UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y COMUNICACIONES

Tema:

SISTEMA DOSIFICADOR AUTOMÁTICO DE ALIMENTO PARA ALEVINES DE TRUCHA, EN FUNCIÓN DE SU BIOMASA DETERMINADA POR UN SISTEMA DE VISIÓN ARTIFICIAL EN LA FINCA DEL SEÑOR ORTIZ DEL CANTÓN SALCEDO

Trabajo de Titulación Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Electrónica y Comunicaciones

ARÉA: Física y electrónica

LÍNEA DE INVESTIGACIÓN: Sistemas electrónicos

AUTORES: Mario Fabian Basantes Tisalema

Pedro Rodrigo Sasig Muso

TUTOR: Ing. Santiago Altamirano Meléndez Mg.

Ambato – Ecuador

septiembre - 2022

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación con el tema: SISTEMA DOSIFICADOR AUTOMÁTICO DE ALIMENTO PARA ALEVINES DE TRUCHA, EN FUNCIÓN DE SU BIOMASA DETERMINADA POR UN SISTEMA DE VISIÓN ARTIFICIAL EN LA FINCA DEL SEÑOR ORTIZ DEL CANTÓN SALCEDO, desarrollado bajo la modalidad Proyecto de Investigación por los señores Mario Fabian Basantes Tisalema y Pedro Rodrigo Sasig Muso, estudiantes de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que los estudiantes han sido tutorados durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, septiembre 2022.

Firmado electrónicamente por:
SANTIAGO MAURICIO ALTAMIRANO MELENDEZ

…….

Ing. Santiago Altamirano Meléndez, Mg.

TUTOR

AUTORÍA

El presente Proyecto de Investigación titulado: SISTEMA DOSIFICADOR AUTOMÁTICO DE ALIMENTO PARA ALEVINES DE TRUCHA, EN FUNCIÓN DE SU BIOMASA DETERMINADA POR UN SISTEMA DE VISIÓN ARTIFICIAL EN LA FINCA DEL SEÑOR ORTIZ DEL CANTÓN SALCEDO es absolutamente original, autentico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, septiembre 2022.

.....

Mario Fabian Basantes Tisalema

C.C. 0504151861

AUTOR

.....

Pedro Rodrigo Sasig Muso

C.C. 0550077689

AUTOR

APROBACIÓN TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por los señores Mario Fabian Basantes Tisalema y Pedro Rodrigo Sasig Muso, estudiantes de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DOSIFICADOR AUTOMÁTICO DE ALIMENTO PARA ALEVINES DE TRUCHA, EN FUNCIÓN DE SU BIOMASA DETERMINADA POR UN SISTEMA DE VISIÓN ARTIFICIAL EN LA FINCA DEL SEÑOR ORTIZ DEL CANTÓN SALCEDO, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, septiembre 2022.

Firmado electrónicamente por:
ELSA PILAR URRUTIA

……..……..……

Ing. Pilar Urrutia, Mg.

PRESIDENTA DEL TRIBUNAL

Firmado electrónicamente por:
ANDREA PATRICIA SANCHEZ ZUMBA

……..……..……

Ing. Andrea Patricia Sánchez, Mg.

PROFESOR CALIFICADOR

Firmado electrónicamente por:
MARLON ANTONIO SANTAMARIA VILLACIS

……..……..……..

Ing. Marlon Antonio Santamaria, Mg.

PROFESOR CALIFICADOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, septiembre 2022.

Mario Fabian Basantes Tisalema

C.C. 0504151861

AUTOR

Pedro Rodrigo Sasig Muso

C.C. 0550077689

AUTOR

DEDICATORIA

A Dios por darme la fortaleza para seguir adelante durante todo mi desarrollo profesional.

A mis queridos padres Silvia y Juan quienes me han apoyado de manera incondicional durante toda mi educación. Aunque ha habido momentos de debilidad ellos han sido los pilares fundamentales que me han permitido llegar hasta la etapa final de mi formación académica.

A mis queridos hermanos Carlos y Luis que me han brindado su apoyo en momentos donde más los necesitaba.

