] ,del prototipo diseñado para elaborar un modelado de robot de un sistema automático de alto nivel capaz de detectar y rastrear objetos en movimiento.

De acuerdo con los autores del artículo [27] , la librería OpenCV se emplea de varias formas, para la detección de la lengua de signos, la cual detecta los gestos de las manos, equivale al alfabeto inglés, ayudando a las personas a comunicarse utilizando la red neuronal convolucional y a la vez clasificando las imágenes por cada gesto de la mano y traducirlos con la ayuda de la librería OpenCV.

La librería OpenCV contiene más de 2500 algoritmos, incluidos algoritmos de aprendizaje automático y visión por computadora, listos para usar. Estos algoritmos le permiten identificar objetos, rostros, clasificar el comportamiento humano en videos, rastrear movimientos de objetos, obtener modelos 3D, encontrar imágenes similares, eliminar ojos rojos, rastrear movimientos oculares, reconocer escenas.

1.2.5. YOLOv3

Los autores Cortez, Giraldo y Vergara, detallan que YOLOv3 es una CNN que detecta y segmenta 80 clases de objetos distintos en una caja [28] . Aunque este modelo es lo suficientemente robusto, se eligió como la única clase para la detección de "pez" porque este objeto tiene características geométricas simples para que la CNN YoloV3 pueda aprender más fácilmente y porque este objeto es ampliamente utilizado.

YOLO es un modelo de alto rendimiento muy popular para la detección de objetos y se considera la tecnología más avanzada para la detección en tiempo real (FPS).

Según el artículo [29] plantea un mejoramiento de Tiny YOLOv3 con una detección alta de objetos utilizando K-means clustering para evaluar el tamaño de las cajas para la agrupación de datos, mejorando la velocidad y precisión para la detección de objetos.

1.2.5.1. Modelo Mobilenet

Una Arquitectura esquematizada que usa capas de redes convolucionales para formar redes neuronales profundas de un peso mínimo y conceder un modelo eficaz para aplicaciones de visión móviles e integradas.

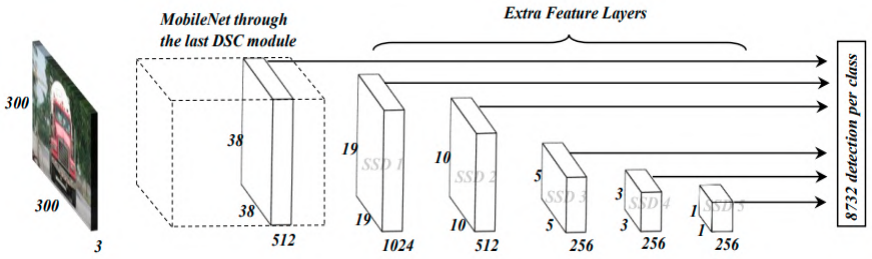


Ilustración 9: Diagrama de Modelo Mobilenet
Fuente: [30]

Mobilenet es un modelo muy ligero que implementa eficientemente redes neuronales convolucionales.[31] MobileNet usa convolución para producir una funcionalidad de nivel como cualquier otra red básica, reduciendo así la cantidad de parámetros de red.

Se puede usar este modelo en diferentes áreas o fines para mejoramiento o detección, como lo mencionan los autores en el artículo [32] , un mejoramiento

para el modelo MobileNet-SSD para el reconocimiento automatizado de fallas en la pintura del chasis de un vehículo, detalla el fin de mejorar y aumentar la precisión de las fallas de pintura en los vehículos, proponen usar el modelo MobileNet-SSD y optimizarlos, da como resultados superiores del 95% de precisión en si mejoran un 10% del modelo mencionado.

1.2.6. Modulo L298N

El controlador de motor L298N está configurado para controlar dos motores de corriente continua de forma bidireccional [33] .

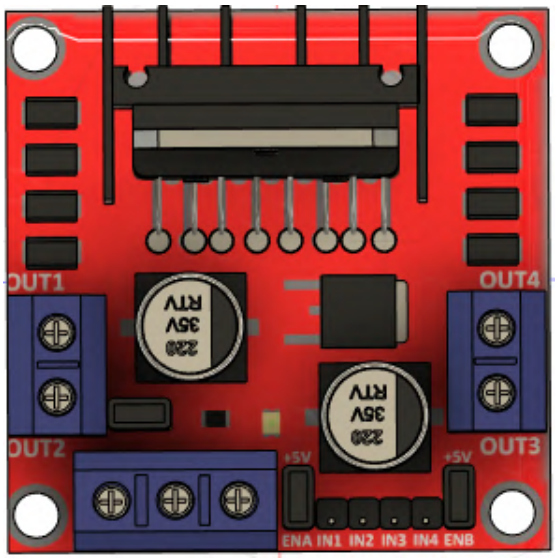


Ilustración 10: Modulo Controlador de Motor L298N
Fuente: Elaboración Propia

Este controlador de motor bidireccional se basa en el muy popular circuito integrado de controlador de motor de doble puente H L298. Este circuito le permite controlar de manera fácil e independiente dos motores con una corriente máxima de 2A para control bidireccional. Es ideal para aplicaciones robóticas y es ideal para conectarse a microcontroladores que requieren solo unos pocos cables de control para impulsar el motor. También puede interactuar con interruptores manuales simples, puertas lógicas TTL, relés y más. La tarjeta está equipada con un indicador LED de alimentación, un regulador de 5V incorporado y diodos de protección [34] .

En la Figura siguiente se encuentra el diagrama del circuito interno del módulo L298N Motor Driver:

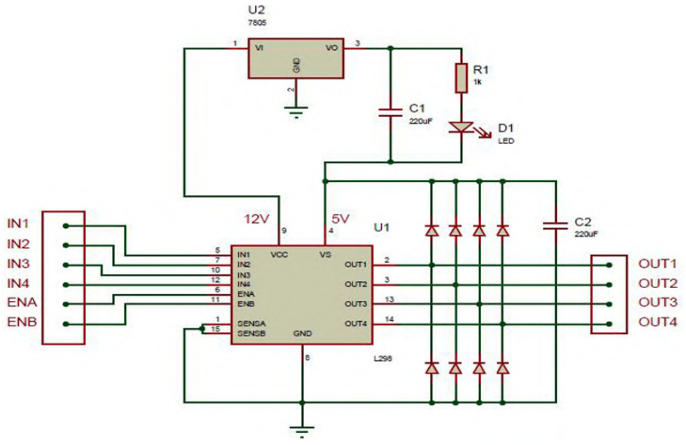


Ilustración 11: Diagrama del circuito interno del L298N
Fuente: [35]

Configuración de los pines del módulo L298N:

PIN	DESCRIPCIÓN
IN1 & IN2	Pines de entrada del motor A, controla el sentido de giro del motor A.
IN3 & IN4	Pines de entrada del motor B, controlan el sentido de giro del motor B.
ENA	Habilita la señal PWM para el motor A.
ENB	Habilita la señal PWM para el motor B.
OUT1 & OUT2	Pines de salida del motor A.
OUT3 & OUT4	Pines de salida del motor B
12V	Voltaje de entrada desde una Fuente de alimentación.
5V	Suministra energía para el circuito lógico de conmutación dentro del CI L298N
GND	Pin a Tierra

Tabla 1: Configuración de Pines del L298N
Fuente: Elaboración propia

1.3. OBJETIVOS DEL PROTOTIPO.

1.3.1. Objetivo General

Desarrollar un prototipo de dispensación de alimentos automático para tilapias mediante el uso del microcomponente ESP32-CAM.

1.3.2. Objetivos Específicos

- Construir el prototipo utilizando componentes electrónicos.

- Desarrollar el código necesario para la implementación del comedero de alimentos.
- Implementar el prototipo de dispensación de alimentos automático para tilapias en una quinta.

1.4. DISEÑO DEL PROTOTIPO.

1.4.1. Componentes Electrónicos

Para la elaboración del prototipo se utilizaron los siguientes componentes electrónicos que detallamos en la Tabla 1.

Tabla 2: Componentes electrónicos utilizados en el prototipo

DISPOSITIVOS Y COMPONENTES	
Microcontrolador	ESP32-CAM
Programador	FTDI PROGRAMMER FT232
Conector	USB mini a USB
Motor	FD RD550 20000 RPM
Modulo para motor	L298N
Cables de Conexión	Cables puente 22AWG - cobre estañado hembra
Fuente de energía	Batería de 12V
Interruptores	Interruptores ojos de cangrejo de 3 tiempos, 6 pines y 3 pines.
Placa solar	Panel Solar POWOXI 12V, 1.7W

Fuente: Elaboración propia

La innovación tecnológica se evidencia gracias a las múltiples opciones de desarrollo para un mismo caso, argumentando similitudes, en el trabajo [36], se presenta una propuesta similar, pero usando otro tipo de componentes electrónicos y totalmente automático sin vista de inteligencia artificial, con intervalos programados, de esta manera se puede tomar como base para futuras aplicaciones. Sin embargo, en el cuidado de tilapias se aplican en muchas variables como son el pH del entorno don se encuentra, si el agua esta cristalina o turbia, usando diferentes sensores se logran medir y dar una métrica para resolver los problemas antes mencionados. [37]

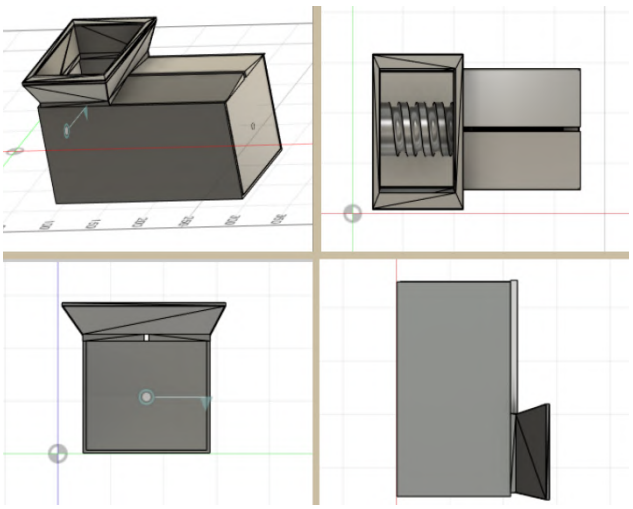


Ilustración 12: Modelo en varias vistas
Fuente: Elaboración propia

Como se puede visualizar en la Ilustración 12 se elaboró el posible diseño del dispensador de alimentos, se buscaba un diseño minimalista donde no sea tan incómodo a la hora de su traslado pero que aun así no pierda fuerza para el desplazamiento de la comida dentro de la tolva. Para la potencia se buscó un motor adecuado para dicho trabajo, que no sufra atascos al momento de la distribución del alimento.

1.5. Alimentación

La tilapia, es una especie muy popular para la acuicultura tropical [38] . Las poblaciones naturales de estas especies de peces se encuentran en África y se han introducido en casi todos los países tropicales con fines acuícolas.

La tilapia necesita esencialmente los ácidos grasos ácido linoleico y ácido araquidónico, que se encuentran en los aceites vegetales [39] . Los lípidos pueden mejorar las tasas de conversión alimenticia como una fuente económica y de alta energía.

Las dietas de las tilapias pueden diferir de las de las demás especies de peces, las enzimas de su estómago tienen un factor de mayor adaptación para generar más proteasa, amilasa que se usa en la digestión de materia vegetal, la cual es más complicada de digerir que la materia animal. Por lo tanto, se puede usar cualquiera y/o ambas dietas. [40]

También Fortes da Silva confirma que las tilapias tienen la capacidad de auto componer una dieta con distintas metodologías: comederos a demanda y el encapsulamiento de macronutrientes. Cabe destacar que auto seleccionan la dieta equilibrada en aminoácidos esenciales y su auto suplementación. [41]

1.5.1. Periodo de alimentación

Los periodos de alimentación según menciona Sousa [42] tiene que ver mucho con el comportamiento de la tilapia y esto evoluciona a una correcta administración para que el pez se desarrolle normalmente en sus hormonas y dando como resultado una buena salud en el ejemplar.

Según Nicovita [43] , menciona que la alimentación en un solo sitio es una forma menos eficaz para alimentar, la razón es que es muy complicado dispersar el alimento a los demás peces. Existen alimentadores automatizados, como tipo péndulo, temporizado, tipo bandeja, etc. Sin embargo, debido a sus altos costos, son sistemas antieconómicos y solo se utilizan en fincas, quintas que superan la relación costo-beneficio.

Como lo detalla Ornelas Luna entre otros [44], alimentar con una sola ración diaria, produce que el alimento perdure más tiempo en el tracto digestivo de la tilapia y que la digestibilidad se vea afectada. Es por eso que se recomienda que en ejemplares de 40g se alimente al menos de 3 a 4 raciones diarias en conjunto de estudiar su desempeño alimenticio.

Aumentar la frecuencia de alimentación complementaria durante el día tiene varias ventajas, especialmente si los alimentos naturales no constituyen una parte significativa de la ingesta diaria.

Estos beneficios son:

- Una reducción de pérdidas de alimentos
- Reducción del consumo de oxígeno disuelto y una mejora del nivel de la calidad del agua;
- Reducción de pérdida de nutrientes, mejorando así la calidad de los alimentos;
- Uniformidad mejorada del tamaño de los peces, para peces menos agresivos una mayor oportunidad de alimentarse;
- Mejor tasa de crecimiento de los peces y mejor utilización del alimento. [45]

Como lo determina FAO [45], antes de decidir con qué frecuencia se debe alimentar a los peces, se debe tomar nota de los puntos siguientes:

- Cuanto más pequeño es el pez, más a menudo se necesita alimentar con frecuencia.

- La comida seca debe administrarse con más frecuencia que la comida húmeda.
- No se debe alimentar más del 3 por ciento del peso total del pez a la vez.
- La frecuencia de alimentación debe reducirse a medida que desciende la temperatura del agua o si supera el nivel óptimo.
- La frecuencia debe adaptarse a la especie de pez.
- A la tilapia le va mejor con comidas pequeñas y frecuentes.
- Se debe verificar el costo del alimento para asegurar que no sea excesivo en relación con los rendimientos obtenidos

Según foros de piscicultura [46], las tablas de alimentación varían según el sistema, datos, estudios, etc. Por consiguiente, los productores de tilapias no siguen al pie de la letra dichas indicaciones, sin embargo, para un mayor control se recomienda tener lo siguiente:

Tabla de porcentajes de alimentación

En la siguiente Tabla 4 se detalla el uso de balanceado recomendado por PRONACA, con bases a las características, temperatura y condiciones de agua normales.

Tabla 3: Uso Recomendado de balanceado PRONACA.

Tipo de Alimento	Proteína %	Peso corporal Tilapia (g)	Tamaño Partícula (m.m+0.5)	Rango. (días)	Tasa alimenticia %Biomasa	Dosis Recomendada. (día)
Tilapia juvenil 1	35	5 a 10	2,2	31 a 50	8	6
Tilapia juvenil 2	32	11 a 60	2,2	51 a 100	6	6
Tilapia Engorde 1	32	61 a 150	2,8	101 a 140	4	4
Tilapia Engorde 2	30	151 a 250	3,5	141 a 180	2,5	3 a 4
Tilapia Engorde 3	28	251 a 350	6	181 a 220	1,5	3
Tilapia Engorde 4	24	350 a 550	6	221 a 275	1,5	3
Tilapia Reproductor	40	150 a 1000	2,8/3,75 y 6	>100	4	3

Fuente: [47]

1.6. EJECUCIÓN Y/O ENSAMBLAJE DEL PROTOTIPO.

1.6.1. Configuración básica del ESP32-CAM en Arduino IDE

En esta parte del documento redactamos las etapas del desarrollo del prototipo, utilizando el ESP32-CAM, FTDI PROGRAMMER FT232.

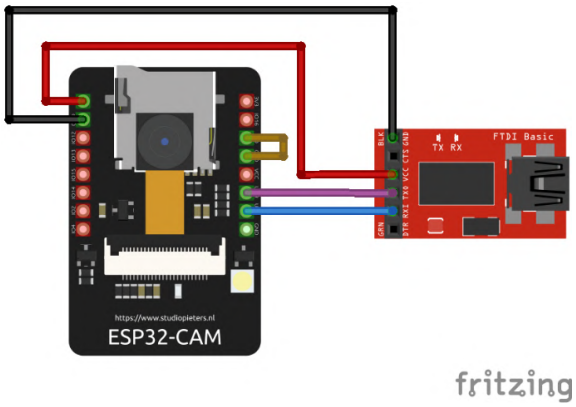


Ilustración 13: Conexión pasiva del ESP32-CAM
Fuente: Elaboración propia

El ensamblado del prototipo comienza con la instalación del complemento ESP32 en Arduino IDE, es recomendable colocar la dirección de url en "Gestor de URLs Adicionales de Tarjetas"

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

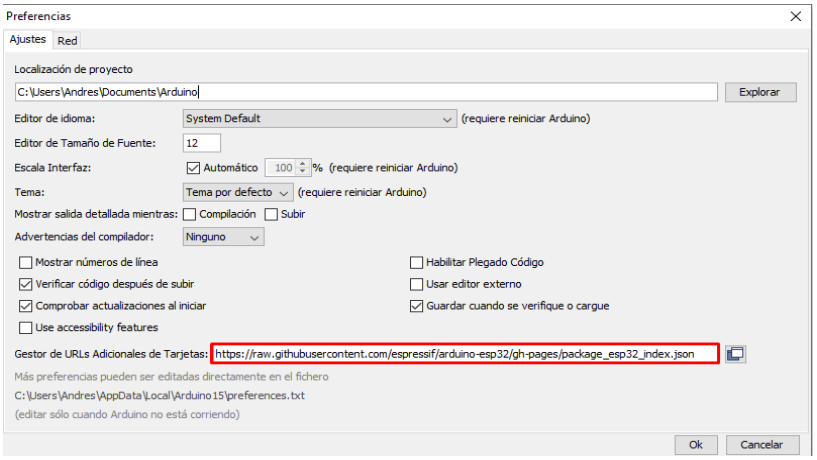


Ilustración 14: Preferencias en Arduino IDE
Fuente: Elaboración propia

A continuación, se configura el "Gestor de Tarjetas". En la barra de menú en ***Herramientas > Placa > Gestor de Tarjetas***

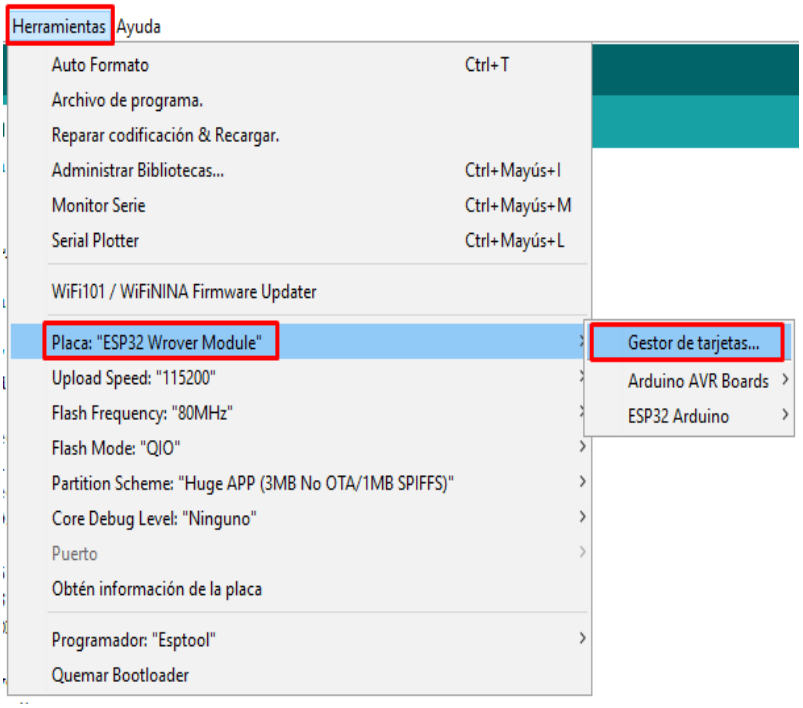


Ilustración 15: Menú para configurar el gestor de tarjetas en Arduino IDE
Fuente: Elaboración Propia

Se realiza una búsqueda del **ESP32** y presione el botón de instalar "***ESP32 by Espressif Systems***":

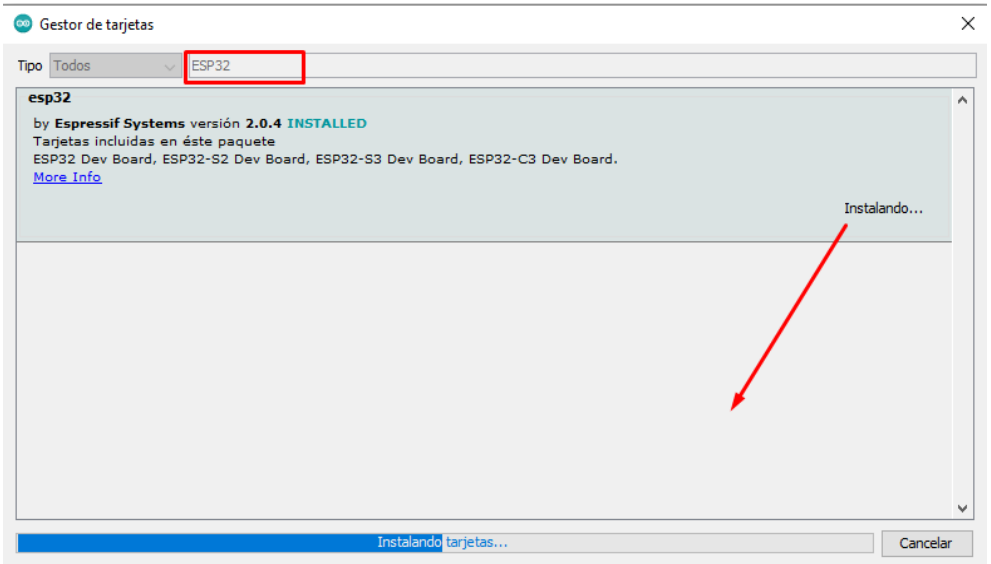


Ilustración 16: Gestor de Tarjetas
Fuente: Elaboración propia

1.6.2. Configuración y Ejecución de Código

Se conecta la placa del ESP32-CAM a la computadora. En el Arduino IDE se coloca la siguiente configuración de placa:

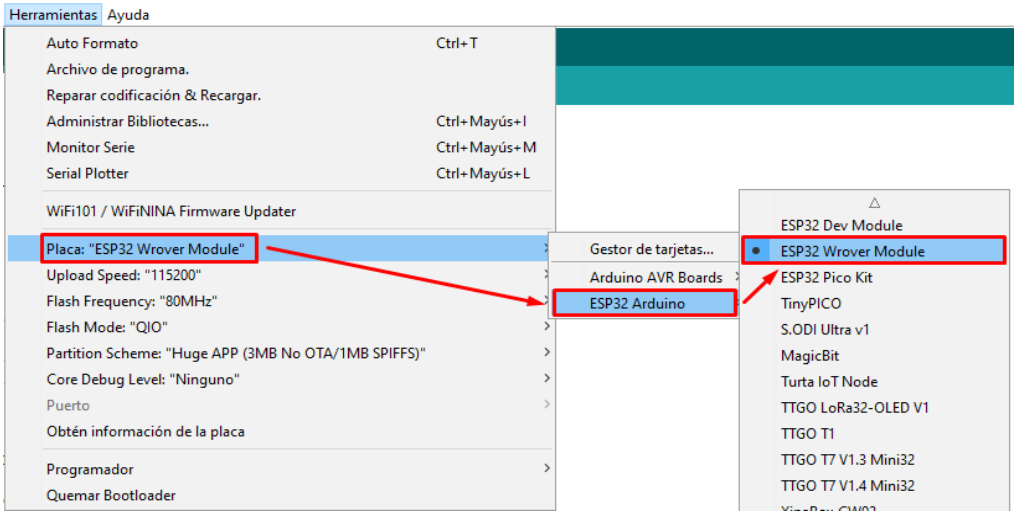


Ilustración 17: Selección de placa ESP32 Wrover Module
Fuente: Elaboración propia

Se selecciona la placa "***ESP32 Wrover Module***" velocidad de subida, Frecuencia, Modo Flash, Esquema de partición y el puerto "***COM***":

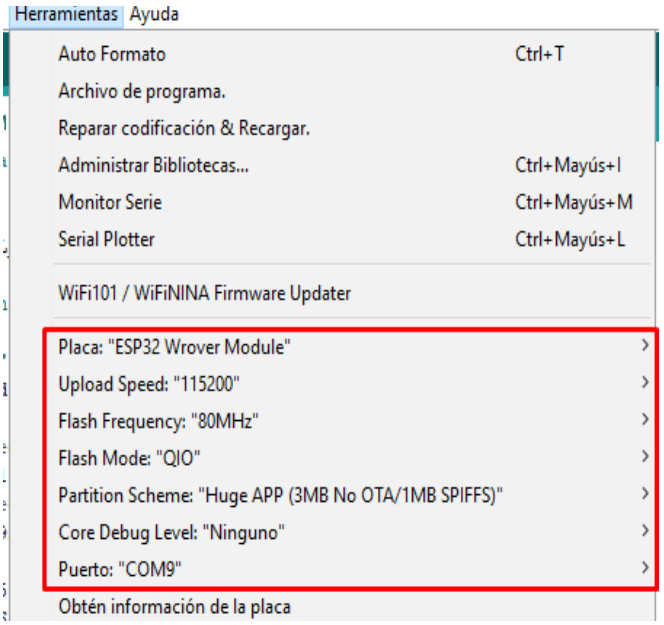


Ilustración 18: Configuración necesaria para cargar código
Fuente: Elaboración propia

Una vez configurada la placa en Arduino IDE, se carga el código, y se procede a subir a la placa ESP32-CAM. Espere unos segundos mientras el código se compila y carga en la placa.

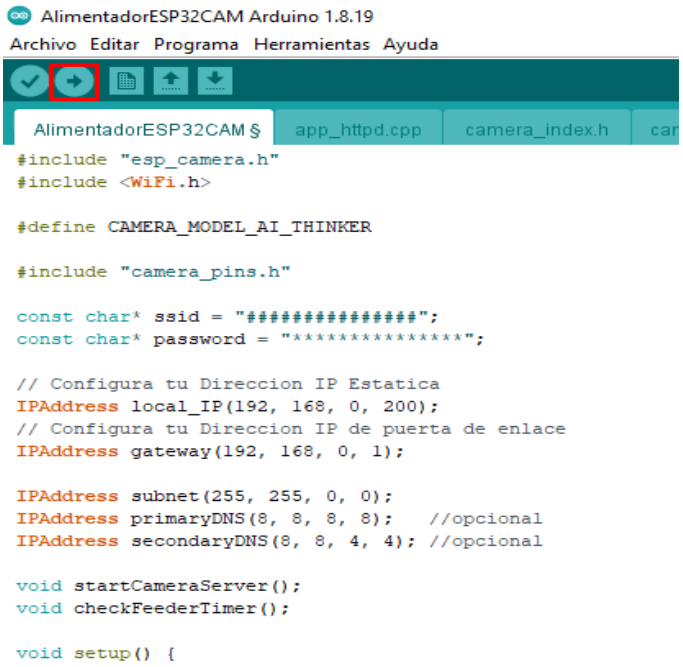


Ilustración 19: Subir código al ESP32-CAM
Fuente: Elaboración propia

Como punto final, si terminó de cargar saldrá un mensaje como se detalla en la figura siguiente:

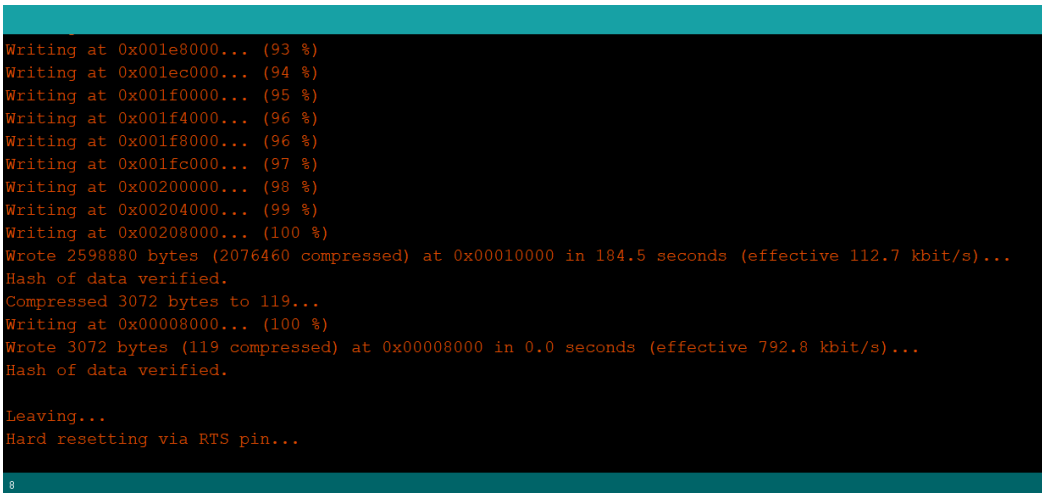


Ilustración 20: Proceso de subida de código en Arduino IDE
Fuente: Elaboración propia

Al final de la subida del código dice: ***"Hard resetting via RTS pin"*** por consiguiente es necesario quitar del ESP32-CAM el puente que se hizo de GND con GPIO0 para poder acceder a Monitor Serial de Arduino IDE, y una vez realizado, se coloca la velocidad de transmisión de 115200 baudios, para que al momento de presionar el reset de la placa se obtiene una dirección IP automáticamente.

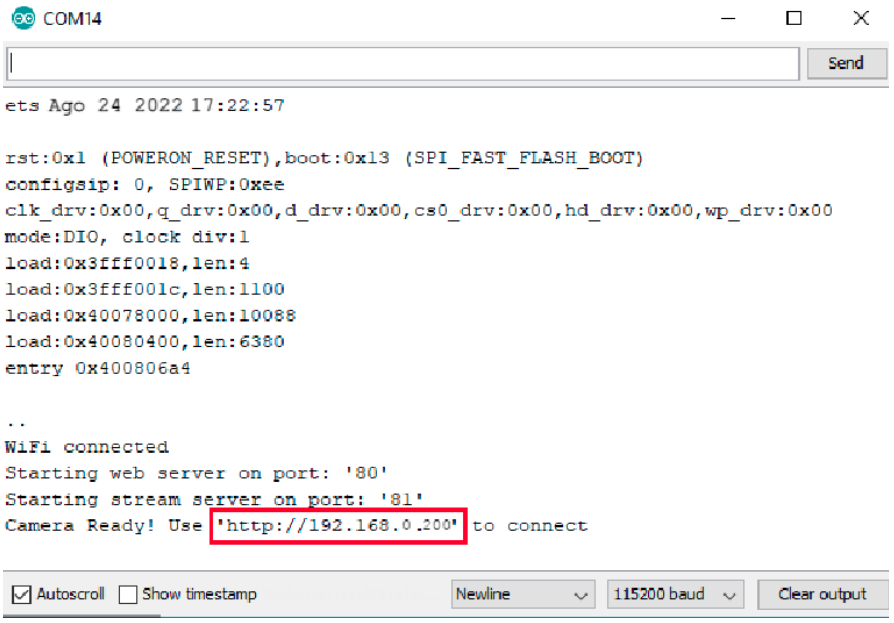


Ilustración 21: Monitor Serie
Fuente: Elaboración propia

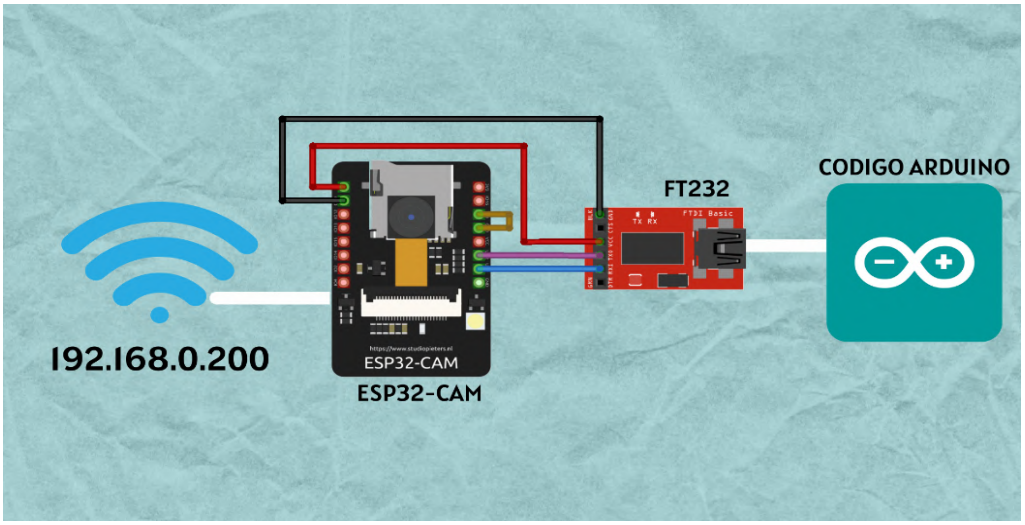


Ilustración 22: Diagrama Estructural
Fuente: Elaboración propia

1.6.3. Descripción de Código

1.6.3.1. Arduino IDE

En el desarrollo del prototipo se utiliza el lenguaje de programación C, es el lenguaje utilizado para desarrollar UNIX, Linux y cientos de otros sistemas informáticos y software.

Todo el archivo con la extensión de Arduino IDE .ino contiene cuatro pestañas que se detallan en la sección de Anexos, en este apartado se desarrollarán ciertas funciones y/o procedimientos.

```
const char* ssid = "***************";
const char* password = "*******************";

// Configura tu Direccion IP Estatica
IPAddress local_IP(192, 168, 1, 184);
// Configura tu Direccion IP de puerta de enlace
IPAddress gateway(192, 168, 1, 1);

IPAddress subnet(255, 255, 0, 0);
IPAddress primaryDNS(8, 8, 8, 8);   //opcional
IPAddress secondaryDNS(8, 8, 4, 4); //opcional
```

Ilustración 23: Configuración de red WiFi y Dirección IP estática
Fuente: Elaboración Propia

Las variables *const char** ssid y password se declaran el nombre de la red y la contraseña del mismo. Mientras que *IPAddress* declara una dirección IP dentro de la red local.

```
  if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("STA Failed to configure");
  }

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Conexión con WiFi exitosa");

  startCameraServer();

  Serial.print("¡Camara Lista! Por favor, use el siguiente link 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' para conectarse");
}
```

Ilustración 24: Configuración de red correcta
Fuente: Elaboración Propia

En la ilustración se configura que, si no existe una conexión correcta con la red WiFi, no se podrá acceder. Pero si el status de la red fue exitoso, el servidor de la cámara empezará, y cede la dirección IP estática antes configurada.

En la siguiente pestaña del Anexo 31, el procedimiento de la Ilustración 25 es para detallar el dutty cycle o ciclo de trabajo para el motor en la programación. El dutty cycle no es más que la relación entre el tiempo de encendido frente al tiempo apagado

```
void feedingAnalogWrite(uint8_t channel, uint32_t value, uint32_t valueMax = 180) {
  // calculate duty, 8191 from 2 ^ 13 - 1
  uint32_t duty = (8191 / valueMax) * min(value, valueMax);
  ledcWrite(channel, duty);
}
```

Ilustración 25: Procedimiento para declarar el ciclo de trabajo
Fuente: Elaboración Propia

Para que ese motor funcione sin trabas y que no haya sobrecarga, se utilizará un módulo L298N para que regule el voltaje que necesita el motor de 12 voltios y que se conviertan en 5v para que alimente al microcontrolador. Ballesteros comenta: La modulación de ancho de pulsos PWM implica generar pulsos que varían en su período útil en función del valor de la señal de entrada[48]. Incluso Marulanda y otros realizaron un estudio experimental de construcción de un PWM ajustables, que pueda realizar cambios en tiempo real, entre esos cambios serían frecuencias y ciclos de trabajo. [49]

El ESP32-CAM tiene PWM, que en pocas palabras es el que ayudará para salidas análogas. Para poder utilizar el motor con el módulo L298N en cuestión debe ser por una de las 16 entradas de PWM que proporciona el ESP32-CAM. Se utiliza en inversores DC/AC monofásicos y trifásicos. Se basan en la comparación de la señal de referencia modulada con una señal portadora triangular o en diente de sierra. [50]

```
pinMode(feederPin, OUTPUT);

// Feeder servo PWM
ledcSetup(4, 50, 16);         // channel, freq, resolution
ledcAttachPin(feederPin, 4);  // pin, channel
```

Ilustración 26: Indicaciones de PWM
Fuente: Elaboración Propia

En la variable ledcSetup se configuran el canal, la frecuencia y la resolución, donde en este caso el canal será el mismo pin de la linterna (4), con una frecuencia (50) que es la cantidad de pulsos por segundos de la señal y una resolución (16).

En ledcAttachPin se declara el pin para la alimentación (feederPin) que es el GPIO12, con el respectivo canal (4).

```
void runFeeder() {
  Serial.print("Start feeding the fish for ");
  Serial.print(feedingtime);
  Serial.println(" seconds");

  feedingAnalogWrite(4, 750);
  delay(feedingtime * 1000);
  feedingAnalogWrite(4, 0);

  Serial.println("Feeding the fish finished");
}
```

Ilustración 27: Procedimiento para correr alimentador

Fuente: Elaboración Propia

Procedimiento para ejecutar el alimentador, donde feedingAnalogWrite muestra el canal (4) y la cantidad de a la que va a trabajar el ciclo de trabajo el motor al encenderse (750) y al apagarse (0) después de ejecutar el programa.

En la siguiente pestaña, en el anexo 36 se muestra una codificación de lenguaje HTML a hexadecimal, mediante la página de CyberChef [51] es un conversor de distintos lenguajes a sistemas y se debe realizar este paso antes, para que Arduino pueda reconocer y ejecutar el código sin problemas.

En la última pestaña, no es más que los pines de las distintas cámaras que puede soportar el microcontrolador ESP32-CAM, las cuáles se encuentran en el Anexo 55.

1.6.3.2. Python

Durante el desarrollo del prototipo se utiliza el lenguaje Python, permite implementar varias bibliotecas disponibles y es compatible con muchos paquetes y módulos.

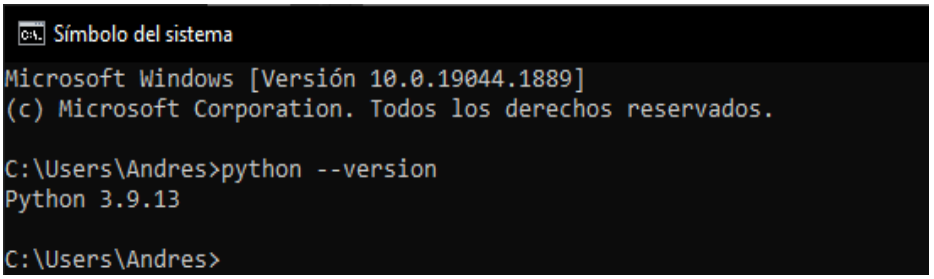


Ilustración 28: Version de Python para la IA

Fuente: Elaboración propia

En la ilustración se observa la versión de Python instalada en el ordenador, es importante configurar con anterioridad correctamente el componente ESP32-CAM para su correcto funcionamiento, para que no cause conflicto con las versiones y dirección de IP.

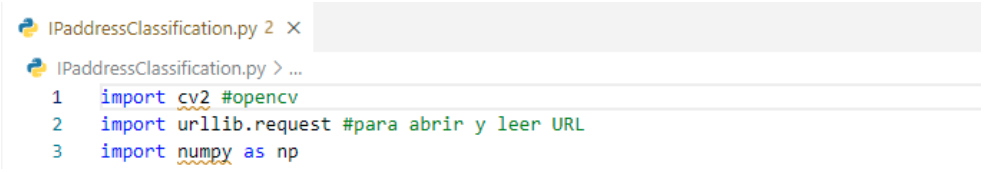


Ilustración 29: Librerías de Python utilizadas.

Fuente: Elaboración propia

Para la Ejecución del programa de Detección de Tilapias, se utiliza librerías muy conocidas como: cv2 (OpenCV), urllib, numpy, también se emplea un algoritmo de reconocimiento de imágenes como lo es YOLOv3. Cada una cumple una función importante en el software como los cálculos de la librería de numpy, permite trabajar directamente con los datos y valores.

```
7   url = 'http://192.168.1.184/capture'
8   winName = 'ESP32 CAMERA'
9   cv2.namedWindow(winName,cv2.WINDOW_AUTOSIZE)
```

Ilustración 30: Configuración de IP y Ventana.
Fuente: Elaboración propia

Para utilizar la Inteligencia Artificial, primero se configura la dirección IP, el nombre de la ventana, como se consigue la dirección ip, esto se detalla en la sección de 6.2.3.1 Arduino IDE.

```
20   net = cv2.dnn_DetectionModel(weightsPath,configPath)
21   net.setInputSize(320,320)
22   #net.setInputSize(480,480)
23   net.setInputScale(1.0/127.5)
24   net.setInputMean((127.5, 127.5, 127.5))
25   net.setInputSwapRB(True)
```

Ilustración 31: Creación del Modelo de Detección.
Fuente: Elaboración propia

En este punto se configura o se crea la red de Modelo de Detección con pesos y configuraciones entrenadas. Se establece el tamaño de la ventana o Frame, la escala, valores medios y banderas o avisos que se genera cuando se cambia de canal.

```
27  while(1):
28      imgResponse = urllib.request.urlopen (url) #abrimos el URL
29      imgNp = np.array(bytearray(imgResponse.read()),dtype=np.uint8)
30      img = cv2.imdecode (imgNp,-1) #decodificamos
31
32      img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE) # vertical
33
34
35      classIds, confs, bbox = net.detect(img,confThreshold=0.5)
36      print(classIds,bbox)
37
38      if len(classIds) != 0:
39          for classId, confidence,box in zip(classIds.flatten(),confs.flatten(),bbox):
40              cv2.rectangle(img,box,color=(0,255,0),thickness = 3) #mostramos en rectangulo lo que se encuentra
41
42      cv2.imshow(winName,img) # mostramos la imagen
43
44      #esperamos a que se presione ESC para terminar el programa
45      tecla = cv2.waitKey(5) & 0xFF
46      if tecla == 27:
47          break
48  cv2.destroyAllWindows()
```

Ilustración 32: Ciclo repetitivo de detección.
Fuente: Elaboración propia

Durante el ciclo repetitivo While (Mientras), se abre la dirección ip del ESP32-CAM, se decodifica la imagen de captura, con una rotación vertical y mientras se decodifica la imagen, se ejecuta unos rectángulos o cajas de colores detectando el objeto o en este caso es la tilapia y al final en caso de que se requiera cerrar la detección solo se presiona la tecla ESC del teclado para terminar la Ejecución de la Inteligencia Artificial.

1.6.4. Ensamblaje del Prototipo

El ensamblaje del prototipo empieza con el modelado de la tolva, es importante conseguir un modelo de contenedor o transportador que satisfagan las necesidades del proyecto, en este caso se utilizara un contenedor de un dispensador con un motor FD RS550.

En la Ilustración 28 se visualiza un transportador con un tornillo sin fin o también conocido tornillo helicoidal, es un contenedor de gran talla, normalmente siempre va en una base, se usa como transportador de vaciado o colector, puede transportar productos de grano tanto fino como grueso y está elaborada de diferentes materiales en este caso es de metal.

Ilustración 33: Transportador de tornillo sin fin.
Fuente: Elaboración propia

El mecanismo de tornillo sin fin, es muy importante para transportar grandes cantidades de material o de alimentos, [52] ayuda a reducir el esfuerzo físico o la manipulación inadecuada de alimentos.

El moldeado del transportador fue muy importante al momento de implementarlo en las pruebas de campo real, en este caso se tomó las medidas necesarias para la comodidad y la portabilidad, la cual se realizó un boceto con las medidas previas a la fase final del prototipo.

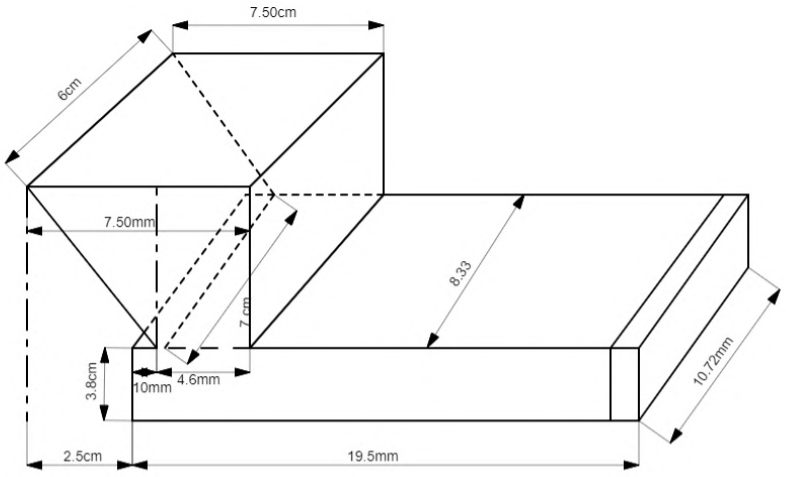


Ilustración 34: Medidas previas del modelo del dispensador.

Fuente: Elaboración propia

Ilustración 35: Proceso de Moldeado del Transportador.

Fuente: Elaboración propia

La adaptación del transportador es un proceso extenso como lo indica en las siguientes Ilustraciones:

Ilustración 36: Proceso de cortes del transportador.

Fuente: Elaboración propia

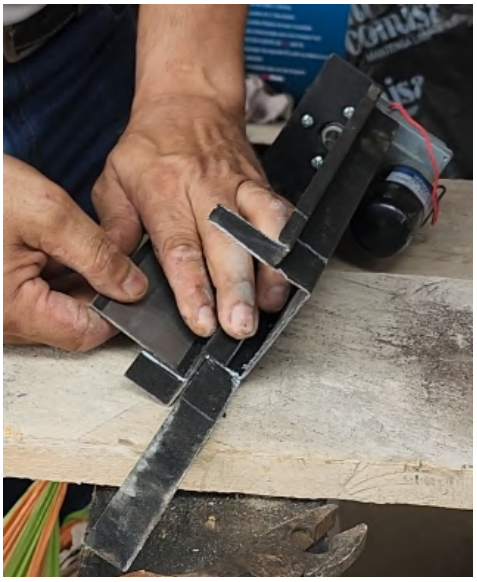


Ilustración 37: Proceso de Doblaje
Fuente: Elaboración propia

En la siguiente ilustración, se puede observar cómo se hace el proceso de limado o lijado de bordes y acabados.

Ilustración 38: Proceso de Lijado

Fuente: Elaboración propia

El tornillo sin fin, transmite un movimiento entre los ejes que se encuentran en un ángulo recto o perpendiculares, este movimiento permite el traslado del alimento de los alevines, en la ilustración 26 se puede ver cómo es el modelo de tornillo sin fin que se emplea en este prototipo:

Ilustración 39: Modelo del Tornillo sin fin
Fuente: Elaboración propia

En la siguiente ilustración 35, se manejó acrílico para diseñar el moldeado de la tolva, aplicando calor sobre la superficie que se desea doblar:

Ilustración 40: Proceso de diseño de la tolva con acrílico
Fuente: Elaboración propia

El siguiente paso es el proceso final de armar, consiste en unir todas las piezas y juntarlas para darle la forma correcta de un dispensador como se demuestra en la Ilustración 41, 42:

Ilustración 41: Proceso de Ensamblaje final.
Fuente: Elaboración propia.

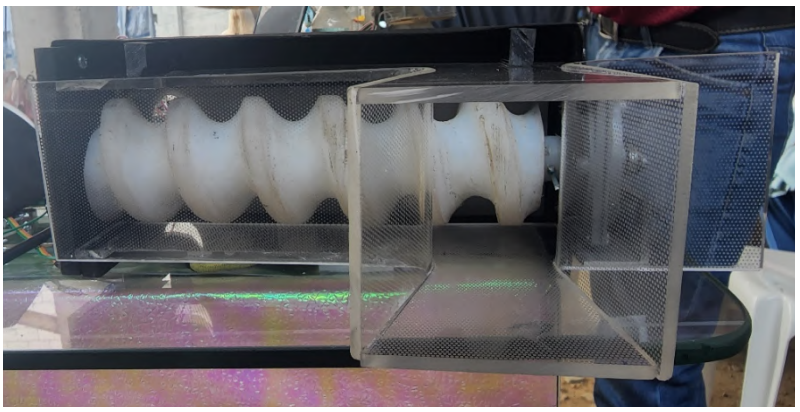


Ilustración 42: Proceso de Ensamblaje Final Vista Superior.
Fuente: Elaboración propia.

A continuación, se agregó el soporte o pilar para una mayor fijación del prototipo a una superficie ya sea en forma vertical:

Ilustración 43: Soporte de dispensador.
Fuente: Elaboración propia

En la Ilustración 44, se puede observar las pruebas previas antes de colocar el respectivo panel solar al dispensador, las pruebas realizadas son para verificar el estado del panel solar y comprobar si está en el voltaje adecuado.

Ilustración 44: Pruebas del panel solar.
Fuente: Elaboración propia

Ilustración 45: Pruebas de Voltaje al Panel Solar.
Fuente: Elaboración propia.

Para culminar el proceso de Ensamblado del prototipo, se fabricó una cajita para poder contener los componentes electrónicos, como los módulos, los cables de conexión, los switches, entre otros, con detalle en la siguiente Ilustración 46, 47:

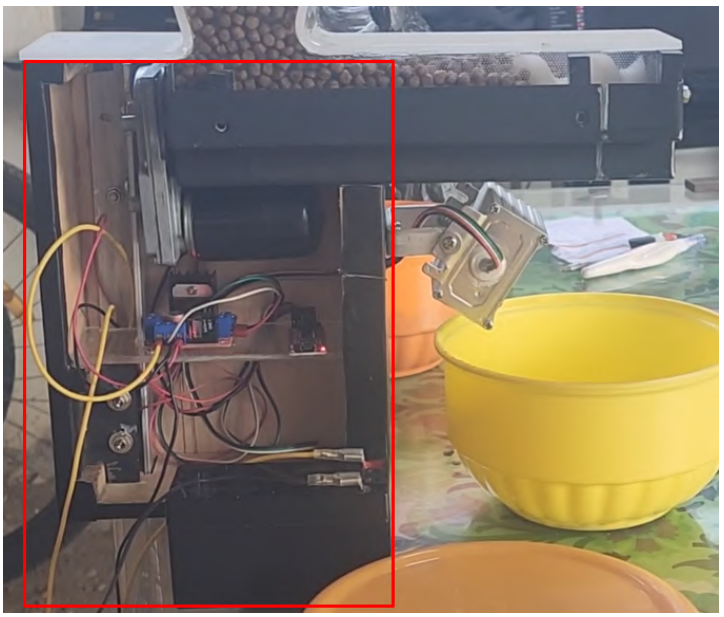


Ilustración 46: Caja contenedora de los componentes electrónicos.

Fuente: Elaboración propia.

Ilustración 47: Caja contendora de componentes electrónicos.

Fuente: Elaboración propia

La Ilustración 48 e Ilustración 49 se puede observar las pruebas de funcionalidad del prototipo final y ejecutando todos los componentes por completo.

Ilustración 48: Pruebas y Ensamble Final.

Fuente: Elaboración propia.

Ilustración 49: Ensamblaje terminado y funcional.

Fuente: Elaboración propia

CAPÍTULO III. EVALUACIÓN DEL PROTOTIPO.

2.1. PLAN DE EVALUACIÓN.

El plan de evaluación se realiza comparando los valores de la cantidad del alimento para tilapia con el respectivo tiempo programado desde el dispensador. Se realizaron pruebas de cuanta es la cantidad que se dispensa por segundos.

2.2. RESULTADOS DE LA EVALUACIÓN.

Tabla 4: Cantidad de alimento dispensado con tolva llena

Tiempo (sg)	Alimento (gramos)	Tiempo (sg)	Alimento (gramos)
2	11	**16**	85
3	16	**17**	91
4	21	**18**	96
5	27	**19**	101
6	32	**20**	107
7	37	**21**	112
8	43	**22**	117
9	48	**23**	123
10	53	**24**	128
11	59	**25**	133
12	64	**26**	139
13	69	**27**	144
14	75	**28**	149
15	80	**29**	155
		30	160

Fuente: Elaboración propia

En el proceso de automatización del dispensador, se optimiza la cantidad de alimento, dado que mientras la tolva se mantenga llena y esté completamente distribuido en el comedero, descarga 160 gramos durante los 30 segundos de funcionamiento, según se puede observar en la Tabla 4.

Considerando que el cultivo de tilapia, desde su siembra hasta la cosecha, tiene un tiempo aproximado de 36 semanas [43], se elaboró la tabla 5, basada en el número de peces que fueron usados para la siembra desde la semana cero (0), dato que fue proporcionado por el dueño de la quinta.

Hasta el momento de la elaboración de este trabajo de titulación, el cultivo se ubica en la semana 23 de la tabla de alimentación (Tabla 5), es decir, el dispensador automático tiene que dispensar 156 gramos aproximadamente, 4 veces durante el día.

Tabla 5: Tabla de alimentación para tilapias en base a NICOVITA - VITAPRO

Semana	Número/Peces	pp/gr	% Tasa Alimentación	Biomasa/gr	Alimento diario(gr)	Numero de Raciones /día	Gramos de alimento/ración
0	200	1	15	200	30	8	4
1	196	3	10	589	59	8	7
2	193	5	8	965	77	8	10
3	188	7	5,8	1318	76	8	10
4	183	10	5,7	1826	104	8	13
5	178	13	5,5	2308	127	8	16
6	172	17	5,1	2922	149	8	19
7	167	22	5,1	3672	187	8	23
8	162	29	5	4702	235	8	29
9	158	37	4,5	5832	262	8	33
10	154	46	4,3	7090	305	8	38
11	150	56	4,2	8376	352	6	59
12	144	69	4,1	9924	407	6	68
13	138	83	4	11434	457	6	76
14	131	100	4	13120	525	6	87
15	127	120	3,5	15280	535	6	89
16	122	140	3,4	17102	581	6	97
17	118	162	3,2	19124	612	6	102
18	112	184	2,9	20614	598	6	100
19	107	207	2,8	22152	620	4	155
20	102	231	2,6	23595	613	4	153
21	97	256	2,4	24904	598	4	149
22	92	282	2,3	25891	595	4	149
23	92	309	2,2	28370	624	4	156
24	92	337	2,1	30941	650	4	162
25	92	355	1,9	32594	619	4	155
26	92	393	1,8	36083	649	4	162
27	92	422	1,7	38745	659	4	165
28	92	451	1,6	41408	663	2	331
29	92	480	1,5	44070	661	2	331
30	92	509	1,4	46733	654	2	327
31	92	538	1,4	49395	692	2	346
32	92	567	1,4	52058	729	2	364
33	92	596	1,3	54721	711	2	356
34	92	629	1,3	57750	751	2	375
35	92	654	1,2	60046	721	2	360
36	92	683	1,1	62708	690	2	345

Fuente: Elaboración propia

A continuación, se indican los resultados obtenidos en las pruebas de funcionamiento del prototipo (Tabla 6), realizada durante una semana, donde se observa la fecha, horarios de activación del dispensador, cantidad de alimento dispensada y observaciones.

Tabla 6: Pruebas de funcionamiento del prototipo

Fecha	H1	H2	H3	H4	Cantidad dispensada	Observaciones
	6:00	12:00	18:00	0:00		
08/08/2022	160 g	94 g			**254 g**	Inicia prueba, con tolva llena y solo abastece para la primera hora de la prueba y la segunda sólo proporcionó 94 g
09/08/2022	160 g	160 g			**320 g**	Inicia prueba, con tolva llena y solo abastece las dos primeras horas teniendo en cuenta de volver a llenar la tolva
10/08/2022	160 g	160 g			**320 g**	Inicia prueba, con tolva llena y solo abastece las dos primeras horas teniendo en cuenta de volver a llenar la tolva, mientras se realiza complemento para aumentar capacidad
11/08/2022	160 g	160 g			**320 g**	Inicia prueba, con tolva llena y solo abastece las tres primeras horas teniendo en cuenta de volver a llenar la tolva, mientras se realiza complemento para aumentar capacidad
12/08/2022	160 g	160 g	160 g		**480 g**	Inicia prueba, con tolva llena y solo abastece las tres primeras horas teniendo en cuenta de volver a llenar la tolva, mientras se realiza complemento para aumentar capacidad
13/08/2022	160 g	160 g	160 g	160 g	**640 g**	Inicia prueba, con tolva llena mientras se prueba complemento para aumentar capacidad
14/08/2022	160 g	160 g	160 g	160 g	**640 g**	Funciona sin problemas y con el complemento

Fuente: Elaboración propia

Se tuvo algunos inconvenientes en las pruebas de funcionamiento del prototipo, durante los primeros días de evaluación, producto de contar con una tolva pequeña. Pero esto se solucionó colocando un complemento, que aumenta la capacidad del dispensador (Ver Anexo 7).