A mi familia en general, que siempre ha estado pendiente de mí.

Mario Fabian Basantes Tisalema

AGRADECIMIENTO

A la Universidad Técnica de Ambato por permitirme formarme en sus aulas de manera profesional. Un agradecimiento especial a la FISEI, por permitirme compartir conocimientos y experiencias con mis docentes y compañeros.

A nuestro tutor, ing. Santiago Altamirano que durante el desarrollo de este proyecto nos ha apoyado con sus conocimientos y consejos.

A la Familia Ortiz por recibirnos en su hogar. Gracias por su total apoyo y gentil acogida durante el desarrollo de nuestro proyecto.

A mi compañero de proyecto Pedro, por acompañarme durante todo el proceso de formación profesional hasta la etapa final. Muchas gracias.

Mario Fabian Basantes Tisalema

DEDICATORIA

A Dios por darme la fortaleza para seguir adelante durante todo mi desarrollo profesional.

A mis queridos padres Segundo Sasig y María Muso quienes me han apoyado de manera incondicional durante toda mi educación. Aunque ha habido momentos de debilidad ellos han sido los pilares fundamentales que me han permitido llegar hasta la etapa final de mi formación académica.

A mis queridos hermanos Cristian y Bryan que me han brindado su apoyo en momentos donde más los necesita.

A mi familia en general, que siempre ha estado pendiente de mí.

Pedro Rodrigo Sasig Muso

AGRADECIMIENTO

A la Universidad Técnica de Ambato por permitirme formarme en sus aulas de manera profesional. Un agradecimiento especial a la FISEI, por permitirme compartir conocimientos y experiencias con mis docentes y compañeros.

A nuestro tutor, ing. Santiago Altamirano que durante el desarrollo de este proyecto nos ha apoyado con sus conocimientos y consejos.

A la Familia Ortiz por recibirnos en su hogar. Gracias por su total apoyo y gentil acogida durante el desarrollo de nuestro proyecto.

A mi compañero de proyecto Mario, por acompañarme durante todo el proceso de formación profesional hasta la etapa final. Muchas gracias.

Pedro Rodrigo Sasig Muso

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

El presente proyecto de investigación está enfocado en el desarrollo de un dispensador de alimento para alevines de truchas. Este sistema, se complementa con un sistema de visión artificial para determinar la biomasa de los alevines y en función a ello, calcular la ración de alimento diario que deben recibir los peces.

El dispensador y el sistema de visión artificial están diseñados para ser compatibles con el internet de las cosas (IOT). Por lo tanto, se ha implementado tarjetas de desarrollo que permiten la conexión hacia la red como la ESP32 (integrada en el dispensador) y raspberry Pi (encargada del sistema de visión artificial). Esto permite al sistema de visión artificial enviar el número de alevines que ha contado, su biomasa y la rutina de alimentación hacia la base de datos Firebase una vez que ha culminado su tarea. El dispensador toma esos datos y los utiliza para empezar a dispersar el alimento según el horario definido. En adición, el usuario puede controlar y visualizar los parámetros del horario de alimentación del dispensador mediante una aplicación móvil desarrollada en React Native o mediante una plataforma web desarrollada en React. La configuración de las rutinas de alimentación puede ser editadas desde ambas plataformas web y móvil, según las necesidades del usuario.

La implementación de este sistema, permite aportar al ámbito de la piscicultura un dispensador de alimento confiable que cumple con la tarea de alimentar a los peces y, además, proporcionar parámetros adicionales como la cantidad de alevines y el peso de los mismos en caso de ser requerida por el usuario.

Palabras clave: IOT, dispensador de alimento, biomasa, piscicultura, visión artificial

ABSTRACT

The present research project proposes the development of a feed dispenser for trout fingerlings, this system is complemented by an artificial vision system to determine the biomass of the fingerlings and according to that information, calculate the daily food ration that the fish should eat.