CONCLUSIONES

- Se construyó el prototipo utilizando componentes electrónicos, especialmente el microcontrolador ESP32-CAM.
- Se desarrolló el código necesario para la implementación del comedero de alimentos, donde se pueda alimentar al momento y/o haya un intervalo de alimentación por determinadas horas.
- Se Implementó el prototipo de dispensación de alimentos automático para tilapias en una quinta, ubicada en el Sitio Río Chico, parroquia Bella María del cantón Santa Rosa.
- Durante la implementación del prototipo, se observó que el microcontrolador ESP32-CAM tiene problemas de conexión cuando se aleja del router WiFi.
- Actualmente en el mercado extranjero existe un microcontrolador que ya contiene el conector para realizar la conexión pasiva sin necesidad de un programador externo, se lo puede encontrar como ESP32-CAM CH340.

RECOMENDACIONES

- Para evitar problemas de conexión de internet se recomienda la compra de un ESP32-CAM con el conector IPEX integrado.
- En caso de que el dispensador automático se aleje a más de nueve (9) metros del router WiFi, se recomienda un extensor o repetidor de señal que se conecte a un tomacorriente externo.
- Para una mejor dispersión del alimento o materia orgánica, se recomienda que se instalen 2 dispensadores adicionales, en ciertos puntos estratégicos para que el alimento tenga más alcance a toda la población de tilapia.
- Se recomienda tener un mejor control de los datos importantes en la cría de las tilapias desde el proceso de siembra hasta su cultivo, para que en próximos proyectos exista una correcta exactitud.
- Para tener mejores resultados se recomienda alimentar in situ a las tilapias, de tal manera se obtiene un monitoreo adecuado.
- Se recomienda que el modo de “Habilitar intervalo de alimentación” sea utilizado en casos especiales, si fuera el caso de que el encargado se tenga que ausentar por más de un día.
- Este prototipo queda como base para futuros proyectos tecnológicos enfocados en la acuicultura, por lo que se recomienda no dejarlo en el abandono ya que con un enfoque más determinado y detallado puede llegar a ser incluso mejor como “Aquisys” en Brasil, que apoya las buenas prácticas en la tilapicultura. [53]

BIBLIOGRAFÍA

[1] F. Neira, “Análisis comparativo entre un monocultivo de camarón) Litopenaeus vannamei), y un policultivo de camarón blanco (Litopenaeus vannamei) y tilapia roja (Oreochromissp).”, UNIVERSIDAD ESTATAL PENINSULA DE SANTA ELENA, La Libertad, 2022.

[2] L. M. Ayala, V. Asesor, y M. E. Valencia, “New technologies and use of technological tools in fish production with emphasis on biofloc systems and pond raceway systems”.

[3] J. Capelo, “Análisis de producción y económico de Camarón(Penaeus vannamei)y Tilapia (Oreochromis spp.)en Ecuador”, 2021.

[4] U. Nacional *et al.*, “Tilapia en Ecuador: paradoja entre la producción acuícola y la protección de la biodiversidad ecuatoriana Correspondencia: * Autor de correspondencia”, *Comentarios Revista peruana de biología*, vol. 26, núm. 4, pp. 543–550, 2019, doi: 10.15381/rpb.v26i4.16343.

[5] R. Chaganti, V. Varadarajan, V. S. Gorantla, T. R. Gadekallu, y V. Ravi, “Blockchain-Based Cloud-Enabled Security Monitoring Using Internet of Things in Smart Agriculture”, *Future Internet 2022, Vol. 14, Page 250*, vol. 14, núm. 9, p. 250, ago. 2022, doi: 10.3390/FI14090250.

[6] C. R. B. dos Santos y E. P. C. Borges, “Sistema de monitoramento de baixo custo para galpões avícolas de pequeno porte utilizando IoT”, *ForScience*, vol. 10, núm. 1, p. e01116, ago. 2022, doi: 10.29069/forscience.2022v10n1.e1116.

[7] P. M. Baltazar, “La Tilapia en el Perú: acuicultura, mercado, y perspectivas”, *Revista Avances de las ciencias biológicas en el Perú*, vol. 13, núm. 3, pp. 267–273, 2007, [En línea]. Available: http://sisbib.unmsm.edu.pe/BVRevistas/biologia/biologiaNEW.htm

[8] I. Ecuador y M. Gallino, “Facultad de Ingeniería Marítima y Ciencias del Mar Cultivo de Tilapia en Ecuador”, 2008.

[9] Ministerio de Agricultura y Ganadería, “El MAG también apunta a los cultivos no tradicionales como la tilapia – Ministerio de Agricultura y Ganadería”, ago. 27, 2018. https://www.agricultura.gob.ec/el-mag-tambien-apunta-a-los-cultivos-no-tradicionales-como-la-tilapia/ (consultado ago. 27, 2022).

[10] E. B. A. Luzón, A. R. C. Alava, E. P. Carpio, y V. J. G. Montealegre, “Gestión de agronegocios de la tilapia roja (Oreochromis Spp. O) y su comercialización”, *Revista*

Metropolitana de Ciencias Aplicadas, vol. 4, núm. 2, pp. 58–67, may 2021, Consultado: sep. 05, 2022. [En línea]. Available: https://remca.umet.edu.ec/index.php/REMCA/article/view/377

[11] C. E. Romero y A. Elustondo, “Análisis de la capacidad de la placa ESP32 para integrar sistemas IoT descentralizados”, *Elektron*, vol. 6, núm. 1, pp. 41–45, may 2022, doi: 10.37537/rev.elektron.6.1.142.2022.

[12] S. Santos y R. Santos, “ESP32-CAM Projects”.

[13] Espressif Systems, “ESP32-S2 Family”, 2021. [En línea]. Available: https://www.espressif.com/en/support/download/documents.

[14] “Placa de desarrollo ESP32, WiFi + Bluetooth, consumo de energía ultrabajo, doble núcleo, ESP 32, ESP 32S, ESP, 32 ESP32 CAM, ESP WROOM 32|Circuitos integrados| - AliExpress”. https://es.aliexpress.com/item/1005001954197937.html?gatewayAdapt=glo2esp (consultado sep. 02, 2022).

[15] “ESP-32S Datasheet”, 2016, Consultado: sep. 02, 2022. [En línea]. Available: http://www.ai-thinker.com

[16] Handson Technology, “Handson Technology Datasheet ESP32-CAM WiFi+Bluetooth+Camera Module”. [En línea]. Available: https://handsontec.com

[17] J. J. Martín Osuna, “Diseño y Programación de un Sistema de Bajo Coste basado en la ESP 32-CAM para el Control de un Brazo Robótico”, jul. 2022, Consultado: ago. 22, 2022. [En línea]. Available: https://riunet.upv.es:443/handle/10251/184354

[18] Espressif Systems, “ESP-32S Datasheet”, 2016. [En línea]. Available: http://www.ai-thinker.com

[19] DFROBOT, “ESP32-CAM Development Board”.

[20] “Diagrama Esquemático ESP32-CAM”. https://github.com/SeeedDocument/forum_doc/blob/master/reg/ESP32_CAM_V1.6.pdf (consultado ago. 22, 2022).

[21] “ESP32-CAM-pinout-new.png (1000×445)”. https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2020/03/ESP32-CAM-pinout-new.png?quality=100&strip=all&ssl=1 (consultado ago. 28, 2022).

[22] “ESP32-CAM AI-Thinker Pinout Guide: GPIOs Usage Explained | Random Nerd Tutorials”. https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/ (consultado ago. 28, 2022).

[23] C. Peña, *Arduino IDE: Domina la programación y controla la placa*. Consultado: ago. 28, 2022. [En línea]. Available: https://books.google.es/books?hl=es&lr=&id=Xgv2DwAAQBAJ&oi=fnd&pg=PP1&dq=arduino+IDE&ots=vNDTJeWw0U&sig=4CNNxbwwYCtrggZ0kr-5lex9_lQ#v=onepage&q=arduino%20IDE&f=false

[24] T. Domínguez Mínguez, “Alexa: desarrollo de aplicaciones IoT para Arduino y ESP8266”, Consultado: ago. 28, 2022. [En línea]. Available: https://dialnet.unirioja.es/servlet/autor?codigo=5533772

[25] C. Villareal Hernández, H. L. Chimil, L. Tonatiuh, C. Serrano, y E. L. Franco, “Dron polinizador de cultivos. Tecnologías aplicadas para alternativas sustentables”.

[26] C. G. de Souza Silva, Y. S. Padua, y S. C. Felipussi, “LoCAR–Low-Cost Autonomous Robot for Object Detection with Voice Command and MobileNets”, *Applied Artificial Intelligence*, vol. 34, núm. 11, pp. 816–831, sep. 2020, doi: 10.1080/08839514.2020.1782004/FORMAT/EPUB.

[27] S. Purkayastha, M. J. Roy, y N. Pradhan, “A demonstration for the detection of sign language”, *https://doi.org/10.1080/02522667.2019.1704519*, vol. 40, núm. 8, pp. 1611–1621, nov. 2020, doi: 10.1080/02522667.2019.1704519.

[28] J. F. Cortes Zarta, Y. A. Giraldo Tique, y C. F. Vergara Ramírez, “Red neuronal convolucional para la percepción espacial del robot InMoov a través de visión estereoscópica como tecnología de asistencia”, *Enfoque UTE*, vol. 12, núm. 4, pp. 88–104, 2021, [En línea]. Available: https://www.redalyc.org/articulo.oa?id=572268461006

[29] W. Gai, Y. Liu, J. Zhang, y G. Jing, “An improved Tiny YOLOv3 for real-time object detection”, *Systems Science and Control Engineering*, vol. 9, núm. 1, pp. 314–321, 2021, doi: 10.1080/21642583.2021.1901156/FORMAT/EPUB.

[30] S. Arabi, A. Haghighat, y A. Sharma, “A deep learning based solution for construction equipment detection: from development to deployment”, 2019, Consultado: ago. 28, 2022. [En línea]. Available: https://www.researchgate.net/publication/332553794_A_deep_learning_based_solution_for_construction_equipment_detection_from_development_to_deployment

[31] C. De y I. de Sistemas, “FACULTAD DE INGENIERÍA CIVIL”.

[32] J. Zhang *et al.*, “An improved MobileNet-SSD algorithm for automatic defect detection on vehicle body paint”, *Multimedia Tools and Applications 2020 79:31*, vol. 79, núm. 31, pp. 23367–23385, jun. 2020, doi: 10.1007/S11042-020-09152-6.

[33] D. N. T. How, M. Z. Baharuddin, S. S. K. Mohideen, K. S. M. Sahari, y A. Anuar, “Modular motor driver with torque control for gripping mechanism”, en *Procedia Engineering*, 2012, vol. 41, pp. 1476–1482. doi: 10.1016/j.proeng.2012.07.338.

[34] Component Datasheet, “Handson Technology User Guide L298N Dual H-Bridge Motor Driver”. [En línea]. Available: www.handsontec.com

[35] “L298N Motor Driver Module Pinout, Datasheet, Features & Specs”. https://components101.com/modules/l293n-motor-driver-module (consultado ago. 29, 2022).

[36] M. D. Shahiran y S. Salimin, “Smart Fish Feeder Using Solar Energy”, *Journal of Electronic Voltage and Application*, vol. 2, núm. 2, pp. 92–101, dic. 2021, doi: 10.30880/jeva.2021.02.02.010.

[37] I. I. Mohd, N. Hikmah, B. Azizan, N. Elfadil, y M. Pahang, “Design and Development of Microcontroller Based Automatic Fish Feeder System”, *Ijesc*, vol. 10, núm. 4, pp. 25380–25383, 2020, [En línea]. Available: http://ijesc.org/

[38] M. C. Makeche *et al.*, “ Characterisation of Oreochromis niloticus fish species of Lake Kariba, Zambia, using morphological, meristic and genetic methods ”, *Aquaculture, Fish and Fisheries*, vol. 2, núm. 2, pp. 116–129, abr. 2022, doi: 10.1002/AFF2.36.

[39] D. M. Torres -Novoa y V. L. Hurtado -Nery, “Requerimientos nutricionales para Tilapia del Nilo (Oreochromis niloticus) Requerimientos nutricionales para Tilapia del Nilo (Oreochromis niloticus) Nile tilapia (Oreochromis niloticus) nutritional requirements Exigências nutricionais para tilápia do Nilo (Oreochromis niloticus)”.

[40] S. N. Hlophe, N. A. G. Moyo, y I. Ncube, “Postprandial changes in pH and enzyme activity from the stomach and intestines of Tilapia rendalli (Boulenger, 1897), Oreochromis mossambicus (Peters, 1852) and Clarias gariepinus (Burchell, 1822)”, *Journal of Applied Ichthyology*, vol. 30, núm. 1, pp. 35–41, feb. 2014, doi: 10.1111/JAI.12290.

[41] R. Fortes da Silva, “Estudio comparado de los ritmos de alimentación y selección dietaria en dos peces teleósteos: lubina y tilapia del nilo”, 2010, Consultado: sep. 03, 2022. [En línea].

Available:
https://dialnet.unirioja.es/servlet/tesis?codigo=216703&info=resumen&idioma=SPA

[42] R. M. R. Sousa, C. A. Agostinho, F. A. Oliveira, D. Argentim, P. K. Novelli, y S. M. M. Agostinho, “Productive performance of Nile tilapia (Oreochromis niloticus) fed at different frequencies and periods with automatic dispenser”, *Arq Bras Med Vet Zootec*, vol. 64, núm. 1, pp. 192–197, feb. 2012, doi: 10.1590/S0102-09352012000100027.

[43] Nicovita, “Manual de CRIANZA de Tilapia”, Consultado: sep. 05, 2022. [En línea]. Available: http://www.industriaacuicola.com/biblioteca/Tilapia/Manual%20de%20crianza%20de%20tilapia.pdf

[44] R. Ornelas-Luna, B. Aguilar-Palomino, A. Hernández-Diaz, J. Á. Hinojosa-Larios, y D. E. Godínez-Siordia, “Vista de Un enfoque sustentable al cultivo de tilapia”, *Acta Universitaria Multidisciplinary Scientific Journal*, pp. 19–25, 2017, doi: 10.15174/au.2017.1231.

[45] “NUTRICION Y ALIMENTACION DE LOS PECES”. https://www.fao.org/fishery/docs/CDrom/FAO_Training/FAO_Training/General/x6709s/x6709s10.htm (consultado sep. 05, 2022).

[46]. : “PisciculturaGlobal :.” https://www.pisciculturaglobal.com/alimentar-peces-utilizando-tablas-de/ (consultado sep. 07, 2022).

[47] C. de INGENIERÍA EN ADMINISTRACIÓN Y PRODUCCIÓN AGROPECUARIA AUTOR Wilson Napoleón Carrera Gallo y E. Cantón Macará, “ÁREA AGROPECUARIA Y DE RECURSOS NATURALES RENOVABLES EVALUACIÓN PRODUCTIVA Y ECONÓMICA DE LA CRIANZA DE DOS VARIEDADES DE TILAPIA ROJA EN”.

[48] D. M. BALLESTEROS, “MODULACIÓN PWM EN FPGA BASADO EN MÁQUINAS DE ESTADO FINITO”, *Scientia Et Technica*, vol. XIV, núm. 38, pp. 421–426, 2008, Consultado: sep. 06, 2022. [En línea]. Available: https://www.redalyc.org/articulo.oa?id=84903874

[49] J. James, M. Durango, J. Jairo Ordoñez, L. Fernando, y M. Machado, “Diseño y construcción de un convertidor dc/dc tipo Boost con PWM ajustable”, *Scientia Et Technica*, vol. 22, núm. 1, pp. 9–14, 2017, Consultado: sep. 06, 2022. [En línea]. Available: https://www.redalyc.org/articulo.oa?id=84953102002

[50] D. Murillo-Yarce, J. J. Marulanda-Durango, A. Escobar-Mejía, D. Murillo-Yarce, J. J. Marulanda-Durango, y A. Escobar-Mejía, “Estudio comparativo de técnicas PWM de banda

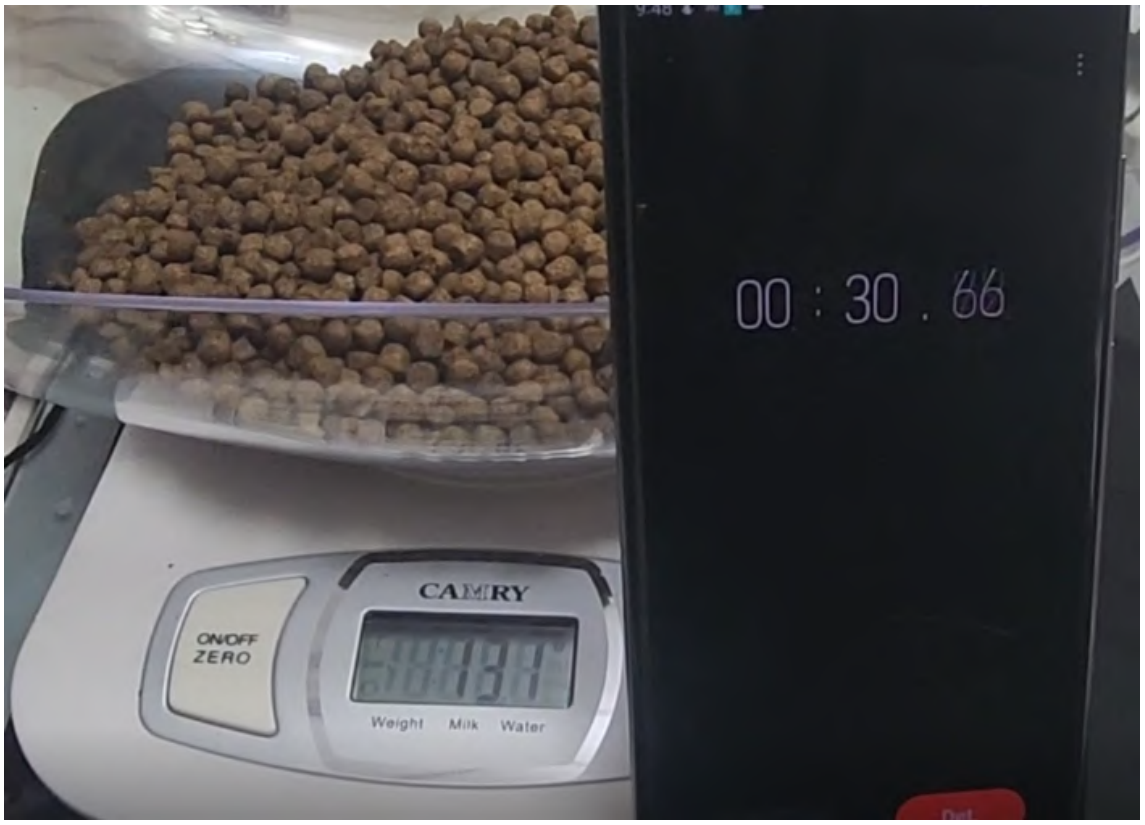
de histéresis para el control de corriente en filtros activos de potencia paralelos", *TecnoLógicas*, vol. 21, núm. 43, pp. 91–106, sep. 2018, doi: 10.22430/22565337.1058.

[51] "CyberChef". https://gchq.github.io/CyberChef/ (consultado sep. 06, 2022).

[52] M. H. Freire Quintanilla, "Sistema móvil de transportación de materia prima mediante el mecanismo de tornillo sin fin en la planta de balanceados de la estación experimental Tunshi – Espoch", 2017, Consultado: ago. 29, 2022. [En línea]. Available: http://dspace.unach.edu.ec/handle/51000/3570

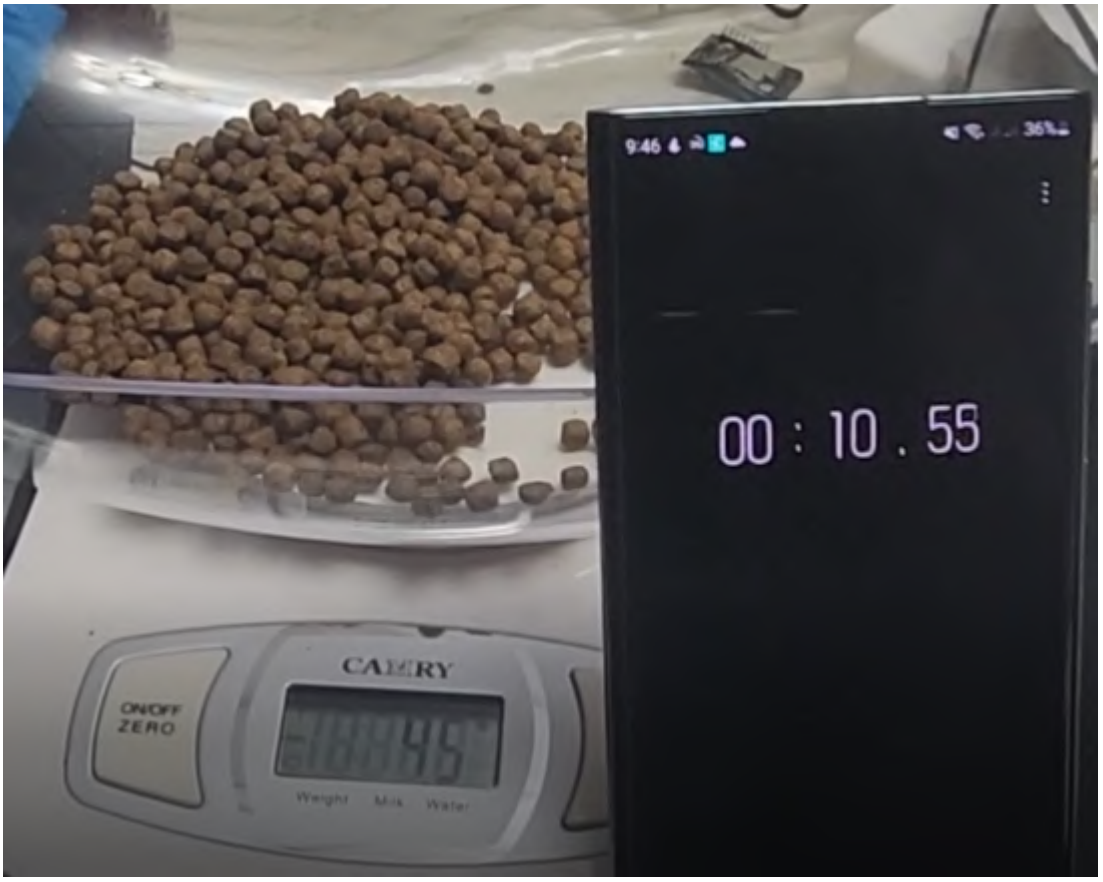
[53] M. C. Peres Young Pessoa, A. F. Rodrigues Seixas, y Marcos Eliseu Losekann, "Sistema computacional para apoyar las buenas prácticas de gestión de la acuicultura en Brasil (AQUISYS) – énfasis en la tilapicultura", *Con la acuicultura cuidamos tu salud*, pp. 714–715, 2009, doi: 10.13140/RG.2.1.1235.7847.

ANEXOS



Anexo 1: Cantidad de comida en proceso manual

Anexo 2: Tiempo de expulsión de comida del dispensador



Anexo 3: Tiempo de alimentación manual

Anexo 4: Cantidad de comida expulsada por dispensador a los 10s

Anexo 5: Cantidad de comida expulsada por dispensador a los 30s

Anexo 6: Proceso de instalación de dispensador

Anexo 7: Instalación y prueba del prototipo

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
#include "esp_camera.h"
#include <WiFi.h>

#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

const char* ssid = "NETLIFE-ROMERO";
const char* password = "RomeroOchoa2013";

// Configura tu Direccion IP Estatica
IPAddress local_IP(192, 168, 1, 184);
// Configura tu Direccion IP de puerta de enlace
IPAddress gateway(192, 168, 1, 1);

IPAddress subnet(255, 255, 0, 0);
IPAddress primaryDNS(8, 8, 8, 8);   //opcional
IPAddress secondaryDNS(8, 8, 4, 4); //opcional

void startCameraServer();
void checkFeederTimer();

void setup() {

  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
```

Anexo 8: Código de AlimentadorESP32CAM (1)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
```

Anexo 9: Código de AlimentadorESP32CAM (2)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Inicio de cámara falló con error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  //initial sensors are flipped vertically and colors are a bit saturate
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);//flip it back
    s->set_brightness(s, 1);//up the blightness just a bit
    s->set_saturation(s, -2);//lower the saturation
  }
  //drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

  pinMode(4, OUTPUT);
```

Anexo 10: Código de AlimentadorESP32CAM (3)

AlimentadorESP32CAM § | app_httpd.cpp | camera_index.h | camera_pins.h

```
  if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("STA Failed to configure");
  }

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Conexión con WiFi exitosa");

  startCameraServer();

  Serial.print("¡Camara Lista! Por favor, use el siguiente link 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' para conectarse");
}

void loop() {

   checkFeederTimer();
   delay(10);

}
```

Anexo 11: Código de AlimentdorESP32CAM (4)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
unsigned long previousMillis = 0;
unsigned long feedinterval = 24UL*60UL*60UL*1000UL;
unsigned int feedingtime = 4;
bool feedingintervalenabled = false;
#define feederPin 12

void runFeeder();
void feedingAnalogWrite();
// --- End header changes for Fish Feeder ---

#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"

#include "fb_gfx.h"
#include "fd_forward.h"
#include "fr_forward.h"

#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7

#define FACE_COLOR_WHITE  0x00FFFFFF
#define FACE_COLOR_BLACK  0x00000000
#define FACE_COLOR_RED    0x000000FF
#define FACE_COLOR_GREEN  0x0000FF00
#define FACE_COLOR_BLUE   0x00FF0000
```

Anexo 12: Código de app_httpd.cpp (1)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
#define FACE_COLOR_YELLOW (FACE_COLOR_RED | FACE_COLOR_GREEN)
#define FACE_COLOR_CYAN   (FACE_COLOR_BLUE | FACE_COLOR_GREEN)
#define FACE_COLOR_PURPLE (FACE_COLOR_BLUE | FACE_COLOR_RED)

typedef struct {
        size_t size;
        size_t index;
        size_t count;
        int sum;
        int * values;
} ra_filter_t;

typedef struct {
        httpd_req_t *req;
        size_t len;
} jpg_chunking_t;

#define PART_BOUNDARY "123456789000000000000987654321"
static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";

static ra_filter_t ra_filter;
httpd_handle_t stream_httpd = NULL;
httpd_handle_t camera_httpd = NULL;

static mtmn_config_t mtmn_config = {0};
static int8_t detection_enabled = 0;
static int8_t recognition_enabled = 0;
```

Anexo 13: Código de app_httpd.cpp (2)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
static int8_t is_enrolling = 0;
static face_id_list id_list = {0};

static ra_filter_t * ra_filter_init(ra_filter_t * filter, size_t sample_size){
    memset(filter, 0, sizeof(ra_filter_t));

    filter->values = (int *)malloc(sample_size * sizeof(int));
    if(!filter->values){
        return NULL;
    }
    memset(filter->values, 0, sample_size * sizeof(int));

    filter->size = sample_size;
    return filter;
}

static int ra_filter_run(ra_filter_t * filter, int value){
    if(!filter->values){
        return value;
    }
    filter->sum -= filter->values[filter->index];
    filter->values[filter->index] = value;
    filter->sum += filter->values[filter->index];
    filter->index++;
    filter->index = filter->index % filter->size;
    if (filter->count < filter->size) {
        filter->count++;
    }
    return filter->sum / filter->count;
```