The dispenser and the artificial vision system are designed to be compatible with the internet of things (IOT). Therefore, the dispenser has an ESP32 for its operation. Meanwhile, the artificial vision system works with a raspberry Pi. These development boards are compatible with internet connection. This allows the artificial vision system to send the number of fingerlings it has counted, their biomass and the feeding routine to the Firebase database once it has completed its task. The dispenser takes this data and uses it to start dispersing the food according to the defined schedule. In addition, the user can control and view the parameters of the dispenser's feeding schedule through a mobile application developed in React Native or through a web platform developed in React. Feeding routine settings can be edited from both web and mobile platforms, according to user needs.

The implementation of this system, it allows to contribute the field of pisciculture with a reliable feed dispenser that fulfills the task of feeding the fish and, in addition, provide additional parameters such as the number of fingerlings and their weight.

Keywords: IOT, food dispenser, biomass, pisciculture, artificial vision

CAPÍTULO I

MARCO TEÓRICO

1.1 Tema de investigación

Sistema Dosificador Automático de Alimento para alevines de trucha, en función de su biomasa determinada por un Sistema de Visión Artificial en la finca del Señor Ortiz del cantón Salcedo

1.2 Antecedentes investigativos

Para el desarrollo del proyecto de investigación se ha buscado información relacionada con la propuesta en repositorios de varias universidades, tanto a nivel nacional como internacional. También se ha tomado en cuenta artículos científicos relevantes con el tema. Por lo tanto, a continuación, se presentan las evidencias encontradas describiendo las características más importantes.

Ramos Javier, en la ciudad de Andahuaylas, en el año 2018, propuso un "Sistema de visión artificial para el conteo y medición de alevinos de trucha "arcoíris", para la dirección subregional de la producción Andahuaylas, 2018". Para la adquisición de imágenes se utilizó una cámara HD. El procesamiento de los fotogramas fue realizado en un computador Core i7. La interfaz HMI fue desarrollada en Matlab en conjunto con la herramienta Toolbox Image Processing. Durante el proceso, las imágenes son capturas a color y pasadas a escala de grises para obtener una imagen binaria. Con la imagen resultante se identifica los valores de los pixeles en cero y uno, información suficiente para determinar las características de tamaño y reconocimiento. Para determinar la precisión del sistema se ha utilizado un análisis estadístico a través de la desviación estándar. El sistema de conteo por visión artificial fue contrastado con los métodos de conteo manual directo, por volumen de agua y gravimétrico. Cada método tuvo un ensayo de tres repeticiones. Como resultado se obtuvo lo siguiente. Con el conteo manual el porcentaje de error es 2,73%, con el sistema de visión artificial ha cambiado a 1,53%. Con el mismo método el tiempo empleado era de 100% y con el sistema es 21,97%. La tasa de mortandad ha disminuido de un 6,26% al 0,93%. Por lo tanto, el sistema propuesto ha entregado mejoras significativas [1].

En la ciudad de Latacunga, en el año 2018, Ortiz Cristian, desarrolló el ¨Diseño e implementación de una máquina automática contadora de alevines para optimizar el tiempo y la fiabilidad de la producción para la empresa ACUIMAGG de la parroquia "Manuel Cornejo Astorga" en la provincia de Pichincha¨. El desarrollo del proyecto cuenta con un sistema embebido Nvidia Jetson TK1 que está enfocado a desarrollar el conteo por visión artificial. Para la adquisición de imágenes se ha usado una cámara SeeCam CU30 con una lente M12. La visualización del sistema se lo hace mediante un monitor conectado a la entrada HDMI del Jetson TK1. Cuando la imagen es adquirida esta entra en un pre procesamiento para una escala de grises. Como técnica de detección se ha utilizado el método Blob. Para el conteo se ha aplicado un filtrado Kalman el cual proporciona parámetros como la posición y velocidad de objetos. Estos parámetros serán analizados para el conteo cuando crucen por una línea horizontal haciendo posible el conteo. Una interfaz HMI ayuda al operador a realizar la calibración de la cámara, el conteo y registro de los alevines. Con la implementación de este proyecto se obtenido los siguientes resultados. El conteo de 60000 alevines de forma manual con una demora de 10 horas y seis personas generó un error del 2%. Mientras que, con la misma cantidad de alevines y con la implementación del sistema con visión artificial generó un error del 0.3% (sin feedback) con una demora de 5 horas y con una persona controlando el sistema. Con el nuevo método el error de conteo ha disminuido un 1.7%. El tiempo empleado se ha reducido un 50%. El porcentaje de error de conteo se puede disminuir mediante el feedback de verificación para calibrar el sistema de numeración. Por lo tanto, para un conteo menor a 100 alevines el error es 0%, mientras para valores mayores a 60000 el error es 0,003% [2].

En la parroquia Rionegro, en el año 2020, Osorio Julio, Vallejo Ferley y Echeverri Luis propusieron el diseño de un "Alimentador automático para perros con plataforma IoT" que busca disminuir problemas alimenticios en las mascotas mediante la planificación de la hora y cantidad de alimento según el dueño. El sistema es controlado mediante una ESP32-DevKitC. Para la sincronización de tiempo un sistema RTC. El alimento es dosificado mediante un tornillo sin fin en conjunto con un motor de 12v y 3A hacia una canastilla. Para determinar el peso correcto de alimento se usa un transductor Hx711. Cuando se haya alcanzado la cantidad definida por el usuario, un servomotor abre la canastilla y el alimento cae por gravedad. La ESP32 ha sido programada con lenguaje C++. Mediante un servidor en la nube, el dispositivo se

suscribe mediante el protocolo MQTT para recibir las notificaciones de tiempo y cantidad de alimento. La información de la mascota es ingresa mediante una plataforma web desarrollada con MEAN STACK. Como resultado se ha obtenido una precisión del 98,375 % en la dosificación de alimento, mediante un ensayo de 8 pruebas. El peso de las croquetas ha sido contrastado mediante una báscula WH-B05. Por otra parte, el sistema al carecer de una memoria EPROM para el registro de datos, no puede sincronizarse con las raciones definidas por el usuario, en caso de un reinicio y no haya posibilidad de conectarse a la red. Por lo tanto, en dicha circunstancia, entrega dosis inconsistentes de alimento a la mascota [3].

En la ciudad de Lima, Mejía Rodrigo y Rosales Gianfranco, en el año 2020, diseñaron un "Sistema de detección y clasificación de peces utilizando visión computacional". Para la adquisición de imágenes se utilizó una cámara Logitech c922 con una resolución de 1080/30 FPS y para el procesamiento Google Colaboratory (GPU Tesla K80 con 2496 CUDA Cores). Como algoritmo de visión de utilizó SURF y las técnicas de background subtraction y border detection como método para la identificación del pez. Con los bordes obtenidos de la imagen de estudio y la imagen genérica de un pez se procede hacer una comparación entre ambas mediante FLANN (Fast Library Approximate Nearest Neighbor) el cual busca comparar los puntos característicos de las imágenes dando como resultado la imagen que será usado para la selección. Para la clasificación de los peces se subió las muestras previamente obtenidas a Google Colaboratory. Con la finalización de la investigación se ha concluido que la fase de detección tiene una confiabilidad del 90% debido que el sistema puede fallar en caso de existir una imagen similar a un pez. Además, durante el proceso de clasificación se obtuvieron falsos positivos debido que se trabajó con un pequeño número de datos por lo que el equipo sugiere que para mejorar el entrenamiento hace falta implementar una base de datos más extensa [4].

Karningsih Putu, Kusumawardani Rini, Syahroni Nur, Mulyadi Yeyes y Saad Amna, en el año 2021 propusieron un "Automated fish feeding system for an offshore aquaculture unit". Este sistema está enfocado en dar apoyo a los operarios con la finalidad de mejorar la eficiencia de alimentación, costos de operación y el daño ambiental. El sistema de control está bajo una tarjeta de desarrollo Arduino, esta se encarga de recibir la señal de los sensores y activar/desactivar actuadores. El nivel de

los tanques es monitoreado con un sensor ultrasónico. Cuando el silo alcanza cierto punto se informa a los operarios con una alarma sonora. Para el control de la apertura del silo se usa servomotores. Para definir los horarios de alimentación, se utiliza una interfaz HMI y para el monitoreo por parte del usuario se aprovecha de una aplicación móvil para el envió de notificaciones. En conclusión, este articulo propone un prototipo para OceanFarmITS. El dosificador entrega dos tipos de alimento, el tipo pellet y alimento de restos de pescado. La alimentación automática está en función del número de peces, el peso promedio y la disponibilidad de alimento. Debido al cultivo de peces que es realizado en alta mar, este sistema busca reemplazar al operario para evitar el error humano y manejar de manera eficiente la alimentación y mejorar la salud de los peces con un mínimo costo [5].