Anexo 14: Código de app_httpd.cpp (3)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
}

static void rgb_print(dl_matrix3du_t *image_matrix, uint32_t color, const char * str){
    fb_data_t fb;
    fb.width = image_matrix->w;
    fb.height = image_matrix->h;
    fb.data = image_matrix->item;
    fb.bytes_per_pixel = 3;
    fb.format = FB_BGR888;
    fb_gfx_print(&fb, (fb.width - (strlen(str) * 14)) / 2, 10, color, str);
}

static int rgb_printf(dl_matrix3du_t *image_matrix, uint32_t color, const char *format, ...){
    char loc_buf[64];
    char * temp = loc_buf;
    int len;
    va_list arg;
    va_list copy;
    va_start(arg, format);
    va_copy(copy, arg);
    len = vsnprintf(loc_buf, sizeof(loc_buf), format, arg);
    va_end(copy);
    if(len >= sizeof(loc_buf)){
        temp = (char*)malloc(len+1);
        if(temp == NULL) {
            return 0;
        }
    }
    vsnprintf(temp, len+1, format, arg);
```

Anexo 15: Código de app_httpd.cpp (4)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
    va_end(arg);
    rgb_print(image_matrix, color, temp);
    if(len > 64){
        free(temp);
    }
    return len;
}

static void draw_face_boxes(dl_matrix3du_t *image_matrix, box_array_t *boxes, int face_id){
    int x, y, w, h, i;
    uint32_t color = FACE_COLOR_YELLOW;
    if(face_id < 0){
        color = FACE_COLOR_RED;
    } else if(face_id > 0){
        color = FACE_COLOR_GREEN;
    }
    fb_data_t fb;
    fb.width = image_matrix->w;
    fb.height = image_matrix->h;
    fb.data = image_matrix->item;
    fb.bytes_per_pixel = 3;
    fb.format = FB_BGR888;
    for (i = 0; i < boxes->len; i++){
        // rectangle box
        x = (int)boxes->box[i].box_p[0];
        y = (int)boxes->box[i].box_p[1];
        w = (int)boxes->box[i].box_p[2] - x + 1;
        h = (int)boxes->box[i].box_p[3] - y + 1;
        fb_gfx_drawFastHLine(&fb, x, y, w, color);
```

Anexo 16: Código de app_httpd.cpp (5)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
        fb_gfx_drawFastHLine(&fb, x, y+h-1, w, color);
        fb_gfx_drawFastVLine(&fb, x, y, h, color);
        fb_gfx_drawFastVLine(&fb, x+w-1, y, h, color);
#if 0
        // landmark
        int x0, y0, j;
        for (j = 0; j < 10; j+=2) {
            x0 = (int)boxes->landmark[i].landmark_p[j];
            y0 = (int)boxes->landmark[i].landmark_p[j+1];
            fb_gfx_fillRect(&fb, x0, y0, 3, 3, color);
        }
#endif
    }
}

static int run_face_recognition(dl_matrix3du_t *image_matrix, box_array_t *net_boxes){
    dl_matrix3du_t *aligned_face = NULL;
    int matched_id = 0;

    aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
    if(!aligned_face){
        Serial.println("Could not allocate face recognition buffer");
        return matched_id;
    }
    if (align_face(net_boxes, image_matrix, aligned_face) == ESP_OK){
        if (is_enrolling == 1){
            int8_t left_sample_face = enroll_face(&id_list, aligned_face);

            if(left_sample_face == (ENROLL_CONFIRM_TIMES - 1)){
```

Anexo 17: Código de app_httpd.cpp (6)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
            Serial.printf("Enrolling Face ID: %d\n", id_list.tail);
        }
        Serial.printf("Enrolling Face ID: %d sample %d\n", id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face);
        rgb_printf(image_matrix, FACE_COLOR_CYAN, "ID[%u] Sample[%u]", id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face);
        if (left_sample_face == 0){
            is_enrolling = 0;
            Serial.printf("Enrolled Face ID: %d\n", id_list.tail);
        }
    } else {
        matched_id = recognize_face(&id_list, aligned_face);
        if (matched_id >= 0) {
            Serial.printf("Match Face ID: %u\n", matched_id);
            rgb_printf(image_matrix, FACE_COLOR_GREEN, "Hello Subject %u", matched_id);
        } else {
            Serial.println("No Match Found");
            rgb_print(image_matrix, FACE_COLOR_RED, "Intruder Alert!");
            matched_id = -1;
        }
    }
} else {
    Serial.println("Face Not Aligned");
    //rgb_print(image_matrix, FACE_COLOR_YELLOW, "Human Detected");
}

dl_matrix3du_free(aligned_face);
return matched_id;
}

static size_t jpg_encode_stream(void * arg, size_t index, const void* data, size_t len){
```

Anexo 18: Código de app_httpd.cpp (7)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
    jpg_chunking_t *j = (jpg_chunking_t *)arg;
    if(!index){
        j->len = 0;
    }
    if(httpd_resp_send_chunk(j->req, (const char *)data, len) != ESP_OK){
        return 0;
    }
    j->len += len;
    return len;
}

static esp_err_t capture_handler(httpd_req_t *req){
    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    int64_t fr_start = esp_timer_get_time();

    fb = esp_camera_fb_get();
    if (!fb) {
        Serial.println("Camera capture failed");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    httpd_resp_set_type(req, "image/jpeg");
    httpd_resp_set_hdr(req, "Content-Disposition", "inline; filename=capture.jpg");

    size_t out_len, out_width, out_height;
    uint8_t * out_buf;
    bool s;
```

Anexo 19: Código de app_httpd.cpp (8)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
bool detected = false;
int face_id = 0;
if(!detection_enabled || fb->width > 400){
    size_t fb_len = 0;
    if(fb->format == PIXFORMAT_JPEG){
        fb_len = fb->len;
        res = httpd_resp_send(req, (const char *)fb->buf, fb->len);
    } else {
        jpg_chunking_t jchunk = {req, 0};
        res = frame2jpg_cb(fb, 80, jpg_encode_stream, &jchunk)?ESP_OK:ESP_FAIL;
        httpd_resp_send_chunk(req, NULL, 0);
        fb_len = jchunk.len;
    }
    esp_camera_fb_return(fb);
    int64_t fr_end = esp_timer_get_time();
    Serial.printf("JPG: %uB %ums\n", (uint32_t)(fb_len), (uint32_t)((fr_end - fr_start)/1000));
    return res;
}

dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);
if (!image_matrix) {
    esp_camera_fb_return(fb);
    Serial.println("dl_matrix3du_alloc failed");
    httpd_resp_send_500(req);
    return ESP_FAIL;
}

out_buf = image_matrix->item;
out_len = fb->width * fb->height * 3;
```

Anexo 20: Código de app_httpd.cpp (9)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
    out_width = fb->width;
    out_height = fb->height;

    s = fmt2rgb888(fb->buf, fb->len, fb->format, out_buf);
    esp_camera_fb_return(fb);
    if(!s){
        dl_matrix3du_free(image_matrix);
        Serial.println("to rgb888 failed");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    box_array_t *net_boxes = face_detect(image_matrix, &mtmn_config);

    if (net_boxes){
        detected = true;
        if(recognition_enabled){
            face_id = run_face_recognition(image_matrix, net_boxes);
        }
        draw_face_boxes(image_matrix, net_boxes, face_id);
        free(net_boxes->box);
        free(net_boxes->landmark);
        free(net_boxes);
    }

    jpg_chunking_t jchunk = {req, 0};
    s = fmt2jpg_cb(out_buf, out_len, out_width, out_height, PIXFORMAT_RGB888, 90, jpg_encode_stream, &jchunk);
    dl_matrix3du_free(image_matrix);
    if(!s){
```

Anexo 21: Código de app_httpd.cpp (10)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
        Serial.println("JPEG compression failed");
        return ESP_FAIL;
    }

    int64_t fr_end = esp_timer_get_time();
    Serial.printf("FACE: %uB %ums %s%d\n", (uint32_t)(jchunk.len), (uint32_t)((fr_end - fr_start)/1000), detected?"DETECTED ":"", face_id);
    return res;
}

static esp_err_t stream_handler(httpd_req_t *req){
    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    size_t _jpg_buf_len = 0;
    uint8_t * _jpg_buf = NULL;
    char * part_buf[64];
    dl_matrix3du_t *image_matrix = NULL;
    bool detected = false;
    int face_id = 0;
    int64_t fr_start = 0;
    int64_t fr_ready = 0;
    int64_t fr_face = 0;
    int64_t fr_recognize = 0;
    int64_t fr_encode = 0;

    static int64_t last_frame = 0;
    if(!last_frame) {
        last_frame = esp_timer_get_time();
    }
```

Anexo 22: Código de app_httpd.cpp (11)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
    res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
    if(res != ESP_OK){
        return res;
    }

    while(true){
        detected = false;
        face_id = 0;
        fb = esp_camera_fb_get();
        if (!fb) {
            Serial.println("Camera capture failed");
            res = ESP_FAIL;
        } else {
            fr_start = esp_timer_get_time();
            fr_ready = fr_start;
            fr_face = fr_start;
            fr_encode = fr_start;
            fr_recognize = fr_start;
            if(!detection_enabled || fb->width > 400){
                if(fb->format != PIXFORMAT_JPEG){
                    bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
                    esp_camera_fb_return(fb);
                    fb = NULL;
                    if(!jpeg_converted){
                        Serial.println("JPEG compression failed");
                        res = ESP_FAIL;
                    }
                } else {
                    _jpg_buf_len = fb->len;
```

Anexo 23: Código de app_httpd.cpp (12)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
                _jpg_buf = fb->buf;
            }
        } else {

            image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);

            if (!image_matrix) {
                Serial.println("dl_matrix3du_alloc failed");
                res = ESP_FAIL;
            } else {
                if(!fmt2rgb888(fb->buf, fb->len, fb->format, image_matrix->item)){
                    Serial.println("fmt2rgb888 failed");
                    res = ESP_FAIL;
                } else {
                    fr_ready = esp_timer_get_time();
                    box_array_t *net_boxes = NULL;
                    if(detection_enabled){
                        net_boxes = face_detect(image_matrix, &mtmn_config);
                    }
                    fr_face = esp_timer_get_time();
                    fr_recognize = fr_face;
                    if (net_boxes || fb->format != PIXFORMAT_JPEG){
                        if(net_boxes){
                            detected = true;
                            if(recognition_enabled){
                                face_id = run_face_recognition(image_matrix, net_boxes);
                            }
                            fr_recognize = esp_timer_get_time();
                            draw_face_boxes(image_matrix, net_boxes, face_id);
```

Anexo 24: Código de app_httpd.cpp (13)


```
                free(net_boxes->box);
                free(net_boxes->landmark);
                free(net_boxes);
            }
            if(!fmt2jpg(image_matrix->item, fb->width*fb->height*3, fb->width, fb->height, PIXFORMAT_RGB888, 90, &_jpg_buf, &_jpg_buf_len)){
                Serial.println("fmt2jpg failed");
                res = ESP_FAIL;
            }
            esp_camera_fb_return(fb);
            fb = NULL;
        } else {
            _jpg_buf = fb->buf;
            _jpg_buf_len = fb->len;
        }
        fr_encode = esp_timer_get_time();
    }
    dl_matrix3du_free(image_matrix);
  }
 }
}
if(res == ESP_OK){
    size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
    res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
}
if(res == ESP_OK){
    res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
}
if(res == ESP_OK){
    res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
```

Anexo 25: Código de app_httpd.cpp (14)


```
 }
 if(fb){
     esp_camera_fb_return(fb);
     fb = NULL;
     _jpg_buf = NULL;
 } else if(_jpg_buf){
     free(_jpg_buf);
     _jpg_buf = NULL;
 }
 if(res != ESP_OK){
     break;
 }
 int64_t fr_end = esp_timer_get_time();

 int64_t ready_time = (fr_ready - fr_start)/1000;
 int64_t face_time = (fr_face - fr_ready)/1000;
 int64_t recognize_time = (fr_recognize - fr_face)/1000;
 int64_t encode_time = (fr_encode - fr_recognize)/1000;
 int64_t process_time = (fr_encode - fr_start)/1000;

 int64_t frame_time = fr_end - last_frame;
 last_frame = fr_end;
 frame_time /= 1000;
 uint32_t avg_frame_time = ra_filter_run(&ra_filter, frame_time);
 Serial.printf("MJPG: %uB %ums (%.1ffps), AVG: %ums (%.1ffps), %u+%u+%u+%u=%u %s%d\n",
     (uint32_t)(_jpg_buf_len),
     (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time,
     avg_frame_time, 1000.0 / avg_frame_time,
     (uint32_t)ready_time, (uint32_t)face_time, (uint32_t)recognize_time, (uint32_t)encode_time, (uint32_t)process_time,
```

Anexo 26: Código de app_httpd.cpp (15)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
            (detected)?"DETECTED ":"", face_id
        );
    }

    last_frame = 0;
    return res;
}

static esp_err_t cmd_handler(httpd_req_t *req){
    char*  buf;
    size_t buf_len;
    char variable[32] = {0,};
    char value[32] = {0,};

    buf_len = httpd_req_get_url_query_len(req) + 1;
    if (buf_len > 1) {
        buf = (char*)malloc(buf_len);
        if(!buf){
            httpd_resp_send_500(req);
            return ESP_FAIL;
        }
        if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
            if (httpd_query_key_value(buf, "var", variable, sizeof(variable)) == ESP_OK &&
                httpd_query_key_value(buf, "val", value, sizeof(value)) == ESP_OK) {
            } else {
                free(buf);
                httpd_resp_send_404(req);
                return ESP_FAIL;
            }
```

Anexo 27: Código de app_httpd.cpp (16)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
        } else {
            free(buf);
            httpd_resp_send_404(req);
            return ESP_FAIL;
        }
        free(buf);
    } else {
        httpd_resp_send_404(req);
        return ESP_FAIL;
    }

    int val = atoi(value);
    sensor_t * s = esp_camera_sensor_get();
    int res = 0;

    if(!strcmp(variable, "framesize")) {
        if(s->pixformat == PIXFORMAT_JPEG) res = s->set_framesize(s, (framesize_t)val);
    }
    else if(!strcmp(variable, "quality")) res = s->set_quality(s, val);
    else if(!strcmp(variable, "contrast")) res = s->set_contrast(s, val);
    else if(!strcmp(variable, "brightness")) res = s->set_brightness(s, val);
    else if(!strcmp(variable, "saturation")) res = s->set_saturation(s, val);
    else if(!strcmp(variable, "gainceiling")) res = s->set_gainceiling(s, (gainceiling_t)val);
    else if(!strcmp(variable, "colorbar")) res = s->set_colorbar(s, val);
    else if(!strcmp(variable, "awb")) res = s->set_whitebal(s, val);
    else if(!strcmp(variable, "agc")) res = s->set_gain_ctrl(s, val);
    else if(!strcmp(variable, "aec")) res = s->set_exposure_ctrl(s, val);
    else if(!strcmp(variable, "hmirror")) res = s->set_hmirror(s, val);
    else if(!strcmp(variable, "vflip")) res = s->set_vflip(s, val);
```

Anexo 28: Código de app_httpd.cpp (17)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
    else if(!strcmp(variable, "agc_gain")) res = s->set_agc_gain(s, val);
    else if(!strcmp(variable, "aec_value")) res = s->set_aec_value(s, val);
    else if(!strcmp(variable, "aec2")) res = s->set_aec2(s, val);
    else if(!strcmp(variable, "dcw")) res = s->set_dcw(s, val);
    else if(!strcmp(variable, "bpc")) res = s->set_bpc(s, val);
    else if(!strcmp(variable, "wpc")) res = s->set_wpc(s, val);
    else if(!strcmp(variable, "raw_gma")) res = s->set_raw_gma(s, val);
    else if(!strcmp(variable, "lenc")) res = s->set_lenc(s, val);
    else if(!strcmp(variable, "special_effect")) res = s->set_special_effect(s, val);
    else if(!strcmp(variable, "wb_mode")) res = s->set_wb_mode(s, val);
    else if(!strcmp(variable, "ae_level")) res = s->set_ae_level(s, val);
    else if(!strcmp(variable, "face_detect")) {
        detection_enabled = val;
        if(!detection_enabled) {
            recognition_enabled = 0;
        }
    }
    else if(!strcmp(variable, "face_enroll")) is_enrolling = val;
    else if(!strcmp(variable, "face_recognize")) {
        recognition_enabled = val;
        if(recognition_enabled){
            detection_enabled = val;
        }
    }
    else if(!strcmp(variable, "fishfeeder")) {
      runFeeder();
    }
    else if(!strcmp(variable, "feedingintervalenabled")) {
      feedingintervalenabled = val == 1;
```

Anexo 29: Código de app_httpd.cpp (18)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
    else if(!strcmp(variable, "lightenabled")) {
      if (val == 1)
        digitalWrite(4 , HIGH);
      else if (val == 0) {
        digitalWrite(4 , LOW);
      }
    }
    else if(!strcmp(variable, "feedinginterval")) {
      Serial.print("Feeding interval changed to: ");
      Serial.println(val);
      feedinterval = val*60UL*60UL*1000UL;
      previousMillis = 0;
    }
    else if(!strcmp(variable, "feedingtime")) {
      Serial.print("Feeding time (seconds) changed to: ");
      Serial.println(val);
      feedingtime = val;
    } else {
        res = -1;
    }

    if(res){
        return httpd_resp_send_500(req);
    }

    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, NULL, 0);
}
```

Anexo 30: Código de app_httpd.cpp (19)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
static esp_err_t status_handler(httpd_req_t *req){
    static char json_response[1024];

    sensor_t * s = esp_camera_sensor_get();
    char * p = json_response;
    *p++ = '{';

    p+=sprintf(p, "\"framesize\":%u,", s->status.framesize);
    p+=sprintf(p, "\"quality\":%u,", s->status.quality);
    p+=sprintf(p, "\"brightness\":%d,", s->status.brightness);
    p+=sprintf(p, "\"contrast\":%d,", s->status.contrast);
    p+=sprintf(p, "\"saturation\":%d,", s->status.saturation);
    p+=sprintf(p, "\"sharpness\":%d,", s->status.sharpness);
    p+=sprintf(p, "\"special_effect\":%u,", s->status.special_effect);
    p+=sprintf(p, "\"wb_mode\":%u,", s->status.wb_mode);
    p+=sprintf(p, "\"awb\":%u,", s->status.awb);
    p+=sprintf(p, "\"awb_gain\":%u,", s->status.awb_gain);
    p+=sprintf(p, "\"aec\":%u,", s->status.aec);
    p+=sprintf(p, "\"aec2\":%u,", s->status.aec2);
    p+=sprintf(p, "\"ae_level\":%d,", s->status.ae_level);
    p+=sprintf(p, "\"aec_value\":%u,", s->status.aec_value);
    p+=sprintf(p, "\"agc\":%u,", s->status.agc);
    p+=sprintf(p, "\"agc_gain\":%u,", s->status.agc_gain);
    p+=sprintf(p, "\"gainceiling\":%u,", s->status.gainceiling);
    p+=sprintf(p, "\"bpc\":%u,", s->status.bpc);
    p+=sprintf(p, "\"wpc\":%u,", s->status.wpc);
    p+=sprintf(p, "\"raw_gma\":%u,", s->status.raw_gma);
    p+=sprintf(p, "\"lenc\":%u,", s->status.lenc);
    p+=sprintf(p, "\"vflip\":%u,", s->status.vflip);
```

Anexo 31: Código de app_httpd.cpp (20)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
    p+=sprintf(p, "\"hmirror\":%u,", s->status.hmirror);
    p+=sprintf(p, "\"dcw\":%u,", s->status.dcw);
    p+=sprintf(p, "\"colorbar\":%u,", s->status.colorbar);
    p+=sprintf(p, "\"face_detect\":%u,", detection_enabled);
    p+=sprintf(p, "\"face_enroll\":%u,", is_enrolling);
    p+=sprintf(p, "\"face_recognize\":%u", recognition_enabled);
    *p++ = '}';
    *p++ = 0;
    httpd_resp_set_type(req, "application/json");
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, json_response, strlen(json_response));
}

static esp_err_t index_handler(httpd_req_t *req){
    httpd_resp_set_type(req, "text/html");
    httpd_resp_set_hdr(req, "Content-Encoding", "gzip");
    sensor_t * s = esp_camera_sensor_get();
    if (s->id.PID == OV3660_PID) {
        return httpd_resp_send(req, (const char *)index_ov3660_html_gz, index_ov3660_html_gz_len);
    }
    return httpd_resp_send(req, (const char *)index_ov2640_html_gz, index_ov2640_html_gz_len);
}

void feedingAnalogWrite(uint8_t channel, uint32_t value, uint32_t valueMax = 1000) {
  // calculate duty, 8191 from 2 ^ 13 - 1
  uint32_t duty = (16383 / valueMax) * min(value, valueMax);
  ledcWrite(channel, duty);
}
```

Anexo 32: Código de app_httpd.cpp (21)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
void startCameraServer(){


    pinMode(feederPin, OUTPUT);

    // Feeder servo PWM
    ledcSetup(4, 50, 16);         // channel, freq, resolution
    ledcAttachPin(feederPin, 4);  // pin, channel



    httpd_config_t config = HTTPD_DEFAULT_CONFIG();

    httpd_uri_t index_uri = {
        .uri       = "/",
        .method    = HTTP_GET,
        .handler   = index_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t status_uri = {
        .uri       = "/status",
        .method    = HTTP_GET,
        .handler   = status_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t cmd_uri = {
        .uri       = "/control",
```

Anexo 33: Código de app_httpd.cpp (22)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
        .method    = HTTP_GET,
        .handler   = cmd_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t capture_uri = {
        .uri       = "/capture",
        .method    = HTTP_GET,
        .handler   = capture_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t stream_uri = {
        .uri       = "/stream",
        .method    = HTTP_GET,
        .handler   = stream_handler,
        .user_ctx  = NULL
    };

    ra_filter_init(&ra_filter, 20);

    mtmn_config.min_face = 80;
    mtmn_config.pyramid = 0.7;
    mtmn_config.p_threshold.score = 0.6;
    mtmn_config.p_threshold.nms = 0.7;
    mtmn_config.r_threshold.score = 0.7;
    mtmn_config.r_threshold.nms = 0.7;
    mtmn_config.r_threshold.candidate_number = 4;
    mtmn_config.o_threshold.score = 0.7;
```

Anexo 34: Código del app_httpd.cpp (23)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h | camera_pins.h

```
    mtmn_config.o_threshold.nms = 0.4;
    mtmn_config.o_threshold.candidate_number = 1;

    face_id_init(&id_list, FACE_ID_SAVE_NUMBER, ENROLL_CONFIRM_TIMES);

    Serial.printf("Empezando servidor web en puerto: '%d'\n", config.server_port);
    if (httpd_start(&camera_httpd, &config) == ESP_OK) {
        httpd_register_uri_handler(camera_httpd, &index_uri);
        httpd_register_uri_handler(camera_httpd, &cmd_uri);
        httpd_register_uri_handler(camera_httpd, &status_uri);
        httpd_register_uri_handler(camera_httpd, &capture_uri);
    }

    config.server_port += 1;
    config.ctrl_port += 1;
    Serial.printf("Empezando servicio de stream en puerto: '%d'\n", config.server_port);
    if (httpd_start(&stream_httpd, &config) == ESP_OK) {
        httpd_register_uri_handler(stream_httpd, &stream_uri);
    }
}

void checkFeederTimer() {

  if (!feedingintervalenabled)
    return;

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= feedinterval) {
```

Anexo 35: Código del app_httpd.cpp (24)

```
        Serial.println("Feeder timer activated.");
        runFeeder();
      }
    }

    void runFeeder() {
      Serial.print("Start feeding the fish for ");
      Serial.print(feedingtime);
      Serial.println(" seconds");

      feedingAnalogWrite(4, 750);
      delay(feedingtime * 1000);
      feedingAnalogWrite(4, 0);

      Serial.println("Feeding the fish finished");
    }
```

Anexo 36: Código del app_httpd.cpp (25)

AlimentadorESP32CAM § | app_httpd.cpp § | camera_index.h § | camera_pins.h

```
//File: index_ov2640.html.gz, Size: 4316
#define index_ov2640_html_gz_len 4907
const uint8_t index_ov2640_html_gz[] = {
  0x1f,0x8b,0x08,0x08,0xf6,0x7c,0x55,0x5d,0x00,0xff,0x69,0x6e,0x64,0x65,0x78,0x5f,0x6f,0x76,0x32,
  0x36,0x34,0x30,0x5f,0x68,0x74,0x6d,0x6c,0x5f,0x67,0x7a,0x5f,0x6c,0x65,0x6e,0x00,0xe5,0x5d,0xfd,
  0x73,0xda,0x46,0xfa,0xff,0xbd,0x7f,0x85,0xac,0xf6,0x62,0x32,0x35,0x60,0xb0,0xeb,0xa4,0x9c,0x4d,
  0xbe,0x89,0xe3,0x24,0x9d,0x69,0xd2,0x5e,0xdc,0x6b,0x3b,0xd3,0xb9,0x49,0x04,0x5a,0x40,0x57,0x21,
  0x71,0x92,0x30,0x76,0x33,0xf9,0xdf,0xef,0xb3,0x2f,0x5a,0xad,0xc4,0x6a,0x25,0x81,0x0d,0xfd,0xf6,
  0x60,0x06,0x84,0xb4,0xcf,0xb3,0xcf,0xfb,0x3e,0xfb,0xca,0xf9,0x81,0x1b,0x8e,0x93,0xbb,0x05,0xb1,
  0x66,0xc9,0xdc,0x1f,0x7e,0x71,0xce,0xbf,0x2c,0xbc,0xce,0x67,0xc4,0x71,0xf9,0x25,0xfb,0x39,0x27,
  0x89,0x63,0x8d,0x67,0x4e,0x14,0x93,0xe4,0xc2,0x5e,0x26,0x93,0xf6,0x53,0xbb,0xf8,0x38,0x70,0xe6,
  0xe4,0xc2,0xbe,0xf1,0xc8,0x6a,0x11,0x46,0x89,0x6d,0x8d,0xc3,0x20,0x21,0x01,0x8a,0xaf,0x3c,0x37,
  0x99,0x5d,0xb8,0xe4,0xc6,0x1b,0x93,0x36,0xfb,0x71,0xe4,0x05,0x5e,0xe2,0x39,0x7e,0x3b,0x1e,0x3b,
  0x3e,0xb9,0xe8,0xa9,0xb8,0x12,0x2f,0xf1,0xc9,0xf0,0xea,0xfa,0xc7,0x93,0xbe,0xf5,0xc3,0xcf,0xfd,
  0xd3,0xb3,0xe3,0xf3,0x2e,0xbf,0x97,0x95,0x89,0x93,0x3b,0xf5,0x37,0x7d,0x8d,0x42,0xf7,0xce,0xfa,
  0x94,0xbb,0x45,0x5f,0x13,0x10,0xd1,0x9e,0x38,0x73,0xcf,0xbf,0x1b,0x58,0xcf,0x23,0xd4,0x79,0xf4,
  0x86,0xf8,0x37,0x24,0xf1,0xc6,0xce,0x51,0xec,0x04,0x71,0x3b,0x26,0x91,0x37,0xf9,0xfb,0x1a,0xe0,
  0xc8,0x19,0xff,0x3e,0x8d,0xc2,0x65,0xe0,0x0e,0xac,0x2f,0x7b,0x4f,0xe9,0x7b,0xbd,0xd0,0x38,0xf4,
  0xc3,0x08,0xcf,0xaf,0x5e,0xd1,0xf7,0xfa,0x73,0x56,0x7b,0xec,0xfd,0x41,0x06,0x56,0xef,0x6c,0x71,
  0x9b,0x7b,0xfe,0xf9,0x8b,0xdc,0xcf,0x59,0xbf,0x8c,0x7a,0x01,0xff,0xd4,0x0c,0x1f,0x93,0x71,0xe2,
  0x85,0x41,0x67,0xee,0x78,0x81,0x06,0x93,0xeb,0xc5,0x0b,0xdf,0x81,0x0c,0x26,0x3e,0x31,0xe2,0xf9,
  0x72,0x4e,0x82,0xe5,0x51,0x05,0x36,0x8a,0xa4,0xed,0x7a,0x11,0x2f,0x35,0xa0,0x72,0x58,0xce,0x83,
  0x4a,0xb4,0x26,0xba,0x82,0x30,0x20,0x1a,0x01,0xd2,0x8a,0x56,0x91,0xb3,0xa0,0x05,0xe8,0xf7,0x7a,
  0x91,0xb9,0x17,0x70,0xa3,0x1a,0x58,0x27,0xa7,0xc7,0x8b,0xdb,0x0a,0x55,0x9e,0x9c,0xd1,0xf7,0x7a,
  0xa1,0x85,0xe3,0xba,0x5e,0x30,0x1d,0x58,0x90,0xb3,0x06,0x45,0x18,0xb9,0x24,0x6a,0x47,0x8e,0xeb,
  0x2d,0xe3,0x81,0x75,0xaa,0x2b,0x33,0x77,0xa2,0x29,0x68,0x49,0x42,0x10,0xdb,0xee,0x69,0x29,0x11,
  0x45,0x22,0x6f,0x3a,0x4b,0xa0,0xd2,0xb5,0x32,0x45,0xa1,0x09,0x17,0xaa,0xd2,0xa7,0x51,0x6e,0x7a,
  0xa9,0x39,0xbe,0x37,0x0d,0xda,0x5e,0x42,0xe6,0x60,0x27,0x4e,0x22,0x92,0x8c,0x67,0x26,0x52,0x26,
  0xde,0x74,0x19,0x11,0x0d,0x21,0x52,0x6e,0x06,0x86,0xf1,0x70,0xfd,0x51,0x7b,0x45,0x46,0xbf,0x7b,
```