1.2.1 Contextualización del problema

Actualmente, en el sector acuícola, específicamente en la producción de alevines de trucha es muy importante tener en cuenta el lugar y la alimentación de estos animalitos, cuando se tiene en un estanque como es en el caso del señor Ortiz del cantón Salcedo es muy importante dispersar su alimento en diferentes puntos del estanque [6].

La alimentación que deben llevar según la Organización de las Naciones Unidas para la Alimentación y la Agricultura debe ser bien estricta y mucho más en la etapa de los alevines que necesitan alimentarse entre 8 a 10 veces al día según su peso, en caso de existir un desbalance existirá pérdida de producción.

Este caso particular sucedió en la finca del señor Ortiz, al no contar con el tiempo necesario para alimentar a estos alevines, debido a que los integrantes de la finca tienen que cumplir otras actividades como: el trabajo, escuela, colegio y universidad se descuidaron y simplemente alimentaban 3 veces al día. Como aportación de la Ingeniería en Electrónica y Comunicaciones para la solución de este problema, se desarrolló un sistema de automatización que puede ser controlado de manera remota por un solo usuario, asegurándose así de entregar la cantidad correcta de alimento, al igual que el número de veces necesaria para la crianza de alevines de truchas.

1.2.2 Fundamentación teórica

Visión artificial

Imagen digital

Una imagen es una representación bidimensional de un mundo tridimensional. Se puede decir que una imagen es la representación de un objeto iluminado mediante una fuente radiante. Estos componentes que forman la imagen también se los denomina como iluminación $i(x, y)$ y reflectancia $r(x, y)$.

Para obtener una imagen digital intervienen varios elementos tales como:

- El objeto/escenario de interés.
- El sistema de iluminación.
- El sistema encargado de crear una imagen. Normalmente conformado por un sistema óptico y un sensor.

Una imagen digital puede ser representada mediante una matriz f de dimensiones NxM.

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,M) \\ f(2,1) & f(2,2) & \cdots & f(2,M) \\ \vdots & \vdots & \cdots & \vdots \\ f(N,1) & f(N,2 & & f(N,M) \end{bmatrix}$$

Donde cada elemento de la matriz representa los píxeles de la imagen que dan la iluminación a la imagen [7, pp. 11-12].

Vecinos de un pixel

Un pixel $\rho(x, y)$ tiene dos vecinos horizontales y dos verticales. También se los denomina $N_4\rho$ cada pixel tiene una distancia unitaria de (x, y). Además, tiene vecinos diagonales denominados como $N_D\rho$ [7, p. 13].

$(x-1, y+1)$	$(x, y+1)$	$(x+1, y+1)$
$(x-1, y)$	(x, y)	$(x+1, y)$
$(x-1, y-1)$	$(x, y-1)$	$(x+1, y-1)$

Figura 1: *Coordenadas de los vecinos del pixel (x,y)*

Fuente:[7, p. 13]

Etapas de la visión artificial

Para llevar a cabo el proceso de visión artificial se implementan una serie de pasos que involucran componentes tanto de hardware como de software.