Anexo 37: Código de camera_index.h (1)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
  0x49,0x5b,0xc8,0x64,0x44,0x26,0x61,0x04,0x3b,0xd7,0x94,0x4c,0x4b,0xf8,0xe1,0xf8,0xf7,0x76,0x9c,
  0x38,0x11,0x64,0x57,0x8d,0xd0,0x99,0x24,0x04,0xbe,0x59,0x85,0x8f,0x50,0xab,0xa8,0xc6,0x56,0x5e,
  0xad,0x28,0xe0,0x05,0xbe,0x17,0x90,0xfa,0xe4,0x95,0xd5,0x9b,0x47,0xc7,0x4b,0xd5,0x50,0x8c,0x37,
  0x9f,0x9a,0xac,0x84,0xf1,0xba,0x5e,0x99,0xf0,0x9b,0xde,0xf1,0xf1,0xdf,0xd6,0x1f,0xce,0x08,0x37,
  0x53,0x67,0x99,0x84,0xdb,0x7b,0xc4,0x9a,0x5b,0x15,0xf8,0xf8,0xbf,0x39,0x71,0x3d,0xc7,0x6a,0x29,
  0xee,0xfc,0xf4,0x18,0x36,0xf5,0xd8,0x72,0x02,0xd7,0x6a,0x85,0x91,0x07,0x47,0x70,0x58,0xb8,0xf1,
  0x71,0x07,0x0d,0xc7,0x82,0x3c,0xd6,0xb0,0x6c,0xf0,0x19,0x55,0x22,0x7a,0xb7,0xa9,0x19,0x72,0x6a,
  0x39,0x90,0x86,0xc7,0x4a,0x7d,0xd5,0xd1,0x19,0x17,0x2c,0x48,0x34,0xe9,0x2e,0x2d,0x94,0xea,0x10,
  0xcd,0xec,0xb8,0x85,0xa2,0x37,0x33,0xab,0x6d,0xd1,0x28,0xf9,0x58,0x0f,0x23,0x90,0xea,0x55,0x5e,
  0x34,0x8a,0x06,0xec,0xea,0x59,0xcd,0x62,0x07,0x7f,0xeb,0x6c,0xa8,0x22,0x8a,0x34,0x8b,0x24,0x0d,
  0xa2,0x49,0xa3,0x88,0x52,0x3b,0xaa,0x34,0x8a,0x2c,0x4d,0xa2,0x4b,0x83,0x08,0x53,0x2b,0xca,0x70,
  0x75,0x56,0xe7,0x1b,0x5f,0x8e,0x96,0x49,0x12,0x06,0xf1,0x56,0x4d,0x54,0x99,0x9f,0xfd,0x7b,0x19,
  0x27,0xde,0xe4,0xae,0x2d,0x5c,0x1a,0x7e,0xb6,0x70,0x90,0x42,0x8e,0x48,0xb2,0x22,0xc4,0x9c,0x6e,
  0x04,0xce,0x0d,0xe2,0xce,0x74,0xea,0xeb,0x6c,0x6f,0xbc,0x8c,0x62,0x9a,0xb7,0x2d,0x42,0x0f,0x88,
  0xa3,0xf5,0x8a,0xf3,0x3e,0x58,0xb3,0xa2,0xf6,0x78,0xa4,0xa9,0x2b,0x5c,0x26,0x54,0xc6,0x5a,0x4d,
  0x84,0x60,0xc7,0x4b,0x50,0x8d,0xe6,0x99,0xf0,0x44,0xcd,0x93,0xd4,0x05,0x8d,0xcd,0x42,0x9e,0xae,
  0xc1,0x78,0x46,0xc6,0xbf,0x13,0xf7,0xeb,0xca,0x34,0xac,0x2a,0x3d,0xec,0x78,0xc1,0x62,0x99,0xb4,
  0x69,0x3a,0xb5,0x78,0x10,0x9d,0x33,0x83,0x4c,0x59,0xec,0xf7,0x4d,0x49,0xc5,0x37,0x8b,0x5b,0xb3,
  0x10,0x54,0x62,0x87,0xbe,0x33,0x22,0xbe,0x89,0x64,0xe1,0x0c,0x25,0x61,0x57,0xc4,0xaa,0xf2,0xdc,
  0xad,0x90,0x8b,0x9e,0x3e,0xf9,0x5b,0x6d,0x39,0xb2,0xeb,0xa3,0xdc,0xad,0x98,0xf8,0x70,0xb0,0xb2,
  0xd4,0x1b,0x65,0x56,0xa0,0xc1,0x58,0x41,0xe4,0x04,0x53,0x82,0x58,0x70,0x7b,0x94,0x5e,0x9a,0x3b,
  0x06,0xb5,0xd8,0xa7,0xa1,0x1a,0x62,0x37,0x55,0xcc,0x03,0xc2,0x06,0xc9,0x88,0xa2,0x56,0x63,0xfd,
  0x3d,0xad,0x51,0xf0,0x7c,0x44,0xeb,0x30,0x79,0x93,0xd2,0xe6,0xf7,0x95,0x11,0x21,0xed,0xe9,0x4d,
  0x26,0x55,0x7d,0xc5,0xc9,0xe4,0xe4,0xf8,0xe4,0xb4,0x32,0x61,0xd2,0x72,0x59,0xe8,0x2f,0x6a,0x22,
  0x86,0x8c,0x26,0xd5,0x2a,0x18,0xcc,0xc2,0x1b,0x12,0x69,0x14,0x51,0x20,0xf7,0xf4,0xdb,0x53,0xb7,
  0x06,0x36,0x07,0xf1,0xfe,0x46,0x17,0x4d,0xf3,0xe8,0xfa,0xbd,0x71,0xdf,0x68,0x98,0x1c,0x5d,0x07,
  0xd6,0xe0,0x8c,0x7c,0xe2,0x1a,0xc2,0xb3,0x4b,0x26,0xce,0xd2,0x4f,0x2a,0xe4,0xed,0x1c,0xd3,0xb7,
  0xa9,0x46,0xe6,0x57,0xbf,0xd1,0x81,0x8e,0x0b,0xe6,0x09,0xff,0xd2,0xd4,0x99,0xb6,0x9d,0xce,0x62,
  0x41,0x1c,0x94,0x1a,0x43,0xc6,0xfa,0x2e,0x69,0xad,0x9c,0x59,0x1f,0xb8,0x6a,0x75,0x44,0x2b,0x4d,
```

Anexo 38: Código de camera_index.h (2)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0x51,0x66,0x43,0x8d,0x78,0x1e,0x4c,0xc2,0xf1,0x52,0xd7,0x4c,0xd7,0x33,0xa9,0x75,0x7c,0x83,0x54,
0x64,0xb1,0xef,0x31,0xc3,0x5e,0x06,0x01,0xd5,0x68,0x3b,0x89,0xc0,0xa6,0xa6,0xa2,0x7a,0x82,0xdb,
0xc8,0x3b,0x73,0x82,0x2d,0x1b,0x8c,0x29,0x38,0xa0,0x26,0x50,0xc8,0x18,0x62,0xc5,0x21,0x98,0x4a,
0x51,0x6d,0x27,0x97,0x64,0xb6,0x9c,0xeb,0x12,0x83,0xb4,0xb2,0x1e,0x5a,0x31,0x5e,0x5d,0x34,0x1d,
0x39,0xad,0xe3,0xa3,0xe3,0xa3,0x13,0x7c,0x68,0x12,0x74,0xb3,0x71,0x09,0xf1,0x96,0x58,0x5e,0x21,
0xf8,0x54,0x8f,0x93,0x94,0x85,0xb1,0x4a,0x5d,0xd4,0xf7,0xa4,0xfc,0x80,0x49,0xaf,0x53,0xd1,0xb0,
0x94,0x98,0x74,0x73,0x43,0xd4,0x58,0x4b,0x53,0x15,0xcf,0xc3,0x3f,0x20,0x4c,0xda,0xaa,0xfe,0xcf,
0x5b,0xbb,0x22,0x8a,0xbf,0xb4,0xa5,0x37,0x96,0x4b,0xbc,0x6f,0xdb,0x40,0x8f,0xba,0x54,0x3e,0x22,
0x9f,0x01,0x85,0x01,0x3a,0x55,0x11,0x7a,0x57,0xa5,0x39,0x8f,0x52,0x66,0x03,0x19,0x4c,0x3c,0xdf,
0x6f,0xfb,0xe1,0xaa,0x3a,0x13,0x31,0x5b,0xf2,0x9a,0x9d,0x56,0x9b,0xfc,0xa6,0xd4,0x2e,0x11,0xb9,
0xfe,0x5f,0x50,0xfb,0xbf,0xd6,0xb4,0x28,0xae,0xb1,0x59,0x43,0xb1,0x81,0x3d,0x6e,0x57,0x51,0x2d,
0x53,0xe2,0x99,0xa0,0xb1,0x33,0x17,0xaf,0x3c,0x8c,0x2f,0x6e,0xd0,0xa9,0x5a,0x84,0x31,0x66,0xda,
0xe8,0x1c,0x4d,0x44,0x7c,0x0c,0x9f,0xde,0x68,0xda,0xe1,0x1a,0x5d,0xee,0xca,0x8e,0x89,0x0a,0x5e,
0x87,0x13,0x26,0xba,0x3f,0xcf,0x70,0x49,0x87,0xe7,0x0e,0xe5,0xb1,0x5a,0x6f,0xd6,0x15,0xe9,0x7e,
0xde,0x33,0xf4,0x85,0x1a,0x44,0xf4,0x34,0x68,0x4f,0x23,0x72,0x57,0x83,0x99,0x23,0xf1,0x3d,0xe0,
0x03,0xa2,0x9b,0xf7,0xfd,0x59,0x03,0x20,0xac,0xa8,0x73,0x1a,0xd7,0xa8,0xba,0xbc,0xca,0x3a,0xf6,
0x28,0x87,0xfb,0x6c,0xbb,0x46,0xb8,0x31,0x34,0xa1,0x7a,0x53,0x4d,0x5b,0x5f,0xed,0x43,0x9f,0x4c,
0x00,0xa8,0x9d,0xcd,0x60,0x79,0xea,0x89,0x39,0xba,0xa5,0x2a,0xa2,0xe3,0x04,0x95,0x91,0x43,0x8e,
0xca,0x95,0x5b,0x9f,0x16,0x33,0x8d,0x9e,0x8d,0x91,0x97,0xab,0x24,0x4d,0x9f,0x99,0x9a,0x51,0x66,
0x2e,0x9a,0x7c,0xa8,0x87,0xfc,0xda,0xea,0x43,0x4c,0x9a,0x06,0xc3,0x50,0xd8,0x3c,0x6a,0x5c,0x32,
0xac,0xb5,0xde,0x64,0x95,0x76,0x90,0xd5,0x58,0xa4,0x55,0x94,0xd9,0x2b,0x4d,0x11,0x66,0x7d,0x8c,
0xc6,0x3c,0x86,0x37,0x77,0x90,0xf6,0x52,0x73,0xc5,0x34,0xb9,0x56,0x7f,0x75,0xcc,0x5d,0x19,0x34,
0xec,0x9d,0x21,0xcc,0x18,0xab,0x1c,0xfb,0x61,0x6c,0xf6,0x2b,0x67,0x04,0xf9,0x2d,0x13,0x4d,0x45,
0x62,0xe8,0x52,0x3b,0xf2,0xc4,0x8c,0x5b,0xfb,0xa4,0x56,0xd3,0x6d,0xf4,0x29,0xb3,0x3b,0x16,0x64,
0x8e,0xa4,0x58,0x1b,0x26,0x4d,0xe3,0x6f,0x09,0xb9,0x45,0x7f,0x93,0x4e,0xc8,0x61,0xa6,0x0b,0x91,
0x43,0x17,0x46,0x73,0x8d,0x1c,0x5d,0x4c,0xb1,0x55,0xc2,0xdf,0x99,0x79,0xae,0x4b,0x8c,0xa3,0x9c,
0xb4,0xcf,0x5b,0x40,0x21,0xd7,0xaf,0x74,0x95,0x05,0x2c,0xe7,0xdd,0x6c,0xad,0xcd,0x39,0x5d,0xc5,
0xa2,0xae,0x73,0xe1,0x93,0x2c,0xd6,0xd8,0x77,0xe2,0xf8,0xc2,0xa6,0xab,0x31,0x94,0xa5,0x32,0xac,
0x88,0xeb,0xdd,0x58,0x9e,0x7b,0x61,0xfb,0xe1,0x34,0x2c,0x3c,0x63,0xcf,0xf9,0xb0,0x37,0x3c,0xf5,
```

Anexo 39: Código de camera_index.h (3)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0xc2,0xce,0x4d,0x09,0xd8,0x0c,0x2a,0xbb,0x65,0x0f,0x1f,0x7d,0xf9,0xed,0x93,0x27,0x67,0x7f,0x7f,
0x14,0x8c,0xe2,0x85,0xf8,0xfc,0x89,0xcf,0xa0,0x60,0x19,0xce,0xd9,0x29,0x12,0x5a,0x92,0x24,0x18,
0x83,0x8d,0xcf,0xbb,0x0c,0x69,0x81,0x90,0x2e,0x28,0x29,0xa1,0x4d,0x04,0x74,0x1d,0x79,0x69,0x91,
0x18,0x31,0x6a,0xe4,0x44,0x9a,0x22,0xac,0x18,0x4f,0x17,0x58,0xa6,0x65,0xb3,0xc0,0x36,0x0a,0x6f,
0x8b,0x1c,0x30,0xa6,0x44,0xd4,0x13,0xa5,0x88,0x5b,0x86,0x10,0x60,0x0c,0x9c,0xce,0x87,0x94,0x94,
0x91,0xf4,0x09,0xe9,0x2b,0xc3,0xf3,0xbc,0xea,0x49,0x84,0x15,0x4f,0x34,0x10,0x89,0x9b,0xe5,0x68,
0x8a,0x9a,0x90,0x90,0xf6,0xf0,0x3d,0x61,0xee,0x0a,0x2d,0x6b,0xc5,0xba,0x86,0x45,0x44,0xd0,0x5c,
0xfd,0x60,0x9b,0x93,0x28,0x46,0x4c,0xdb,0x74,0xac,0x36,0x2c,0x9a,0x8a,0x16,0x5d,0xb8,0x60,0x06,
0x76,0xe3,0xf8,0x4b,0x88,0xb6,0x77,0x6c,0x0f,0xff,0xf9,0xeb,0xeb,0xe7,0x2d,0x04,0xa2,0xe3,0xdb,
0x5e,0xff,0xf8,0xf8,0xf1,0x79,0x97,0x17,0x69,0x8c,0xeb,0x5b,0x7b,0x78,0xcd,0x50,0xf5,0x9f,0x02,
0xd5,0x71,0xff,0x74,0x73,0x54,0x58,0x6b,0xc6,0x30,0x01,0xc9,0xed,0x93,0xb3,0xa7,0x9b,0x23,0x7a,
0x02,0x9a,0x7e,0x06,0x26,0xac,0x2c,0xb8,0x05,0x87,0x9b,0x23,0x3a,0xb3,0x87,0x14,0x0f,0xbc,0xe2,
0xf6,0xf4,0xe9,0x16,0x78,0xbe,0xb1,0x45,0x93,0x48,0x4d,0x36,0xbd,0xb2,0x87,0x97,0xdf,0xbd,0x6a,
0x9d,0x82,0xc6,0xfe,0xb7,0x67,0x9b,0xe3,0x3e,0xb5,0x87,0xff,0xa0,0x44,0x9e,0xf4,0x81,0xe8,0x74,
0x0b,0x22,0x4f,0xec,0xe1,0x1b,0x86,0x09,0x58,0x6e,0x7b,0x4f,0xb6,0x20,0x09,0xe6,0xf5,0x0f,0x86,
0x09,0xf6,0x45,0xcd,0xab,0x26,0x26,0xc4,0x4b,0x26,0x1a,0x83,0x9f,0xae,0x47,0x9f,0xda,0x6e,0xfc,
0x9f,0x25,0x9a,0x8e,0xe4,0xae,0xb1,0x13,0x0b,0x38,0xb0,0xc4,0x2f,0xea,0xf9,0xaf,0x42,0x89,0x9c,
0x96,0xb3,0x87,0x3d,0x2c,0x72,0x34,0x72,0xb0,0x16,0x05,0x19,0x70,0x8e,0x01,0x9b,0x66,0x12,0xcc,
0x87,0xe9,0xd2,0x0f,0xd8,0xe8,0x89,0xad,0xf8,0xf5,0x46,0x21,0x42,0x43,0xad,0x73,0x6b,0x0f,0xcf,
0x4e,0xaa,0xe4,0xbd,0x85,0x3a,0x46,0x2c,0x4d,0x09,0x48,0x1c,0x37,0xd6,0x48,0x06,0x6a,0x0f,0x5f,
0xc8,0xeb,0x6d,0xf4,0xd2,0xee,0x6f,0xa1,0x17,0x85,0x1c,0xae,0x9a,0x76,0x5f,0xa8,0x06,0xdf,0xd2,
0x23,0xee,0x53,0x31,0x55,0xd4,0x6e,0xa3,0x17,0xda,0x88,0x47,0x4e,0x9c,0xde,0xab,0xaf,0x95,0x14,
0x10,0x61,0x4d,0x5c,0xed,0x4d,0x23,0x92,0x94,0xbf,0x80,0x3e,0x62,0x27,0x59,0x46,0x6c,0x41,0x5c,
0x63,0x8d,0x64,0xa0,0x68,0x0f,0xe5,0xf5,0xde,0xb4,0xa2,0x90,0xf3,0x57,0xd0,0xcb,0x82,0x8c,0xb1,
0xcc,0xfc,0x03,0x99,0x4c,0xd0,0x64,0x35,0xd7,0x4d,0x0e,0x1c,0xfa,0xe1,0xbf,0xad,0x2b,0xf6,0xbb,
0x71,0x8e,0x58,0x40,0x77,0x5f,0x89,0x22,0xd4,0xa1,0xcb,0x5b,0xde,0x85,0x92,0xce,0x0d,0x33,0x04,
0x6c,0x05,0x78,0x47,0xa6,0xac,0xa7,0xbc,0x31,0x8e,0xbe,0x3d,0x7c,0x1d,0x39,0x77,0x6c,0x6f,0xc1,
0x36,0x49,0xcf,0x7b,0xac,0x4e,0xf8,0x09,0x5d,0xc1,0x6d,0x32,0xb0,0xd7,0x11,0x56,0xaa,0x6d,0x87,
0xe5,0x1b,0x34,0x66,0xb8,0xd8,0x0e,0x09,0x12,0xd6,0x6b,0xb2,0xf0,0x9c,0x3f,0x43,0xc2,0xe5,0xac,
```

Anexo 40: Código de camera_index.h (4)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0x46,0x8d,0xdd,0x02,0x30,0xf6,0xf0,0xf9,0x2f,0x2f,0x1a,0x07,0x29,0x3e,0xde,0x5c,0xc7,0xc2,0x79,
0x74,0x12,0x04,0xda,0x6b,0x9d,0x4d,0xbd,0xe7,0xd4,0xed,0x70,0x6a,0xf8,0x4a,0x09,0x64,0xc3,0x73,
0xb6,0xc2,0x66,0x3d,0x1e,0x1f,0x2e,0x82,0x81,0x88,0x0f,0x53,0x0c,0x35,0x6c,0xa2,0x24,0x06,0xc8,
0x34,0x65,0xbd,0xc6,0xd5,0xae,0xd4,0xc5,0xab,0xdd,0x9b,0xce,0x04,0xd7,0xfb,0x56,0x1c,0x08,0x99,
0x87,0x6e,0xf3,0xe1,0x08,0x01,0x67,0x0f,0xa1,0xb5,0xb7,0xb8,0x68,0xdc,0xca,0xa4,0x08,0x1e,0xb8,
0x79,0x79,0x8e,0x45,0xe7,0xdb,0xb4,0x2c,0xd7,0x58,0x19,0x82,0x6e,0xda,0xe6,0xcd,0xca,0xa5,0x1f,
0x2e,0xdd,0xcd,0x31,0xa0,0x4d,0xf9,0x61,0x32,0xc1,0x2e,0xb8,0x6d,0x5a,0x94,0x37,0xe1,0xbc,0x26,
0xfc,0x03,0x47,0x71,0x32,0x6e,0x1e,0x20,0xc8,0x18,0x5a,0xbc,0xba,0xb4,0xae,0xaf,0xde,0x5d,0xff,
0xf0,0x7e,0x37,0xd1,0x01,0x75,0xee,0x29,0x30,0x50,0x6e,0xf7,0x1e,0xcc,0xc9,0xb8,0xbf,0x89,0x9e,
0x60,0xed,0x54,0x51,0x2f,0xaf,0x7f,0xdc,0x95,0x96,0x90,0xec,0xef,0x4b,0x4d,0x60,0x76,0xff,0x7a,
0xfa,0xe0,0x93,0x1b,0xe2,0x6f,0xa0,0x2b,0x0e,0x48,0xf5,0x65,0x7d,0x4f,0xaf,0xf6,0xd6,0x91,0x93,
0xa4,0xfc,0x05,0xba,0x71,0xb0,0x8a,0x0f,0x8c,0xe8,0x4d,0x9c,0x87,0x43,0xda,0xc3,0xab,0x5b,0xcc,
0xcb,0x61,0xff,0xd3,0x36,0x1a,0xd9,0x66,0x64,0x30,0x23,0x85,0x6b,0x24,0x1d,0x1a,0xa4,0x23,0xfb,
0x52,0x27,0xfd,0xe3,0xd3,0x7b,0xd5,0x0a,0x45,0xfe,0x90,0x8a,0x99,0x6e,0xd0,0xee,0x4c,0x69,0xbb,
0xf3,0xfa,0x72,0x37,0xa1,0x0c,0x95,0xed,0x29,0x92,0x51,0x36,0xf7,0x17,0xc8,0x2c,0x3e,0x29,0x2a,
0xd5,0xb4,0x61,0x27,0x42,0x00,0xa2,0xef,0xbc,0x49,0x07,0x42,0x1d,0x54,0xbf,0xdd,0xc6,0x75,0x52,
0x32,0xf2,0x9e,0x73,0x92,0xf9,0x0d,0xe6,0x6e,0xee,0xd1,0x6b,0x4e,0x2a,0xa9,0xdd,0xc6,0x69,0x28,
0x27,0x63,0xe2,0x61,0x12,0x7c,0xda,0x58,0x21,0x0a,0x2c,0xd7,0x89,0x75,0xc9,0x7f,0x6d,0xa3,0x9b,
0xfe,0x36,0xba,0x51,0x29,0xca,0xab,0xe7,0xec,0x81,0x5a,0x1a,0x4c,0x62,0x3e,0xa4,0x7a,0x46,0x8b,
0xe6,0x31,0x0d,0x30,0x18,0x18,0xfa,0x71,0x37,0x31,0x8d,0x56,0x56,0x33,0xa6,0x6d,0x15,0xc1,0x18,
0x53,0x7b,0xef,0x46,0x6f,0xa0,0x0d,0xc0,0xa0,0xfb,0xbc,0x23,0x6d,0xd0,0xca,0xf6,0xd3,0xc2,0x30,
0x36,0xf7,0xad,0x9f,0xc8,0x59,0x7d,0x98,0xce,0x9d,0xc6,0x3a,0x12,0x70,0x18,0xd8,0x75,0x56,0xd6,
0xeb,0xb7,0xcf,0x77,0xa2,0xab,0xb4,0xd2,0xfd,0xe8,0x4b,0xb2,0xbc,0x6f,0x9d,0xf9,0x24,0x68,0xee,
0x54,0x14,0xc8,0x1e,0x7e,0x4f,0xb0,0xbd,0xfc,0x32,0x8c,0xc4,0x69,0x33,0x3b,0xd1,0x1a,0xab,0x79,
0x3f,0x2a,0xe3,0x4c,0xef,0x5b,0x5f,0xb3,0xb9,0x17,0x45,0x61,0xd4,0x58,0x65,0x02,0x0e,0xc3,0x54,
0xed,0xb7,0xec,0x6a,0x27,0xea,0x4a,0x6b,0xdd,0x8f,0xc6,0x24,0xcf,0xfb,0x56,0xda,0xcd,0xc4,0xf7,
0x16,0x8d,0x55,0xc6,0xa0,0xb0,0x9e,0xa9,0xfd,0x0a,0xdf,0x3b,0x51,0x17,0xaf,0x71,0x3f,0xca,0x12,
0xdc,0xee,0x5b,0x55,0xee,0x78,0xd5,0x58,0x51,0x80,0xb1,0x87,0x2f,0x2f,0x7f,0xb1,0x5a,0x2f,0xc3,
0x15,0x56,0xe4,0xff,0x41,0xac,0xab,0x77,0x58,0x42,0xb5,0x03,0x8d,0xd1,0xaa,0xf7,0xa3,0x2f,0xc6,
```

Anexo 41: Código de camera_index.h (5)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0xf4,0xbe,0xb5,0xc5,0x56,0x1f,0x63,0x79,0xea,0x06,0x6b,0x5f,0x38,0x20,0x5d,0xfb,0x82,0x2b,0xeb,
0x85,0xb3,0x9b,0x80,0x28,0xeb,0xdd,0x45,0xd2,0x9e,0x31,0x79,0x8f,0x7a,0x2a,0x7f,0x9e,0x2e,0x8a,
0x66,0xfd,0x12,0x7e,0xfa,0x4c,0x15,0xfd,0x86,0xa7,0xe7,0xe2,0xbc,0x0a,0xd6,0xaf,0x24,0xd8,0x58,
0x90,0x60,0xdf,0x16,0xfa,0xb9,0x24,0xb1,0xae,0xe9,0xe5,0x79,0x97,0x17,0x30,0xd5,0xa0,0xe2,0x10,
0x6b,0x91,0xe9,0x89,0x50,0xce,0x1c,0x53,0x4e,0xf4,0xc4,0x1e,0x60,0xa2,0xbf,0x6a,0xa1,0x3a,0x68,
0xb7,0x73,0xf8,0x26,0x38,0xf1,0xe6,0x03,0x09,0xa2,0x10,0x54,0x49,0xe5,0x89,0xb3,0x13,0x6c,0xba,
0xda,0x9c,0x5d,0x29,0xf7,0x86,0x57,0xac,0xb0,0xf5,0x0a,0x80,0xb2,0x46,0xab,0xdd,0x2e,0xab,0x95,
0xce,0x12,0x31,0x81,0x9a,0x64,0x5e,0xd1,0x55,0xaf,0x7a,0x4e,0x0f,0x94,0x8c,0xba,0x46,0x0c,0x5b,
0x1a,0x05,0x13,0x9b,0x71,0xb1,0x36,0x95,0xa3,0x4b,0x92,0x4d,0xd6,0xe4,0x28,0xb0,0xf6,0x90,0xca,
0xd5,0x7a,0xc9,0x7e,0xec,0x2a,0x1b,0x55,0xeb,0xdf,0x85,0x43,0xe7,0xf8,0xdd,0x77,0xec,0x65,0xc4,
0x20,0xf7,0x0f,0xa7,0xc1,0x46,0x4b,0xed,0x73,0xe0,0x42,0x7d,0xef,0xf9,0xef,0xdd,0x2a,0x30,0x23,
0x62,0x67,0x3a,0x54,0xf8,0xbe,0x3f,0x35,0xa6,0xdb,0xc6,0x4a,0x23,0x8a,0x21,0xde,0x6e,0xd8,0x79,
0xa4,0xeb,0x68,0x09,0x3b,0xd4,0xc1,0xad,0x61,0x00,0xb9,0x2e,0xa4,0x02,0x8a,0xae,0x24,0xfd,0x65,
0x89,0x9f,0x83,0x1a,0x22,0xd9,0x40,0xf1,0x6a,0x2f,0x52,0xad,0xfc,0x9e,0x94,0x6e,0xea,0x39,0xe6,
0x78,0xad,0xc3,0x5d,0x3d,0x75,0x9b,0xdb,0x85,0x72,0xc7,0x25,0x38,0xd1,0x31,0xc0,0x09,0x1f,0xd8,
0x4b,0x85,0x41,0xd9,0xcd,0xf4,0xa7,0x47,0x02,0x37,0xc6,0x7d,0x6c,0x74,0xe4,0x77,0x77,0xa7,0xd1,
0x12,0x72,0x1e,0x5e,0xb7,0x65,0x72,0xb8,0x47,0x2d,0xdf,0xaf,0xcf,0x16,0x08,0xde,0x4e,0xed,0x5a,
0x94,0xac,0xb8,0xb0,0x84,0xef,0xc4,0xbd,0x81,0xf5,0xcc,0x9a,0x85,0xd8,0xc7,0xd7,0xd4,0x14,0x94,
0xa9,0x89,0xb3,0xca,0x99,0x89,0xb2,0x79,0x89,0x35,0xaa,0xd9,0xdc,0x04,0xe6,0x24,0xf8,0x2c,0x38,
0xa6,0x59,0xd3,0x29,0x57,0x5c,0xc6,0x09,0x59,0xb0,0x87,0xcd,0x6d,0xa5,0x6c,0x3a,0xfc,0x74,0xdb,
0x2c,0xaa,0x86,0x42,0x13,0x6f,0x5e,0xa7,0x11,0xd6,0x28,0x93,0x42,0xae,0xa1,0x52,0x94,0x88,0x3b,
0x16,0xbd,0x65,0xb5,0x90,0x90,0x3e,0xde,0x42,0x81,0xd5,0x4b,0x18,0x2a,0x14,0xc8,0x29,0x65,0xca,
0x4b,0x97,0x30,0x28,0xf3,0x7e,0x52,0x77,0xbd,0x7b,0xd4,0xdd,0x49,0xcd,0x29,0xf3,0x2d,0x94,0x87,
0x13,0x17,0x9b,0x2a,0xce,0x8b,0x67,0x14,0x12,0x51,0x88,0xbb,0x19,0x50,0xd4,0xd5,0x4b,0x45,0xa2,
0xa1,0xf6,0x70,0x0a,0xb5,0x1c,0x64,0xbd,0x96,0x87,0xe9,0x30,0x9c,0x77,0xb1,0xfb,0x53,0xb3,0x17,
0x55,0x8f,0xf1,0x9c,0x1f,0x61,0x5b,0xb2,0x8f,0x54,0xee,0x5f,0x65,0x3d,0xbc,0x6c,0x2f,0xb8,0x34,
0x8d,0xe2,0x1e,0x71,0x31,0x47,0x5e,0x2f,0x19,0x66,0xbb,0xbd,0xc5,0x10,0x04,0xbd,0x94,0xdd,0xca,
0x5f,0xab,0xd8,0xa7,0xe7,0x0b,0x67,0x74,0xc1,0x62,0xa3,0xf1,0x85,0x5d,0xb6,0x19,0xb6,0x84,0xf1,
0xae,0x8e,0xf3,0x42,0x61,0xa5,0xf7,0x98,0x75,0xd0,0xc7,0x91,0xb7,0xc0,0xaa,0x43,0xfc,0xbb,0xc0,
```

Anexo 42:Código de camera_index.h (6)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

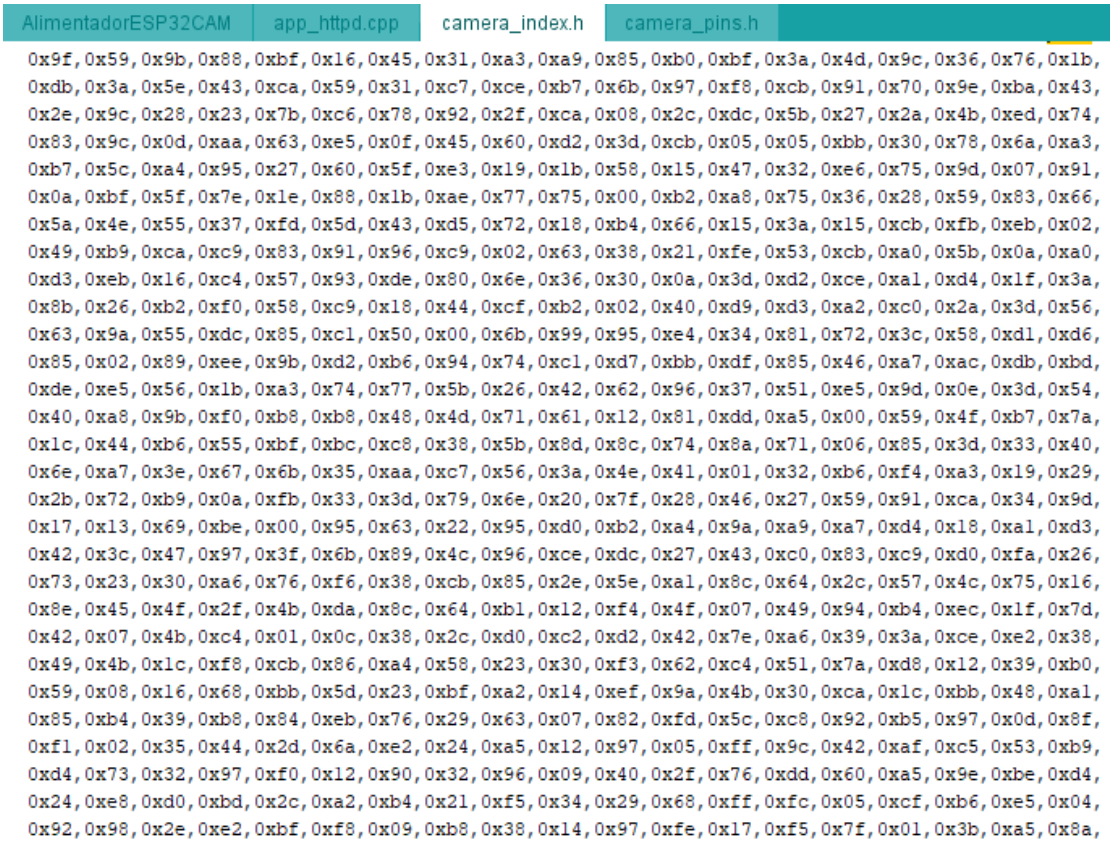
```
0x12,0xfb,0x6b,0x93,0x0e,0xce,0x6e,0xbc,0xba,0xc1,0xc5,0xf7,0x1e,0x1c,0x07,0x42,0x68,0x1d,0xbe,
0xfc,0xe1,0x2d,0xdd,0x0c,0x44,0xef,0x85,0x8e,0x4b,0xdc,0xc3,0x23,0x6b,0xb2,0x0c,0x78,0xbf,0xbe,
0x85,0x25,0x44,0x41,0xc2,0xcf,0x76,0xbe,0x71,0x22,0x6c,0x40,0x8f,0xc9,0x9b,0x30,0x4e,0xac,0x0b,
0x4b,0x62,0xc4,0x41,0x15,0x6c,0xc3,0x48,0x07,0xe7,0x42,0x23,0xde,0x8a,0x92,0x9c,0xd9,0x7f,0x46,
0x3e,0x8a,0x4a,0xa8,0xaf,0xad,0xc3,0xc1,0xd3,0xde,0x21,0x35,0x05,0x68,0x01,0x37,0xa0,0x01,0x82,
0x02,0x30,0xf2,0x8b,0xa1,0xd8,0xc0,0x4d,0x7c,0x6c,0xac,0x87,0xc4,0x29,0x81,0x94,0xda,0xd6,0x21,
0x57,0xd3,0x21,0x3d,0xcc,0x80,0xee,0xdb,0xe6,0x90,0xf1,0x2c,0x5c,0x99,0x20,0x23,0x32,0xc7,0x91,
0x8b,0x05,0x60,0x09,0x2d,0x7a,0xe8,0x95,0x55,0xa7,0x3d,0x79,0x06,0xcf,0x0a,0xc8,0xb3,0x12,0x2f,
0x70,0xd8,0xc2,0x92,0xee,0x2a,0x57,0xd0,0xf2,0x54,0xa0,0x0e,0x59,0x46,0xc4,0x13,0xc7,0x8f,0x0b,
0x98,0x97,0x0b,0x17,0xa7,0x3a,0xfc,0x4c,0xa3,0x1e,0x0a,0xb4,0x88,0x7f,0xc4,0x43,0xe0,0x91,0x78,
0xf2,0x1e,0x78,0x13,0x1c,0xc2,0x2d,0x6b,0x55,0x6f,0x03,0x22,0xff,0xf3,0xc2,0x0a,0x96,0x18,0x96,
0x78,0xc6,0x58,0xb0,0x06,0xb9,0xa7,0x0c,0xda,0xc7,0x90,0x8b,0xf8,0x5f,0x08,0x56,0x27,0xbb,0xe9,
0x4d,0x68,0xc5,0x1d,0xf6,0x2f,0x15,0x17,0xc0,0x71,0x98,0xe6,0x58,0x87,0xd9,0xe1,0xdf,0x2a,0x10,
0x93,0x43,0x47,0x8c,0x3c,0x8a,0xe7,0x8c,0x68,0x3c,0x38,0x38,0x60,0x57,0xe2,0x6e,0x56,0x0c,0x8f,
0xb2,0x07,0x9f,0xf1,0x40,0x39,0x61,0x61,0x1d,0x77,0x01,0x47,0x8a,0x5c,0xc1,0xc0,0xa3,0x0e,0xa5,
0x3c,0x27,0x81,0x47,0x8f,0xf2,0xd8,0x0e,0xc0,0x0e,0x83,0xca,0x38,0xe1,0xe5,0xe1,0x19,0xf0,0x3c,
0xb0,0x2d,0xce,0xdd,0x10,0x24,0x79,0x93,0xd6,0x41,0x4e,0xf0,0x92,0xc6,0x09,0x15,0x11,0x8e,0xcc,
0xa0,0x02,0x62,0xab,0x4e,0xe5,0xa3,0x94,0x79,0x64,0x45,0xb0,0xfa,0x16,0x11,0xeb,0xd1,0x1e,0x43,
0xfe,0xd4,0x98,0xb3,0x1b,0xa2,0x7c,0x56,0x95,0x8a,0x11,0xcb,0x8a,0x14,0x8c,0x94,0xb1,0x02,0xdd,
0xec,0x68,0x0f,0x8a,0x8f,0xae,0xcc,0x10,0xab,0x43,0xd4,0x73,0x40,0x58,0xe5,0x40,0x43,0x57,0x8f,
0x64,0xf7,0x0b,0xa2,0x96,0x05,0x4b,0x90,0xb0,0x0a,0xd6,0x91,0x18,0x29,0x4f,0x97,0xe6,0x6b,0x04,
0xc2,0xd0,0xad,0x46,0x54,0x14,0xac,0x56,0x5c,0x8a,0x32,0xdd,0xae,0x16,0x59,0xa1,0x47,0xad,0xa0,
0xec,0x76,0x53,0xa4,0xdc,0x19,0x5b,0x7c,0xbc,0xee,0x05,0x6b,0xc5,0x68,0x05,0xc2,0xcf,0xf2,0xf7,
0x53,0xc2,0x85,0xd1,0x08,0xc3,0x91,0x61,0x50,0xb5,0x05,0xea,0x03,0xa9,0xb4,0xa9,0x9b,0x64,0xc6,
0x26,0x8e,0x53,0x4a,0x7d,0x24,0x53,0xc9,0x18,0xf1,0x4f,0xf1,0x96,0x41,0x81,0x7f,0xd5,0x4d,0x40,
0x77,0x0f,0x44,0x66,0x07,0x24,0x8d,0x10,0x46,0xd3,0x53,0xaf,0x39,0x1e,0x96,0xa9,0x48,0x24,0xfc,
0x1e,0x5f,0x5c,0xde,0xc6,0x31,0x17,0x7a,0xec,0xaa,0xa3,0xe8,0x70,0xf2,0x46,0xbe,0x88,0x74,0x39,
0x9a,0x7b,0x89,0x06,0xe1,0x21,0x42,0xb8,0x0e,0x97,0x48,0xbd,0x32,0x00,0x9c,0xc4,0xbf,0x8c,0xf8,
0xe1,0xe0,0xc2,0x13,0x79,0x34,0xfb,0xcf,0x92,0x44,0x77,0x40,0xf4,0xf1,0xab,0x4f,0x69,0xdb,0xf0,
```

Anexo 43: Código de camera_index.h (7)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0xb9,0xcb,0x36,0x84,0x86,0xfe,0x33,0xb4,0x1e,0x17,0x5f,0x7d,0x62,0xca,0xfe,0xfc,0x08,0x55,0xe2,
0x07,0xab,0xf8,0xf3,0x47,0x8e,0x62,0x42,0x8f,0xf7,0x6f,0x31,0x14,0xa9,0xde,0x3a,0xc9,0x8c,0x04,
0xad,0x88,0xc4,0x0b,0xa0,0x07,0x85,0x69,0x10,0x4c,0x6b,0x0c,0x7d,0x82,0x66,0x6a,0xda,0xfa,0x18,
0x11,0xc0,0x81,0x80,0x24,0xb4,0xbe,0xfa,0xc4,0x50,0x7c,0xc6,0xb1,0xf8,0x01,0xb2,0x1e,0xe2,0x1e,
0xa1,0xcd,0xc2,0xee,0x47,0x9c,0x7b,0xf2,0xd5,0xa7,0x14,0x55,0x87,0xdf,0xfa,0xfc,0x51,0x5a,0x88,
0x6c,0x48,0xd2,0xf6,0x8f,0x3d,0xe8,0x30,0x5c,0xd7,0x4c,0x0b,0x61,0xf4,0xdc,0xf7,0x5b,0x87,0xfc,
0x78,0x18,0x11,0xdf,0x3b,0xc8,0xe1,0xae,0x1c,0x90,0xad,0x36,0x0c,0x2c,0x66,0x85,0xc1,0xd8,0xf7,
0x70,0x32,0x21,0x82,0xba,0x12,0xbd,0xa5,0xff,0xa1,0x04,0x3f,0xee,0xef,0x1d,0xf6,0x74,0x14,0xcc,
0xf4,0x31,0x25,0xa3,0xdb,0x85,0x94,0x1d,0xda,0xff,0x66,0xe1,0x8c,0xeb,0x88,0x9e,0x0b,0xc5,0xc5,
0x94,0x93,0x30,0x67,0x46,0xf0,0xc2,0x65,0x96,0xb5,0xf4,0x29,0xcb,0x99,0xd9,0x72,0xed,0xe1,0x4b,
0xc8,0xe2,0xdf,0x71,0x18,0xb4,0x38,0x30,0x13,0xc3,0x3a,0x0e,0x5a,0x81,0x82,0x20,0x27,0xa2,0x32,
0x31,0xe5,0xf3,0x75,0x21,0x2f,0x83,0xcc,0xb2,0xd8,0xcc,0x42,0x37,0x6b,0x0a,0x59,0xbd,0xbf,0x31,
0x93,0xf9,0x17,0xb2,0x17,0xda,0x7c,0x2a,0x51,0x49,0x92,0x2c,0x5b,0x53,0xfa,0xc7,0x48,0x6a,0x0a,
0x83,0xc9,0x86,0x2b,0x9f,0xd0,0xcb,0x17,0x77,0xdf,0xa1,0xd9,0xe7,0xc9,0x0b,0xa3,0x25,0x03,0xa0,
0x49,0x12,0xcf,0x1b,0x2b,0x21,0xb3,0x1c,0x53,0xc1,0xc1,0x66,0x33,0x78,0xbc,0x31,0x61,0x90,0x13,
0x1f,0x39,0x50,0x8a,0xb5,0x1a,0x36,0x37,0xe1,0xc1,0xe0,0xbb,0xd4,0xad,0x58,0x66,0x92,0x45,0x3b,
0x13,0x06,0x65,0x8a,0x43,0xa9,0x9f,0x99,0x72,0x35,0xb0,0x9a,0x17,0x2b,0xd0,0xb4,0x43,0x41,0x3b,
0x16,0x35,0x6a,0x97,0xdd,0x0f,0x80,0x2b,0xdc,0x87,0x0b,0x3e,0x6b,0x53,0xf0,0x92,0x95,0x17,0xb8,
0xe1,0x0a,0x3e,0x1a,0x2e,0x5a,0xa2,0x7d,0x56,0x25,0x85,0x43,0xdc,0xa1,0x81,0x37,0x3f,0xbd,0xfd,
0x9e,0xc6,0x2c,0x75,0xf6,0x87,0x86,0x2f,0x25,0xb9,0x62,0x7f,0xe5,0xa0,0xad,0x81,0xea,0xbd,0x83,
0x7c,0x9d,0xc7,0x2a,0x99,0xd3,0x52,0x57,0xa2,0x97,0x1f,0x79,0x9d,0xb4,0xf5,0xca,0x59,0x08,0xb7,
0x38,0x13,0x2d,0xe1,0xa2,0x48,0x0a,0x1c,0xf9,0x79,0x92,0xc0,0xde,0x2d,0xee,0x09,0x31,0x0d,0x52,
0x62,0x4e,0x0d,0x8f,0x15,0xeb,0x29,0x89,0x19,0x99,0x98,0x84,0x93,0xe6,0x89,0x57,0x02,0xad,0xb3,
0x80,0x63,0x93,0x67,0x1f,0xc6,0x23,0xc4,0xd6,0x97,0x70,0x9d,0x0e,0x3a,0x93,0xad,0xc7,0x08,0xb0,
0xe5,0xec,0x70,0x71,0x65,0x76,0x50,0x97,0x08,0x16,0xc5,0xf4,0xd8,0x72,0xf2,0xd1,0xa3,0x53,0xcd,
0xff,0x8a,0x8f,0xb3,0xa1,0x44,0x99,0x60,0x69,0x4e,0x5a,0x10,0x2d,0x4f,0x91,0x72,0x08,0xb2,0xf8,
0xb4,0x46,0x6c,0x21,0x0b,0x52,0xec,0x22,0x2d,0xa0,0x68,0x4b,0x75,0x29,0x2d,0xf9,0x28,0x53,0x4c,
0x23,0x0b,0x39,0x07,0x4a,0xb0,0x58,0x4e,0xc3,0xb5,0x34,0x7d,0x83,0x28,0xd2,0x76,0xcc,0x7e,0x85,
0xd2,0xcc,0xaf,0x10,0x8d,0x44,0xf7,0x9d,0xc1,0x60,0x18,0x92,0x13,0x9a,0xab,0x36,0xc3,0x2d,0x65,
```

Anexo 44: Código de camera_index.h (8)



Anexo 45: Código de camera_index.h (9)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
  0xdb,0xa2,0x7a,0x00,0x00
};


//File: index_ov3660.html.gz, Size: 4408
#define index_ov3660_html_gz_len 4408
const uint8_t index_ov3660_html_gz[] = {
 0x1F, 0x8B, 0x08, 0x08, 0x28, 0x5C, 0xAE, 0x5C, 0x00, 0x03, 0x69, 0x6E, 0x64, 0x65, 0x78, 0x5F,
 0x6F, 0x76, 0x33, 0x36, 0x36, 0x30, 0x2E, 0x68, 0x74, 0x6D, 0x6C, 0x00, 0xE5, 0x5D, 0xEB, 0x92,
 0xD3, 0xC6, 0x12, 0xFE, 0xCF, 0x53, 0x08, 0x41, 0x58, 0x6F, 0x65, 0xED, 0xF5, 0x6D, 0xCD, 0xE2,
 0xD8, 0xE6, 0xC0, 0xB2, 0x84, 0x54, 0x01, 0x49, 0x20, 0x21, 0xA9, 0x4A, 0xA5, 0x60, 0x2C, 0x8D,
 0xED, 0x09, 0xB2, 0xE4, 0x48, 0x23, 0x7B, 0x37, 0xD4, 0x3E, 0xC7, 0x79, 0xA0, 0xF3, 0x62, 0xA7,
 0xE7, 0x22, 0x69, 0x24, 0x8F, 0x2E, 0xB6, 0x59, 0x9B, 0xC3, 0x31, 0x55, 0x20, 0x5B, 0xD3, 0x3D,
 0xDD, 0xFD, 0xF5, 0x6D, 0x46, 0x17, 0x06, 0x77, 0x6D, 0xCF, 0xA2, 0xD7, 0x0B, 0x6C, 0xCC, 0xE8,
 0xDC, 0x19, 0xDD, 0x19, 0x88, 0x7F, 0x0C, 0xF8, 0x0C, 0x66, 0x18, 0xD9, 0xE2, 0x90, 0x7F, 0x9D,
 0x63, 0x8A, 0x0C, 0x6B, 0x86, 0xFC, 0x00, 0xD3, 0xA1, 0x19, 0xD2, 0x49, 0xFD, 0xDC, 0xCC, 0x9E,
 0x76, 0xD1, 0x1C, 0x0F, 0xCD, 0x25, 0xC1, 0xAB, 0x85, 0xE7, 0x53, 0xD3, 0xB0, 0x3C, 0x97, 0x62,
 0x17, 0x86, 0xAF, 0x88, 0x4D, 0x67, 0x43, 0x1B, 0x2F, 0x89, 0x85, 0xEB, 0xFC, 0xCB, 0x09, 0x71,
 0x09, 0x25, 0xC8, 0xA9, 0x07, 0x16, 0x72, 0xF0, 0xB0, 0xA5, 0xF2, 0xA2, 0x84, 0x3A, 0x78, 0x74,
 0xF9, 0xF6, 0xA7, 0x4E, 0xDB, 0xF8, 0xF1, 0x5D, 0xA7, 0xD7, 0x6B, 0x0E, 0x4E, 0xC5, 0x6F, 0xC9,
 0x98, 0x80, 0x5E, 0xAB, 0xDF, 0xD9, 0x67, 0xEC, 0xD9, 0xD7, 0xC6, 0xA7, 0xD4, 0x4F, 0xEC, 0x33,
 0x01, 0x21, 0xEA, 0x13, 0x34, 0x27, 0xCE, 0x75, 0xDF, 0x78, 0xE2, 0xC3, 0x9C, 0x27, 0x2F, 0xB0,
 0xB3, 0xC4, 0x94, 0x58, 0xE8, 0x24, 0x40, 0x6E, 0x50, 0x0F, 0xB0, 0x4F, 0x26, 0xDF, 0xAD, 0x11,
 0x8E, 0x91, 0xF5, 0x71, 0xEA, 0x7B, 0xA1, 0x6B, 0xF7, 0x8D, 0x7B, 0xAD, 0x73, 0xF6, 0x67, 0x7D,
 0x90, 0xE5, 0x39, 0x9E, 0x0F, 0xE7, 0x2F, 0x9F, 0xB3, 0x3F, 0xEB, 0xE7, 0xF9, 0xEC, 0x01, 0xF9,
 0x07, 0xF7, 0x8D, 0x56, 0x6F, 0x71, 0x95, 0x3A, 0x7F, 0x73, 0x27, 0xF5, 0x75, 0xD6, 0xCE, 0x93,
 0x5E, 0xD2, 0x9F, 0x17, 0xD3, 0x07, 0xD8, 0xA2, 0xC4, 0x73, 0x1B, 0x73, 0x44, 0x5C, 0x0D, 0x27,
 0x9B, 0x04, 0x0B, 0x07, 0x81, 0x0D, 0x26, 0x0E, 0x2E, 0xE4, 0x73, 0x6F, 0x8E, 0xDD, 0xF0, 0xA4,
 0x84, 0x1B, 0x63, 0x52, 0xB7, 0x89, 0x2F, 0x46, 0xF5, 0x99, 0x1D, 0xC2, 0xB9, 0x5B, 0xCA, 0xB6,
```

Anexo 46: Código de camera_index.h (10)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
 0x48, 0x2E, 0xD7, 0x73, 0xB1, 0xC6, 0x80, 0x6C, 0xA2, 0x95, 0x8F, 0x16, 0x6C, 0x00, 0xFB, 0x77,
 0x7D, 0xC8, 0x9C, 0xB8, 0xC2, 0xA9, 0xFA, 0x46, 0xA7, 0xDB, 0x5C, 0x5C, 0x95, 0x40, 0xD9, 0xE9,
 0xB1, 0x3F, 0xEB, 0x83, 0x16, 0xC8, 0xB6, 0x89, 0x3B, 0xED, 0x1B, 0xE7, 0x5A, 0x16, 0x9E, 0x6F,
 0x63, 0xBF, 0xEE, 0x23, 0x9B, 0x84, 0x41, 0xDF, 0xE8, 0xEA, 0xC6, 0xCC, 0x91, 0x3F, 0x05, 0x59,
 0xA8, 0x07, 0xC2, 0xD6, 0x5B, 0x5A, 0x49, 0xE4, 0x10, 0x9F, 0x4C, 0x67, 0x14, 0x20, 0x5D, 0x1B,
 0x93, 0x35, 0x9A, 0x0C, 0xA1, 0x32, 0x3C, 0x0B, 0xED, 0xA6, 0xB7, 0x1A, 0x72, 0xC8, 0xD4, 0xAD,
 0x13, 0x8A, 0xE7, 0xA0, 0x4E, 0x40, 0x7D, 0x4C, 0xAD, 0x59, 0x91, 0x28, 0x13, 0x32, 0x0D, 0x7D,
 0xAC, 0x11, 0x24, 0xB6, 0x5B, 0x81, 0xC2, 0x70, 0x72, 0xFD, 0x54, 0x7D, 0x85, 0xC7, 0x1F, 0x09,
 0xAD, 0x4B, 0x9B, 0x8C, 0xF1, 0xC4, 0xF3, 0xB1, 0x76, 0x64, 0x34, 0xC2, 0xF1, 0xAC, 0x8F, 0xF5,
 0x80, 0x22, 0x9F, 0x56, 0x61, 0x88, 0x26, 0x14, 0xFB, 0xE5, 0xFC, 0x30, 0xF3, 0x8A, 0x72, 0x6E,
 0xF9, 0xD3, 0xCA, 0x01, 0xC4, 0x75, 0x88, 0x8B, 0xAB, 0x8B, 0x97, 0x37, 0x6F, 0x9A, 0x9D, 0x18,
 0x55, 0x01, 0x18, 0x32, 0x9F, 0x16, 0x79, 0x09, 0xD7, 0x75, 0x7D, 0x32, 0x19, 0x37, 0xAD, 0x66,
 0xF3, 0x9B, 0xF5, 0x93, 0x33, 0x2C, 0xDC, 0x14, 0x85, 0xD4, 0xDB, 0x3D, 0x22, 0xD6, 0xC2, 0x2A,
 0xA3, 0xC7, 0xBF, 0xE6, 0xD8, 0x26, 0xC8, 0xA8, 0x29, 0xE1, 0x7C, 0xDE, 0x04, 0x9F, 0x3A, 0x36,
 0x90, 0x6B, 0x1B, 0x35, 0xCF, 0x27, 0x10, 0x08, 0x88, 0xA7, 0x1B, 0x07, 0x7E, 0x81, 0xC2, 0xB1,
 0xC0, 0xC7, 0x1A, 0x95, 0x0B, 0x62, 0x46, 0xB5, 0x88, 0x3E, 0x6C, 0xD8, 0xA7, 0x42, 0xCA, 0x61,
 0x9F, 0xD2, 0x00, 0xD2, 0xE8, 0xC8, 0xD9, 0x17, 0xE1, 0xA5, 0x4A, 0x98, 0x87, 0x19, 0xFB, 0xCC,
 0xD1, 0x55, 0xBD, 0x10, 0xBB, 0x68, 0x50, 0x84, 0x21, 0x94, 0x59, 0xAB, 0x06, 0x43, 0x97, 0x33,
 0xA3, 0x6E, 0xB0, 0x2C, 0x79, 0xAC, 0xA7, 0x91, 0x4C, 0xF5, 0x90, 0xB3, 0x8F, 0xEA, 0x14, 0x1B,
 0xA8, 0xAB, 0x57, 0x35, 0xC9, 0x1D, 0xE2, 0x8F, 0xCE, 0x87, 0x84, 0x26, 0xB9, 0x59, 0x84, 0x7D,
 0xAA, 0x67, 0x92, 0x84, 0x59, 0x69, 0x36, 0xD1, 0x30, 0xCE, 0xCF, 0x28, 0x6B, 0x7C, 0xF3, 0xA2,
 0x5B, 0xC3, 0xB5, 0x58, 0x84, 0xAA, 0xD9, 0x45, 0xC3, 0xB8, 0x48, 0x86, 0xD2, 0x2C, 0xC3, 0x3E,
 0x37, 0x15, 0xFA, 0x8D, 0x7B, 0xE3, 0x90, 0x52, 0xCF, 0x0D, 0x76, 0x2A, 0x51, 0x79, 0x71, 0xF6,
 0x57, 0x18, 0x50, 0x32, 0xB9, 0xAE, 0xCB, 0x90, 0x86, 0x38, 0x5B, 0x20, 0x68, 0x21, 0xC7, 0x98,
 0xAE, 0x30, 0x2E, 0x6E, 0x37, 0x5C, 0xB4, 0x84, 0xBC, 0x33, 0x9D, 0x3A, 0x3A, 0xDF, 0xB3, 0x42,
 0x3F, 0x60, 0x7D, 0xDB, 0xC2, 0x23, 0xC0, 0xD8, 0x5F, 0x9F, 0x38, 0x1D, 0x83, 0x15, 0x27, 0xAA,
 0x5B, 0x63, 0xCD, 0x5C, 0x5E, 0x48, 0x99, 0x8D, 0xB5, 0x48, 0x78, 0xA0, 0x0E, 0xA1, 0xD7, 0xDA,
 0x73, 0x32, 0x12, 0x35, 0x67, 0xA2, 0x10, 0x2C, 0x2C, 0x0B, 0x69, 0xB9, 0xFA, 0xD6, 0x0C, 0x5B,
 0x1F, 0xB1, 0xFD, 0x6D, 0x69, 0x1B, 0x56, 0xD6, 0x1E, 0x36, 0x88, 0xBB, 0x08, 0x69, 0x9D, 0xB5,
```

Anexo 47: Código de camera_index.h (11)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0x53, 0x8B, 0x5B, 0xC1, 0x9C, 0x3B, 0x64, 0xA4, 0x62, 0xBB, 0x5D, 0xD4, 0x54, 0x9C, 0x2D, 0xAE,
0x8A, 0x8D, 0xA0, 0x0A, 0x3B, 0x72, 0xD0, 0x18, 0x3B, 0x45, 0x22, 0xCB, 0x60, 0xC8, 0x49, 0xBB,
0x32, 0x57, 0xE5, 0xF7, 0x6E, 0x5C, 0xB2, 0xA4, 0x78, 0x75, 0x1F, 0x7E, 0x53, 0xD9, 0x8E, 0xFC,
0xF8, 0x24, 0xF5, 0x53, 0x80, 0x1D, 0x08, 0xB0, 0xBC, 0xD6, 0x1B, 0xC6, 0xAC, 0x40, 0x86, 0xC2,
0x09, 0x7C, 0xE4, 0x4E, 0x31, 0xE4, 0x82, 0xAB, 0x93, 0xE8, 0xB0, 0x78, 0x61, 0x50, 0x49, 0x7D,
0x96, 0xAA, 0xCF, 0x8A, 0x17, 0x22, 0x22, 0x21, 0x6C, 0xD1, 0x8C, 0x28, 0xB0, 0x16, 0xCE, 0xDF,
0xD2, 0x3A, 0x85, 0xE8, 0x47, 0xB4, 0x01, 0x93, 0x76, 0x29, 0x6D, 0x7F, 0x5F, 0x9A, 0x11, 0xA2,
0x95, 0xDE, 0x64, 0x52, 0xB6, 0x56, 0x9C, 0x4C, 0x3A, 0xCD, 0x4E, 0xB7, 0xB4, 0x61, 0xD2, 0x6A,
0x99, 0x59, 0x2F, 0x6A, 0x32, 0x46, 0x9C, 0x4D, 0xCA, 0x21, 0xE8, 0xCF, 0xBC, 0x25, 0xF6, 0x35,
0x40, 0x64, 0xC4, 0xED, 0x3E, 0xEA, 0xDA, 0x15, 0xB8, 0x21, 0xC8, 0xF7, 0x4B, 0x5D, 0x36, 0x4D,
0xB3, 0x6B, 0xB7, 0xAC, 0x76, 0xA1, 0x63, 0x0A, 0x76, 0x0D, 0xF0, 0x06, 0x34, 0x76, 0xB0, 0x5D,
0x90, 0x9E, 0x6D, 0x3C, 0x41, 0xA1, 0x43, 0x4B, 0xEC, 0x8D, 0x9A, 0xEC, 0x4F, 0xD1, 0x8C, 0x3C,
0xAE, 0xFE, 0x60, 0x1B, 0x1D, 0x43, 0x1E, 0x09, 0x7F, 0x6A, 0xE6, 0x8C, 0x6A, 0x27, 0x5A, 0x2C,
0x30, 0x82, 0x51, 0x16, 0xCE, 0x5B, 0x92, 0x56, 0xEA, 0x99, 0xF5, 0x89, 0xAB, 0xD2, 0x42, 0xB4,
0xD4, 0x15, 0xE3, 0x6E, 0x68, 0x23, 0x9D, 0xFB, 0x13, 0xCF, 0x0A, 0x75, 0x65, 0xBA, 0x9A, 0x4B,
0xAD, 0xF3, 0xEB, 0x47, 0x26, 0x0B, 0x1C, 0xC2, 0x1D, 0x3B, 0x74, 0x5D, 0x86, 0x68, 0x9D, 0xFA,
0xA0, 0xA6, 0x66, 0xA2, 0x6A, 0x86, 0xDB, 0x2A, 0x3A, 0x53, 0x86, 0xCD, 0xDB, 0x8C, 0xC9, 0x04,
0xA0, 0x26, 0x51, 0xC4, 0x39, 0xC4, 0x08, 0x3C, 0x50, 0x2A, 0x62, 0xB5, 0x9B, 0x5D, 0xE8, 0x2C,
0x9C, 0xEB, 0x1A, 0x83, 0x68, 0xB2, 0x16, 0x54, 0x31, 0x31, 0x9D, 0x3F, 0x1D, 0xA3, 0x5A, 0xF3,
0xA4, 0x79, 0xD2, 0x81, 0xBF, 0x34, 0x0D, 0x7A, 0xB1, 0x73, 0x49, 0xF3, 0xE6, 0x78, 0x5E, 0x26,
0xF9, 0x94, 0xEF, 0x93, 0xE4, 0xA5, 0xB1, 0x52, 0x2C, 0xAA, 0x47, 0x52, 0x7A, 0xC3, 0xA4, 0xD5,
0x28, 0x29, 0x2C, 0x39, 0x2E, 0xBD, 0xB9, 0x23, 0x6A, 0xBC, 0x65, 0x53, 0x88, 0xE7, 0xDE, 0x3F,
0x75, 0x51, 0x55, 0xFF, 0xEF, 0xBD, 0x5D, 0x31, 0xC5, 0x57, 0xED, 0xE9, 0x1B, 0xDB, 0x25, 0x38,
0xB4, 0x6F, 0x34, 0xF3, 0x51, 0xAF, 0xCB, 0x7E, 0x06, 0x24, 0x74, 0x61, 0x51, 0xE5, 0xC3, 0xEA,
0x2A, 0xB7, 0xE7, 0x51, 0xC6, 0x6C, 0x61, 0x83, 0x09, 0x71, 0x9C, 0xBA, 0xE3, 0xAD, 0xCA, 0x3B,
0x91, 0x62, 0x4F, 0x5E, 0xF3, 0xD3, 0x72, 0x97, 0xDF, 0x56, 0xDA, 0x10, 0x32, 0xD7, 0xFF, 0x84,
0xB4, 0x5F, 0x77, 0xC0, 0x15, 0x86, 0xC6, 0x76, 0x85, 0x62, 0x0B, 0x7F, 0xDC, 0x6D, 0xA2, 0x4A,
0xAE, 0x24, 0x3A, 0xC1, 0xC2, 0xC5, 0x5C, 0xB0, 0x22, 0xD4, 0x9A, 0x6D, 0xB1, 0xA8, 0x5A, 0x78,
0x01, 0x11, 0xD7, 0x68, 0x7C, 0xEC, 0x20, 0xD6, 0xC1, 0x6F, 0xB5, 0xE4, 0x2E, 0x5D, 0x98, 0xA8,
```

Anexo 48: Código de camera_index.h (12)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0xE4, 0x55, 0x34, 0xE1, 0xA6, 0xFB, 0x72, 0xB6, 0x4B, 0x1A, 0xA2, 0x77, 0xC8, 0xCF, 0xD5, 0x7A,
0xB7, 0x2E, 0x69, 0xF7, 0xD3, 0x91, 0xA1, 0x1F, 0xB4, 0x41, 0x46, 0x8F, 0x92, 0xF6, 0xD4, 0xC7,
0xD7, 0x15, 0x94, 0x39, 0x91, 0xFF, 0xF6, 0xC5, 0x86, 0xE8, 0xF6, 0x6B, 0x7F, 0x5E, 0x00, 0xA4,
0x17, 0x35, 0xBA, 0x41, 0x85, 0xA9, 0xF3, 0xA7, 0xAC, 0xE2, 0x8F, 0xF1, 0x76, 0x9F, 0x69, 0x56,
0x48, 0x37, 0x05, 0x25, 0x54, 0xEF, 0xAA, 0x51, 0xF5, 0xD5, 0x9E, 0x74, 0xF0, 0x84, 0xE6, 0x5C,
0xCD, 0xE0, 0x7D, 0x6A, 0xA7, 0x38, 0xBB, 0xD5, 0x95, 0x7D, 0x82, 0xD2, 0xCC, 0x11, 0xEF, 0xCA,
0xE5, 0x7B, 0x9F, 0x96, 0x33, 0xCB, 0x9E, 0x1B, 0x33, 0xCF, 0x87, 0x24, 0x6A, 0x9F, 0x39, 0xCC,
0x30, 0x66, 0x2E, 0x4B, 0x3E, 0xC0, 0x83, 0x7F, 0xAF, 0xB5, 0x7B, 0xDA, 0x8B, 0x05, 0x05, 0x83,
0x8B, 0x44, 0xCB, 0xDD, 0xD6, 0x5A, 0x2F, 0x59, 0xB9, 0x0B, 0x64, 0x35, 0x17, 0x69, 0x81, 0x2A,
0x8E, 0xCA, 0xA2, 0x0C, 0xB3, 0xBE, 0x47, 0x53, 0xE8, 0xEC, 0x64, 0x8E, 0xA0, 0xED, 0x65, 0xEE,
0x8A, 0x80, 0xA3, 0x0E, 0xBF, 0x2A, 0xEE, 0xAE, 0x6C, 0x1A, 0xB6, 0x7A, 0xCD, 0x92, 0x29, 0x2D,
0xC7, 0x0B, 0x8A, 0xE3, 0x0A, 0x8D, 0xC1, 0x7E, 0x21, 0xD5, 0x4C, 0x24, 0xB7, 0x2E, 0xB5, 0x3B,
0x4F, 0xDC, 0xB9, 0xB5, 0x67, 0x2A, 0x95, 0xEE, 0xC2, 0x98, 0x2A, 0x0E, 0xC7, 0x8C, 0xCD, 0x5B,
0x4D, 0x6D, 0xA6, 0x2D, 0xDC, 0x7F, 0xA3, 0xF8, 0x0A, 0xD6, 0x9B, 0xEC, 0x82, 0x5C, 0xDF, 0xB0,
0xB0, 0x3E, 0x8D, 0xA6, 0x8A, 0x5C, 0xAB, 0xCA, 0x26, 0x60, 0x21, 0x0E, 0x33, 0x62, 0xDB, 0xB8,
0x70, 0x97, 0x93, 0xAD, 0x79, 0x2B, 0x36, 0x0F, 0x4C, 0x7E, 0xDD, 0xA6, 0xD4, 0xAD, 0x04, 0x45,
0xE1, 0x75, 0xFA, 0xD6, 0x6D, 0x47, 0x8C, 0x2C, 0x34, 0x79, 0x7B, 0xC4, 0xE9, 0x56, 0xA4, 0x50,
0x54, 0x6D, 0x70, 0xC7, 0xDB, 0xC4, 0xCC, 0x64, 0x60, 0x07, 0x36, 0x6A, 0x3D, 0x9B, 0x2B, 0x52,
0x0D, 0x4E, 0x95, 0x7B, 0x89, 0x06, 0xA7, 0xC9, 0x6D, 0x4F, 0x03, 0x76, 0x43, 0x91, 0x7A, 0xCB,
0x91, 0xB8, 0xDE, 0x65, 0x58, 0x0E, 0x0A, 0x82, 0xA1, 0xC9, 0x6E, 0x8C, 0x31, 0xD3, 0x77, 0x20,
0x0D, 0x6C, 0xB2, 0x34, 0x88, 0x3D, 0x34, 0x1D, 0x6F, 0xEA, 0x65, 0xCE, 0xF1, 0xF3, 0xE2, 0x0A,
0x04, 0x24, 0xCD, 0xA1, 0x99, 0xBA, 0x3A, 0x63, 0x72, 0xAA, 0xE4, 0x27, 0x73, 0xF4, 0xE0, 0xDE,
0xA3, 0x87, 0x0F, 0x7B, 0xDF, 0x3D, 0x70, 0xC7, 0xC1, 0x42, 0xFE, 0xFD, 0x8B, 0xB8, 0x98, 0x25,
0xEE, 0x88, 0x82, 0x3C, 0x4A, 0x29, 0xE8, 0x19, 0x0C, 0x4E, 0x39, 0xD3, 0x8C, 0x20, 0xA7, 0x20,
0x49, 0x8E, 0x6C, 0xB2, 0xB6, 0xEA, 0xC4, 0x8B, 0x86, 0x04, 0x50, 0x2E, 0xC6, 0xC8, 0xD7, 0x0C,
0xE1, 0xC3, 0x44, 0xE7, 0xC6, 0xFD, 0xD6, 0xE4, 0x35, 0x66, 0xEC, 0x5D, 0x65, 0x35, 0xE0, 0x4A,
0xC9, 0x02, 0x24, 0x47, 0x61, 0x3B, 0x8F, 0x21, 0x90, 0x71, 0x72, 0x76, 0x69, 0x2A, 0x67, 0x4C,
0x2C, 0x9F, 0xB4, 0xBE, 0x72, 0xA5, 0x44, 0x4C, 0x3D, 0xF1, 0xD1, 0x1C, 0x33, 0xF7, 0x97, 0x3F,
0xE6, 0xB3, 0xC9, 0x22, 0x11, 0x53, 0x9A, 0xA3, 0x37, 0x98, 0x67, 0x4E, 0x40, 0x59, 0x6B, 0xD6,
```

Anexo 49: Código de camera_index.h (13)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0x35, 0x2E, 0xB2, 0x98, 0xA5, 0xE6, 0x37, 0x23, 0x11, 0xE5, 0xE6, 0x75, 0x1D, 0x71, 0xB7, 0x29,
0x11, 0x88, 0xB3, 0xF3, 0x16, 0xDC, 0xC1, 0x96, 0xC8, 0x09, 0xC1, 0xB4, 0xAD, 0x96, 0x39, 0xFA,
0xF9, 0xF7, 0xEF, 0x9F, 0xD4, 0xDA, 0xCD, 0xEE, 0xF9, 0x55, 0xEB, 0xAC, 0xD7, 0x3D, 0x1E, 0x9C,
0x8A, 0x21, 0x9B, 0xF3, 0x6A, 0x9A, 0xA3, 0x5F, 0x19, 0x2F, 0xA8, 0x2F, 0xCD, 0xAB, 0x56, 0xBB,
0xD9, 0xDC, 0x9E, 0xD7, 0x23, 0x73, 0xF4, 0x96, 0xB3, 0x6A, 0x9F, 0x03, 0xAB, 0x66, 0x7B, 0x07,
0xB1, 0xCE, 0xCD, 0x11, 0xE7, 0x04, 0x4C, 0xAE, 0x1E, 0xF6, 0xCE, 0xB7, 0x67, 0xF4, 0x10, 0x64,
0x7A, 0x07, 0x9C, 0xCE, 0x41, 0xBB, 0xDE, 0x2E, 0xCA, 0xF5, 0xCC, 0x11, 0xE3, 0xD3, 0xEB, 0x36,
0xAF, 0xBA, 0xE7, 0x3B, 0xF0, 0x39, 0x33, 0x65, 0xA7, 0xC3, 0xDC, 0x3F, 0x3A, 0x32, 0x47, 0x17,
0x3F, 0x3C, 0xAF, 0x75, 0x41, 0xC6, 0xF6, 0xA3, 0xDE, 0xF6, 0xBC, 0xBB, 0xE0, 0x17, 0x4C, 0xC8,
0x4E, 0x1B, 0x18, 0x75, 0x77, 0x10, 0xB2, 0x63, 0x8E, 0x5E, 0x70, 0x4E, 0xC0, 0xE5, 0xAA, 0xF5,
0x70, 0x07, 0x91, 0xC0, 0xBD, 0x7E, 0xE6, 0x9C, 0xC0, 0xBF, 0x98, 0x7B, 0x55, 0xE4, 0x04, 0xB9,
0x97, 0x9B, 0xA6, 0x20, 0xE6, 0xD7, 0x33, 0x59, 0xEA, 0x74, 0x51, 0x4A, 0xF8, 0x3B, 0x84, 0x8E,
0x80, 0x5E, 0x6F, 0x9C, 0x10, 0x24, 0x1D, 0xA8, 0x24, 0x0E, 0xAA, 0xE5, 0x02, 0x45, 0x92, 0xF8,
0x6A, 0xAB, 0x39, 0xEA, 0x96, 0x28, 0xC0, 0x49, 0xD5, 0x84, 0xCA, 0x69, 0x53, 0xF2, 0x9B, 0xAC,
0x3F, 0x64, 0xA8, 0xB3, 0xFB, 0x79, 0xC0, 0x43, 0x3B, 0xA6, 0x12, 0xD5, 0x5B, 0x25, 0x1B, 0x8D,
0xAC, 0xE8, 0xCA, 0x1C, 0xF5, 0x3A, 0x65, 0xD6, 0xDE, 0x01, 0x8C, 0x31, 0xEF, 0x3D, 0x5D, 0x1C,
0x04, 0x1B, 0xE3, 0x91, 0x90, 0x9A, 0xA3, 0xA7, 0xF1, 0xF1, 0x2E, 0xA8, 0xD4, 0xCB, 0x34, 0xE5,
0xB4, 0x39, 0xB0, 0x28, 0xE2, 0x08, 0x64, 0xEA, 0x1D, 0x09, 0x4D, 0x82, 0xCC, 0xE7, 0x05, 0xE6,
0x36, 0x71, 0x61, 0xED, 0x80, 0x8F, 0x02, 0xBA, 0x31, 0x2A, 0x11, 0x21, 0x24, 0x35, 0x79, 0x74,
0x30, 0x44, 0x62, 0x51, 0xBE, 0x02, 0x3C, 0x02, 0x44, 0x43, 0x9F, 0xDF, 0xE5, 0xB8, 0x31, 0x22,
0x09, 0x29, 0x54, 0xC3, 0xF8, 0x78, 0x27, 0x54, 0x76, 0x49, 0x5F, 0x8A, 0x38, 0x12, 0x97, 0x28,
0x85, 0x75, 0x6F, 0x09, 0x97, 0x32, 0x69, 0x77, 0xC2, 0x65, 0x86, 0xFC, 0xC5, 0x56, 0xE9, 0x2B,
0xA6, 0x04, 0x54, 0xA2, 0xC3, 0x83, 0x85, 0x4A, 0x22, 0xCC, 0x57, 0x10, 0x2B, 0xB0, 0xFE, 0xF6,
0x48, 0xB0, 0x79, 0xC7, 0x2F, 0xE9, 0xCC, 0xD1, 0x33, 0x5C, 0x7F, 0xCD, 0x8E, 0x76, 0x81, 0xE3,
0x49, 0x48, 0xBD, 0x1D, 0x00, 0x89, 0x64, 0x11, 0x70, 0x34, 0x25, 0x1A, 0xE7, 0xB7, 0x84, 0xC6,
0xF9, 0x2D, 0xA2, 0x81, 0xF0, 0x7B, 0x07, 0x2F, 0xB1, 0xB3, 0x31, 0x1C, 0x11, 0xA1, 0x39, 0xBA,
0xBC, 0x5A, 0x78, 0x01, 0xBB, 0x5B, 0xF8, 0x25, 0xFB, 0xBE, 0x53, 0x90, 0x9C, 0xED, 0x80, 0x49,
0x2C, 0x90, 0x8C, 0x91, 0x33, 0x89, 0xCA, 0xD9, 0x2D, 0xA1, 0x52, 0x26, 0xEB, 0x2E, 0xA8, 0x4C,
0x11, 0x71, 0x2D, 0x4C, 0x1C, 0x76, 0xE7, 0xE2, 0xA6, 0xC0, 0x28, 0xB4, 0xE6, 0xE8, 0xFB, 0xE4,
```

Anexo 50: Código de camera_index.h (14)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0xCB, 0x2E, 0xC0, 0x34, 0x77, 0xC0, 0x45, 0x95, 0x27, 0x1D, 0x2F, 0x67, 0xB0, 0x58, 0xBE, 0x25,
0x6C, 0x5A, 0xAD, 0xDB, 0xAC, 0x2A, 0x0B, 0x6C, 0x11, 0xE4, 0xBC, 0xC7, 0x93, 0x09, 0x2C, 0x83,
0x36, 0x2F, 0x2D, 0x29, 0x72, 0xA8, 0x2F, 0xE2, 0xBB, 0x71, 0xC9, 0xBF, 0x6F, 0xBC, 0x87, 0x91,
0x61, 0xF7, 0xB9, 0x36, 0x32, 0x9A, 0xFA, 0xB5, 0xF0, 0x6B, 0x2F, 0x96, 0x73, 0xDB, 0x5D, 0x0D,
0x60, 0x82, 0xA7, 0x7C, 0x53, 0x7D, 0x6B, 0x1E, 0x6D, 0xF0, 0x6C, 0x1F, 0x5D, 0xF3, 0xC7, 0x10,
0x77, 0x59, 0x48, 0xBF, 0xC1, 0xB6, 0xF1, 0x0B, 0x71, 0xB7, 0x57, 0xA6, 0xCB, 0x04, 0xC1, 0xD8,
0xDD, 0x8D, 0xCB, 0x19, 0x2C, 0x91, 0xE0, 0x60, 0x37, 0x26, 0x3D, 0xF0, 0x24, 0xBC, 0x20, 0xE8,
0x4B, 0x58, 0xC4, 0xA3, 0xD5, 0x78, 0xF3, 0x82, 0xB2, 0x1A, 0x43, 0x5D, 0xFE, 0xED, 0xA9, 0x71,
0xC9, 0x6F, 0x03, 0xDB, 0x38, 0x5D, 0x89, 0x2B, 0xD4, 0x55, 0x1C, 0x5D, 0x24, 0x2A, 0x29, 0xA7,
0xB9, 0xB6, 0x27, 0xAA, 0x0F, 0xA0, 0xAA, 0xFB, 0xA2, 0x1A, 0xF5, 0x22, 0x01, 0xF9, 0x05, 0x3D,
0x53, 0xD1, 0xB6, 0x9A, 0x8E, 0xB7, 0xD8, 0x8A, 0x59, 0xAB, 0xCD, 0xDB, 0x30, 0x6B, 0x05, 0x30,
0xD9, 0x4B, 0x76, 0x87, 0xA0, 0x6D, 0x00, 0x5E, 0x7B, 0x01, 0x8A, 0xCD, 0x7A, 0x18, 0xA0, 0xB8,
0xBE, 0x87, 0x06, 0x0A, 0xBC, 0xE5, 0x3D, 0xAB, 0xA3, 0xDB, 0x04, 0x15, 0x27, 0x34, 0x47, 0xAF,
0x90, 0x1B, 0x42, 0x91, 0xD9, 0x17, 0x60, 0xF1, 0xC4, 0x07, 0x0B, 0x2F, 0xA9, 0xF7, 0xA1, 0xA1,
0x03, 0x41, 0xE6, 0x9E, 0xBD, 0xF9, 0x72, 0x47, 0xD2, 0x89, 0x94, 0xF8, 0x0A, 0x8E, 0x36, 0x6E,
0x0C, 0x22, 0x0E, 0xB7, 0xDC, 0x11, 0x88, 0xA5, 0xD4, 0xF6, 0xCD, 0xC0, 0xDB, 0xD0, 0x75, 0xAF,
0x77, 0xE9, 0x04, 0x2E, 0x1C, 0x2F, 0xB4, 0xB7, 0xE7, 0x00, 0x6D, 0xC0, 0x8F, 0x93, 0x09, 0xB1,
0xB6, 0x6F, 0x24, 0xA0, 0x09, 0x78, 0xE1, 0xCD, 0x2B, 0xD2, 0xDF, 0x72, 0xE1, 0xC5, 0xD6, 0x16,
0x2B, 0x39, 0x0B, 0x50, 0xBC, 0xBC, 0xD8, 0x6B, 0xE1, 0x85, 0x39, 0x0F, 0x94, 0x19, 0x98, 0xB6,
0x87, 0x4E, 0x0A, 0x20, 0xC4, 0x7B, 0xEE, 0x3C, 0xDB, 0x80, 0x25, 0x28, 0xE3, 0x8C, 0x1E, 0x2D,
0xBF, 0x0F, 0xB5, 0xBE, 0x4B, 0x24, 0x4A, 0xAF, 0xEE, 0x5A, 0x67, 0x9D, 0x5E, 0xBC, 0xBC, 0xEB,
0xB4, 0x3F, 0xEF, 0x02, 0x8F, 0x31, 0xBF, 0x5D, 0x7C, 0xDA, 0xDB, 0x40, 0x03, 0xD9, 0xE8, 0x35,
0xBB, 0xCE, 0xB0, 0x41, 0xC2, 0xDE, 0x3D, 0x90, 0xDA, 0x87, 0x8B, 0xA4, 0xF6, 0x17, 0x10, 0x4A,
0xD3, 0x2D, 0x32, 0xDE, 0x94, 0x65, 0xBC, 0xEF, 0x2F, 0xF6, 0x83, 0xD0, 0xF4, 0x60, 0xA9, 0x6E,
0x7A, 0xD0, 0x54, 0x67, 0x88, 0x9B, 0xAD, 0x62, 0x98, 0xB6, 0xEC, 0x60, 0x25, 0xA1, 0xD8, 0xCB,
0xDA, 0x25, 0xC9, 0xB5, 0xAE, 0x76, 0xC9, 0x72, 0x91, 0x18, 0xE9, 0x24, 0xD7, 0x4B, 0xAE, 0x8A,
0x9C, 0x7D, 0xDE, 0xCB, 0xBA, 0xDD, 0x32, 0x69, 0x77, 0x09, 0x1A, 0x1F, 0xAD, 0xDE, 0x4F, 0xE7,
0x68, 0x63, 0x30, 0x24, 0x1D, 0x60, 0xF1, 0xEA, 0xC9, 0x3E, 0xDB, 0x85, 0x68, 0xDE, 0xC3, 0xC4,
0x51, 0xAC, 0xF5, 0xA1, 0x73, 0x9D, 0x83, 0xDD, 0xCD, 0x93, 0x1D, 0x23, 0x32, 0x47, 0x2F, 0xB1,
```

Anexo 51: Código de camera_index.h (15)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0x1B, 0x18, 0x17, 0x9E, 0x2F, 0xDF, 0xFD, 0xB4, 0x17, 0xD4, 0xF8, 0xCC, 0x87, 0x81, 0x4C, 0x28,
0x7D, 0x68, 0xBC, 0x66, 0x73, 0xE2, 0xFB, 0x9E, 0xBF, 0x31, 0x64, 0x92, 0x0E, 0x96, 0x15, 0xF5,
0x57, 0xFC, 0x68, 0x2F, 0x70, 0x45, 0xB3, 0x1E, 0x06, 0xB1, 0x58, 0xE7, 0x43, 0x83, 0xB6, 0x9C,
0x38, 0x64, 0xB1, 0x31, 0x64, 0x9C, 0xCA, 0x1C, 0xBD, 0xAB, 0x3F, 0x87, 0x7F, 0xF7, 0x02, 0x97,
0x98, 0xF1, 0x30, 0x60, 0x49, 0x6D, 0x0F, 0x0D, 0xD5, 0x78, 0xB1, 0x79, 0x3A, 0x04, 0x1A, 0x73,
0xF4, 0xF4, 0xA7, 0xFD, 0xF4, 0x7E, 0x6C, 0xB2, 0x8A, 0x08, 0xED, 0x84, 0x07, 0x57, 0xEA, 0xD0,
0x68, 0xAC, 0xB6, 0x40, 0x63, 0xC5, 0x04, 0xFF, 0x6D, 0x4F, 0x68, 0xAC, 0xAA, 0xA3, 0xF1, 0x99,
0xE3, 0x65, 0xF5, 0x25, 0xE0, 0xC3, 0x9F, 0xC5, 0x18, 0xA3, 0xCD, 0xCB, 0x51, 0x44, 0xC8, 0x6E,
0x1A, 0x83, 0x23, 0xE3, 0x29, 0xDA, 0x4F, 0x41, 0x8A, 0xE7, 0xDD, 0x47, 0x08, 0x25, 0x4A, 0x1E,
0x1A, 0xA7, 0x09, 0xB2, 0xF0, 0x7B, 0x1B, 0xD3, 0x6D, 0xAE, 0x2D, 0x2B, 0xB4, 0xE6, 0xE8, 0x39,
0x7C, 0x31, 0x9E, 0xF1, 0x2F, 0xFB, 0x6A, 0xF9, 0xD4, 0xF9, 0xF7, 0x81, 0x5A, 0x4A, 0xDF, 0x2F,
0x02, 0x38, 0x68, 0xB0, 0xBD, 0xA9, 0xBB, 0xD5, 0x23, 0x0D, 0x29, 0x72, 0x09, 0xDF, 0x1B, 0xF1,
0x7D, 0xBF, 0x00, 0x26, 0x42, 0xEC, 0x0D, 0x43, 0x45, 0xEF, 0x7D, 0xC0, 0x18, 0x3D, 0x16, 0xC4,
0x8B, 0xB4, 0x78, 0x15, 0x5E, 0x19, 0x52, 0xF2, 0xE1, 0x27, 0x7E, 0x4B, 0x0B, 0xA6, 0xF5, 0x80,
0x12, 0xC7, 0x81, 0x85, 0x30, 0xA6, 0xC6, 0x5B, 0x76, 0x38, 0x38, 0x15, 0x03, 0xAA, 0x73, 0x91,
0xCF, 0xDC, 0xB0, 0x97, 0x50, 0xA2, 0xB9, 0x39, 0x7A, 0xCB, 0x5E, 0x12, 0x08, 0xBC, 0xD8, 0xB7,
0xCD, 0x99, 0x71, 0x23, 0x62, 0xD7, 0xF7, 0x40, 0xA8, 0x18, 0x24, 0xF9, 0xAE, 0x26, 0xD3, 0x88,
0x8E, 0x94, 0xDF, 0x46, 0x97, 0x7C, 0xB0, 0xC1, 0xBC, 0xAC, 0x7C, 0x3A, 0x76, 0xD5, 0xC2, 0xCA,
0xBF, 0xB8, 0x31, 0x38, 0x75, 0x91, 0xC6, 0xDC, 0x39, 0x28, 0x0C, 0xC4, 0xDB, 0x25, 0x73, 0x58,
0xC5, 0xCF, 0x33, 0x71, 0x4B, 0x24, 0x8F, 0x69, 0xC6, 0x6A, 0x65, 0x1F, 0xDF, 0x94, 0xDB, 0x4C,
0xD5, 0x82, 0x96, 0x3F, 0x88, 0x29, 0xEB, 0x21, 0x3B, 0x8C, 0xCD, 0xFF, 0x9F, 0x7F, 0x97, 0xF9,
0x0C, 0x7B, 0xF7, 0x67, 0x22, 0x98, 0x69, 0x04, 0xBE, 0x35, 0x34, 0xF3, 0x9E, 0x8E, 0xCA, 0xD1,
0xFC, 0x54, 0xA7, 0x7A, 0x66, 0xB0, 0xC6, 0xD6, 0x83, 0xC0, 0xF2, 0xC9, 0x82, 0x8E, 0xEE, 0xD8,
0x9E, 0x15, 0xCE, 0xB1, 0x4B, 0x1B, 0xC8, 0xB6, 0x2F, 0x97, 0x70, 0xF0, 0x92, 0x04, 0x14, 0x83,
0x15, 0x6A, 0x47, 0xCF, 0x7E, 0x7C, 0x75, 0x21, 0x9E, 0x12, 0x7B, 0xE9, 0x21, 0x1B, 0xDB, 0x47,
0x27, 0xC6, 0x24, 0x74, 0x85, 0x9B, 0xD7, 0x30, 0x1B, 0x2B, 0xDE, 0xBB, 0xBA, 0x44, 0xBE, 0x31,
0x46, 0x01, 0x7E, 0xE1, 0x05, 0xD4, 0x18, 0x1A, 0x31, 0x47, 0xC7, 0xB3, 0xF8, 0x7D, 0xBF, 0x0D,
0xCF, 0x27, 0x53, 0xE2, 0xCA, 0x91, 0x42, 0xD9, 0x5F, 0x7D, 0x07, 0x86, 0xC6, 0x54, 0xDF, 0x1A,
0x47, 0xFD, 0xF3, 0xD6, 0x11, 0x7B, 0x1C, 0x0F, 0x60, 0x80, 0x1F, 0x00, 0x02, 0x0C, 0x03, 0x20,
```

Anexo 52: Código de camera_index.h (16)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0xC0, 0x87, 0x23, 0xF9, 0x78, 0x20, 0x76, 0x1A, 0xDC, 0xE4, 0x4C, 0x40, 0x26, 0x6D, 0xED, 0x48,
0xE0, 0x74, 0xC4, 0x1E, 0x34, 0xBE, 0x89, 0x29, 0x83, 0x99, 0xB7, 0x2A, 0xA2, 0xF4, 0xF1, 0xDC,
0x5B, 0xE2, 0x0C, 0x71, 0x4C, 0x2D, 0xBD, 0xB9, 0x74, 0xEA, 0xC8, 0xEB, 0x8F, 0x8E, 0xA3, 0x01,
0xF1, 0x7B, 0xCC, 0x86, 0x06, 0xF5, 0x43, 0x9C, 0x66, 0x8B, 0xDD, 0x32, 0xAE, 0x91, 0x58, 0x85,
0x8C, 0x27, 0xC8, 0x09, 0x32, 0x9C, 0xC3, 0x85, 0x8D, 0x28, 0x7E, 0xC7, 0x76, 0x0C, 0x61, 0x40,
0x0D, 0x3B, 0x27, 0x62, 0xFB, 0xF0, 0x44, 0x9E, 0x79, 0x03, 0x7C, 0x29, 0x3E, 0x4E, 0x66, 0x55,
0x7F, 0x06, 0x8A, 0xF4, 0xD7, 0xA1, 0xE1, 0x86, 0x10, 0xC2, 0x8F, 0xB9, 0x0A, 0x46, 0x3F, 0x75,
0x96, 0x53, 0x3B, 0x90, 0x9D, 0xE4, 0x3B, 0xDB, 0xF9, 0x9C, 0xFC, 0x47, 0x32, 0x61, 0x13, 0x37,
0xF8, 0x1B, 0xE4, 0x87, 0xC0, 0xE3, 0x28, 0xCA, 0xEE, 0x47, 0xC9, 0x8B, 0x79, 0x55, 0x22, 0x6E,
0x87, 0x86, 0xEC, 0x83, 0xE5, 0xF9, 0xA5, 0x3C, 0x71, 0xF7, 0xEE, 0x32, 0xE6, 0x6B, 0x28, 0xC3,
0xE0, 0x54, 0x72, 0xE2, 0x06, 0x4E, 0x28, 0x4F, 0x3F, 0xAF, 0xF3, 0xCE, 0xF0, 0x88, 0x98, 0x2B,
0x1C, 0xEE, 0xC4, 0x92, 0xA7, 0x2C, 0xF0, 0xE0, 0x41, 0x9A, 0xDB, 0xDD, 0xA1, 0xA4, 0x4A, 0x34,
0x11, 0xE3, 0x21, 0x32, 0x20, 0xF2, 0x40, 0x6D, 0xF9, 0x4C, 0xBC, 0x14, 0x89, 0x4C, 0x6A, 0x77,
0x53, 0x86, 0x8F, 0x65, 0x9C, 0x30, 0x13, 0x11, 0x9B, 0x1B, 0x88, 0x5F, 0x33, 0x3C, 0x4E, 0x9E,
0x7A, 0x15, 0xF2, 0x3D, 0xE6, 0x5E, 0x5F, 0xC3, 0xF2, 0xF2, 0xDB, 0x31, 0xD8, 0x9F, 0x39, 0x73,
0xF2, 0x83, 0x1C, 0x9F, 0x4C, 0xA5, 0x72, 0x9C, 0xA6, 0x38, 0x32, 0xC5, 0x32, 0x72, 0xB3, 0x0F,
0x9F, 0x00, 0x86, 0xB2, 0x9D, 0xEF, 0xE4, 0xF9, 0xFC, 0x8C, 0x39, 0xD9, 0x87, 0x4F, 0xBC, 0x3E,
0xB0, 0x50, 0x82, 0xE8, 0x0E, 0x09, 0x8D, 0x62, 0x9C, 0xDD, 0x6A, 0xCC, 0x54, 0xE2, 0x22, 0xC0,
0x61, 0x11, 0xAB, 0x4C, 0x01, 0xD7, 0x30, 0x14, 0x01, 0x55, 0x13, 0xF5, 0xE9, 0x29, 0xAF, 0x35,
0x8C, 0xB9, 0x8C, 0x95, 0xF4, 0xEF, 0x77, 0x54, 0xE1, 0x6F, 0xA2, 0xF0, 0x89, 0x53, 0x99, 0x8A,
0x27, 0xF3, 0xE3, 0xC8, 0x62, 0xCC, 0xD5, 0x13, 0x87, 0x91, 0xAF, 0x2B, 0x89, 0xFC, 0x3C, 0x31,
0xAB, 0x05, 0x39, 0x4C, 0xF1, 0xF8, 0x7E, 0x46, 0x54, 0xD5, 0xD5, 0x41, 0xEE, 0x96, 0xA1, 0xBE,
0x80, 0x64, 0x0C, 0xA9, 0xF0, 0x63, 0x8A, 0x0F, 0xDF, 0xB0, 0x8F, 0x99, 0x88, 0xDF, 0xC4, 0xE5,
0xFD, 0xBA, 0xE7, 0x62, 0x3D, 0x77, 0xD5, 0xD9, 0x75, 0x3C, 0x45, 0x29, 0xCE, 0x32, 0x0D, 0xC7,
0x73, 0x42, 0x35, 0x0C, 0x8F, 0x20, 0x0D, 0xEB, 0x78, 0xC9, 0x06, 0x2D, 0x21, 0xF0, 0x31, 0x0D,
0x7D, 0x57, 0x8D, 0x26, 0x91, 0x91, 0xFE, 0x0E, 0xB1, 0x7F, 0x0D, 0x8C, 0x3E, 0xDC, 0xFF, 0x14,
0xE5, 0xF7, 0x9B, 0x53, 0xFE, 0x6C, 0x8E, 0xE7, 0x3C, 0x86, 0x0A, 0x30, 0xBC, 0xFF, 0x89, 0x43,
0x7D, 0xF3, 0x00, 0xA6, 0x84, 0x2F, 0x7C, 0xE2, 0x9B, 0x0F, 0x82, 0xC5, 0x84, 0xBD, 0x3E, 0xBB,
0xC6, 0x59, 0x44, 0xB8, 0x35, 0xE8, 0x0C, 0xBB, 0x35, 0x1F, 0x07, 0x0B, 0x60, 0x8F, 0x93, 0x44,
```

Anexo 53: Código de camera_index.h (17)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
0x16, 0xCD, 0xE8, 0x39, 0x18, 0x4A, 0xCD, 0xB4, 0xF6, 0xC1, 0xC7, 0x40, 0x07, 0x02, 0x50, 0xCF,
0xB8, 0xFF, 0x89, 0xB3, 0xB8, 0x31, 0x26, 0x10, 0xCD, 0xC1, 0x0C, 0xDB, 0x27, 0x50, 0x77, 0x10,
0x65, 0x4F, 0xA6, 0xDF, 0xFF, 0x14, 0xB1, 0x6A, 0x88, 0x9F, 0x6E, 0x3E, 0xC4, 0x1E, 0x12, 0x17,
0x83, 0xA8, 0x86, 0xF1, 0x13, 0x0D, 0xCE, 0xEB, 0x2D, 0x47, 0xC1, 0xF3, 0x9F, 0x38, 0x4E, 0xED,
0x48, 0xBC, 0x7E, 0x41, 0xE6, 0xE8, 0x06, 0x34, 0x9D, 0x97, 0x08, 0xC4, 0x56, 0x93, 0x3B, 0xCF,
0x3B, 0x9E, 0x6B, 0x39, 0xC4, 0xFA, 0xC8, 0x12, 0xF3, 0x71, 0x5A, 0x70, 0x11, 0xE9, 0x4E, 0x43,
0xBC, 0x4E, 0xEB, 0xB5, 0x67, 0xE3, 0x8C, 0x9B, 0x1E, 0x33, 0x31, 0x4E, 0x4F, 0xC1, 0xCA, 0xC8,
0x8E, 0x52, 0x92, 0xC0, 0x88, 0xBD, 0x77, 0x45, 0x98, 0x29, 0x65, 0x61, 0xA1, 0x8C, 0xD4, 0x45,
0xD8, 0x2C, 0xA9, 0xD6, 0x91, 0xCA, 0x89, 0xDB, 0x0A, 0xF4, 0x8C, 0xD8, 0x16, 0x7F, 0x05, 0x9E,
0x5B, 0x3B, 0xBE, 0x13, 0x9B, 0x61, 0x9D, 0x07, 0x9B, 0x40, 0x61, 0x90, 0x32, 0x51, 0x9E, 0x99,
0xD2, 0x5D, 0xFD, 0x51, 0x92, 0x49, 0x72, 0x6C, 0x26, 0x3E, 0x4A, 0x4D, 0xE3, 0x05, 0x8D, 0xCF,
0xFC, 0x07, 0x77, 0x9A, 0x3F, 0x4F, 0x44, 0x11, 0x54, 0x72, 0xD2, 0xB1, 0x62, 0x30, 0xE1, 0x81,
0xEC, 0xBF, 0x1E, 0x51, 0x1B, 0x11, 0xE8, 0xAE, 0x2F, 0x1D, 0xCC, 0x0E, 0x9F, 0x5E, 0xFF, 0x00,
0xC5, 0x5B, 0xB4, 0x20, 0x5C, 0x9A, 0x84, 0xE0, 0x22, 0x6E, 0xFF, 0x4A, 0x29, 0x93, 0x56, 0x51,
0xE1, 0xC1, 0xDB, 0x77, 0x91, 0x71, 0x8A, 0x38, 0xC4, 0x9D, 0x7E, 0x8A, 0x94, 0x71, 0x2D, 0xA7,
0x4D, 0xF5, 0xF7, 0x0A, 0xBD, 0x9A, 0xED, 0x8A, 0xE8, 0x95, 0x96, 0x5E, 0xA1, 0xE6, 0xAE, 0x5C,
0x4E, 0xAC, 0x36, 0xB7, 0x47, 0x8A, 0xB1, 0x03, 0xEA, 0x2D, 0xC4, 0x1A, 0x23, 0xE3, 0xE6, 0x2B,
0xE2, 0xDA, 0xDE, 0xAA, 0xC1, 0xCE, 0xD7, 0x64, 0x91, 0x54, 0x15, 0x6D, 0x10, 0x17, 0x0C, 0xF8,
0xE2, 0x97, 0x57, 0x2F, 0x59, 0xD2, 0x51, 0xD7, 0x2A, 0x47, 0xE9, 0x0E, 0x87, 0xBF, 0xEB, 0x5C,
0x3B, 0x03, 0x83, 0xAD, 0x01, 0x4D, 0xB3, 0x48, 0x36, 0x71, 0x63, 0xC9, 0x62, 0x81, 0x1D, 0x7E,
0x10, 0x73, 0xB2, 0xD2, 0x93, 0x02, 0xF8, 0xB8, 0x54, 0x16, 0x6F, 0x91, 0x15, 0x05, 0x22, 0xF1,
0x09, 0xA5, 0xE0, 0xB0, 0x86, 0x70, 0xE5, 0x80, 0x65, 0x19, 0xB9, 0xCE, 0xBB, 0x63, 0xA8, 0xE0,
0xE7, 0x04, 0x7D, 0x62, 0x26, 0x19, 0x65, 0x69, 0xE1, 0x95, 0x4C, 0x89, 0x16, 0x10, 0x99, 0xF8,
0xF1, 0x7B, 0x6B, 0x0C, 0xC9, 0xF1, 0x19, 0x78, 0x7E, 0xC3, 0x05, 0x0D, 0x8E, 0x6F, 0x8A, 0xD4,
0x11, 0xE6, 0x4A, 0x80, 0xAC, 0x2A, 0x04, 0x4F, 0x43, 0x7A, 0x6E, 0x29, 0xFB, 0xE8, 0xD9, 0xA9,
0xDE, 0x2B, 0xAE, 0xDD, 0xB2, 0x36, 0x2D, 0xCF, 0xB0, 0xC3, 0x75, 0xD3, 0x8A, 0x3E, 0x25, 0xC5,
0x20, 0x49, 0x30, 0x6B, 0xC2, 0x66, 0xDA, 0x14, 0xC5, 0x2F, 0xA2, 0x01, 0x91, 0xEC, 0x6A, 0x40,
0xE4, 0xC8, 0x9E, 0xEE, 0xE2, 0x32, 0xED, 0x42, 0x06, 0x72, 0x99, 0xC5, 0x0C, 0xF6, 0xD6, 0x8F,
0x19, 0x2B, 0xD0, 0xD2, 0x09, 0xAA, 0x14, 0x0A, 0x6D, 0x06, 0x2C, 0xAC, 0x18, 0x62, 0x86, 0x48,
```

Anexo 54: Código de camera_index.h (18)

```
 0xDA, 0x6C, 0xB7, 0x99, 0xAE, 0x0E, 0x17, 0x21, 0x58, 0x69, 0x1E, 0xF9, 0xA4, 0xF8, 0x8D, 0xB5,
 0x6C, 0x71, 0xF0, 0x40, 0x0B, 0x57, 0x14, 0xD4, 0x70, 0x5A, 0xC9, 0x04, 0xB2, 0xDF, 0x2B, 0x21,
 0x50, 0xEE, 0xBA, 0xE0, 0xB4, 0xF0, 0xD3, 0xBA, 0xD8, 0x1A, 0x23, 0xC3, 0xB8, 0xE3, 0x18, 0x73,
 0x46, 0x24, 0xBB, 0xA2, 0x04, 0xF1, 0xF5, 0xEE, 0x34, 0x0B, 0xF9, 0x5A, 0x57, 0x7A, 0xA3, 0xA0,
 0x15, 0xDD, 0xB7, 0x96, 0xE8, 0x83, 0x8B, 0x95, 0xC7, 0xAA, 0xF2, 0x51, 0x97, 0x5D, 0x42, 0xA1,
 0xDE, 0x65, 0x27, 0xD4, 0xC7, 0x15, 0xD5, 0xC7, 0x52, 0x7D, 0x46, 0x90, 0x34, 0x84, 0xE5, 0x2D,
 0x7F, 0xEC, 0x8C, 0xBF, 0x3D, 0x4D, 0x34, 0x5B, 0x8D, 0x0B, 0xE5, 0x94, 0xAD, 0xB8, 0xA2, 0x5E,
 0x31, 0x41, 0xEA, 0x9E, 0x62, 0xA1, 0xD6, 0x6A, 0x5C, 0x4D, 0xAD, 0xA8, 0x95, 0x67, 0x04, 0x89,
 0x5A, 0xFA, 0x86, 0x3F, 0x52, 0x25, 0xDE, 0x42, 0xE6, 0xFF, 0xA7, 0x4B, 0xFC, 0xCE, 0x94, 0x58,
 0x58, 0xB1, 0xFF, 0x5A, 0x5A, 0xCA, 0xC4, 0x30, 0x45, 0xC9, 0x78, 0xC9, 0x50, 0x4A, 0x1A, 0x8F,
 0x54, 0xA8, 0x63, 0x39, 0x0A, 0xA9, 0xA3, 0x41, 0xA2, 0x06, 0xC6, 0x5F, 0x2B, 0x19, 0x2B, 0x1E,
 0x9D, 0x04, 0x42, 0xC2, 0x40, 0x34, 0xE0, 0x23, 0xE3, 0x2C, 0xBB, 0xD4, 0x14, 0x8D, 0x90, 0x50,
 0x36, 0xD3, 0xFE, 0xA8, 0x03, 0x62, 0x95, 0x52, 0x63, 0xE2, 0x00, 0x11, 0xF4, 0x79, 0x62, 0x96,
 0x8A, 0x82, 0x1C, 0xEC, 0xD3, 0x9A, 0xF9, 0x93, 0x83, 0xD9, 0xF2, 0x41, 0xDE, 0x14, 0x7E, 0xF1,
 0xC3, 0x73, 0xC3, 0xF3, 0x0D, 0xF1, 0x16, 0x4D, 0x3F, 0x7E, 0x6B, 0x8E, 0x21, 0x5F, 0x31, 0xC7,
 0x17, 0x69, 0xC4, 0x9D, 0x1A, 0x74, 0x46, 0x02, 0xE8, 0x59, 0xD9, 0x93, 0xE0, 0xF8, 0xAE, 0x19,
 0xBF, 0x45, 0xAE, 0x54, 0x3D, 0xD1, 0xA4, 0x7E, 0x17, 0x2B, 0x92, 0x31, 0xA7, 0xA0, 0x49, 0x6C,
 0x79, 0x57, 0xEA, 0xB8, 0x96, 0x58, 0x8A, 0x96, 0x85, 0x1B, 0x98, 0x30, 0x3E, 0xFD, 0xC5, 0x5A,
 0x51, 0xAF, 0x40, 0xA9, 0x21, 0x63, 0xB2, 0xC4, 0x96, 0x89, 0xAE, 0x6B, 0xD6, 0xD4, 0xAD, 0xBD,
 0x0B, 0x10, 0x65, 0x5B, 0x49, 0xDA, 0x6C, 0x9E, 0x8F, 0x8A, 0xB0, 0xB8, 0xA8, 0x72, 0xE2, 0x33,
 0x38, 0x8D, 0x36, 0x2C, 0xC5, 0x37, 0xF1, 0x52, 0xAE, 0xC1, 0xA9, 0xF8, 0x9F, 0x0A, 0xFF, 0x0B,
 0x9B, 0xFC, 0x8E, 0x51, 0xC1, 0x70, 0x00, 0x00
};
```

Anexo 55: Código de camera_index.h (19)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
#if defined(CAMERA_MODEL_WROVER_KIT)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    21
#define SIOD_GPIO_NUM    26
#define SIOC_GPIO_NUM    27

#define Y9_GPIO_NUM      35
#define Y8_GPIO_NUM      34
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      19
#define Y4_GPIO_NUM      18
#define Y3_GPIO_NUM       5
#define Y2_GPIO_NUM       4
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22

#elif defined(CAMERA_MODEL_ESP_EYE)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    4
#define SIOD_GPIO_NUM    18
#define SIOC_GPIO_NUM    23

#define Y9_GPIO_NUM      36
#define Y8_GPIO_NUM      37
#define Y7_GPIO_NUM      38
```

Anexo 56: Código de camera_pins.h (1)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
#define Y6_GPIO_NUM      39
#define Y5_GPIO_NUM      35
#define Y4_GPIO_NUM      14
#define Y3_GPIO_NUM      13
#define Y2_GPIO_NUM      34
#define VSYNC_GPIO_NUM   5
#define HREF_GPIO_NUM    27
#define PCLK_GPIO_NUM    25

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   15
#define XCLK_GPIO_NUM    27
#define SIOD_GPIO_NUM    25
#define SIOC_GPIO_NUM    23

#define Y9_GPIO_NUM      19
#define Y8_GPIO_NUM      36
#define Y7_GPIO_NUM      18
#define Y6_GPIO_NUM      39
#define Y5_GPIO_NUM       5
#define Y4_GPIO_NUM      34
#define Y3_GPIO_NUM      35
#define Y2_GPIO_NUM      32
#define VSYNC_GPIO_NUM   22
#define HREF_GPIO_NUM    26
#define PCLK_GPIO_NUM    21
```

Anexo 57: Código de camera_pins.h (2)

AlimentadorESP32CAM | app_httpd.cpp | camera_index.h | camera_pins.h

```
#elif defined(CAMERA_MODEL_M5STACK_WIDE)
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM     22
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
#define Y5_GPIO_NUM        5
#define Y4_GPIO_NUM       34
#define Y3_GPIO_NUM       35
#define Y2_GPIO_NUM       32
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     26
#define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
```

Anexo 58: Código de camera_pins.h (3)

```
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

#else
#error "Camera model not selected"
#endif
```

Anexo 59: Código de camera_pins.h (4)

IPaddressClassification.py > ...

```python
import cv2 #opencv
import urllib.request #para abrir y leer URL
import numpy as np

#PROGRAMA DE CLASIFICACION DE OBJETOS PARA VIDEO EN DIRECCION IP

url = 'http://192.168.1.184/capture'
winName = 'ESP32 CAMERA'
cv2.namedWindow(winName,cv2.WINDOW_AUTOSIZE)
#scale_percent = 80 # percent of original size    #para procesamiento de imagen

classNames = []
classFile = 'coco.names'
with open(classFile,'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')

configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weightsPath = 'frozen_inference_graph.pb'

net = cv2.dnn_DetectionModel(weightsPath,configPath)
net.setInputSize(320,320)
#net.setInputSize(480,480)
net.setInputScale(1.0/127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
```

Anexo 60: Código IPaddressClassification.py (1)

IPaddressClassification.py > ...

```python
while(1):
    imgResponse = urllib.request.urlopen (url) #abrimos el URL
    imgNp = np.array(bytearray(imgResponse.read()),dtype=np.uint8)
    img = cv2.imdecode (imgNp,-1) #decodificamos

    img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE) # vertical


    classIds, confs, bbox = net.detect(img,confThreshold=0.5)
    print(classIds,bbox)

    if len(classIds) != 0:
        for classId, confidence,box in zip(classIds.flatten(),confs.flatten(),bbox):
            cv2.rectangle(img,box,color=(0,255,0),thickness = 3) #mostramos en rectangulo lo que se encuentra

    cv2.imshow(winName,img) # mostramos la imagen

    #esperamos a que se presione ESC para terminar el programa
    tecla = cv2.waitKey(5) & 0xFF
    if tecla == 27:
        break
cv2.destroyAllWindows()
```

Anexo 61: Código IPaddressClassification.py (2)