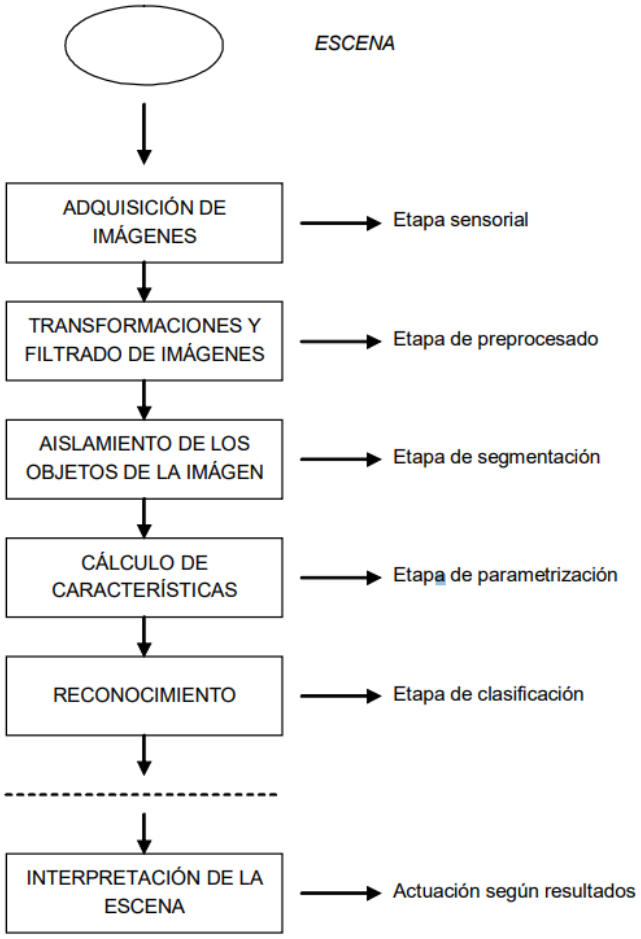


Figura 2: *Diagrama de bloques de un proceso de visión artificial*

Fuente:[7, p. 16]

La primera etapa consiste en la adquisición de la imagen. Para ello se utiliza dispositivos que sean capaces de digitalizar una imagen, como una cámara.

Durante la digitalización de una imagen se introducen ruidos. Estos pueden ser producidos por la naturaleza de la misma cámara o por factores externos como la iluminación. Con el fin de mejorar la imagen o resaltar sus detalles se implementa la etapa de preprocesamiento.

La siguiente etapa es la segmentación que consiste en aislar el objeto de interés del resto de la imagen capturada. Esta etapa es importante ya que dependerá el correcto funcionamiento del sistema.

Con la imagen segmentada se procede a interpretar sus fronteras o regiones. Esta etapa corresponde a la parametrización. La implementación de una o de otra dependerá del objetivo del sistema. Por ejemplo, las características de frontera son útiles cuando se necesita especificar las formas externas del objeto. Mientras la representación por regiones se centra en las características internas como la textura.

Para la etapa de clasificación se usan descriptores para asignar una etiqueta a cada objeto dentro de una escena. Finalmente, el sistema interpretará los resultados [7, p. 15].

Tipos de iluminación

La iluminación dentro de un sistema de visión juega un papel importante. El ojo humano es capaz de captar imágenes en ambientes donde haya poca luz. Por el contrario, dependiendo del tipo de cámara, no sucede lo mismo. Las cámaras pueden ser menos sensibles a la luz en comparación con el ojo.

Por lo tanto, para solucionar la falta de iluminación hay varias técnicas disponibles que pueden ser implementadas dependiendo del problema a solucionar [8, p. 11].

- **Iluminación frontal**

Es el tipo de iluminación más común. Donde se coloca la luz directamente sobre el objeto de interés. La cámara irá enfocada hacia el objeto. Para esta técnica se recomiendan aros de luz o iluminadores puntuales. Hay que mencionar, que esta técnica no es adecuada para escenarios con objetos con alta reflectancia [8, p. 12].

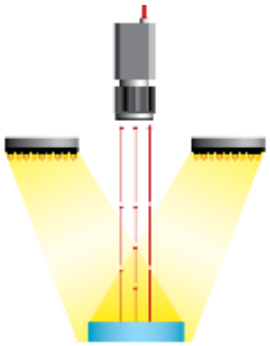


Figura 3: *Iluminación frontal*

Fuente: [8, p. 12]

- **Iluminación por campo oscuro**

La iluminación se coloca de forma perpendicular a la dirección de la cámara. Es utilizada para la detección de defectos superficiales como grietas o surcos [8, p. 13].

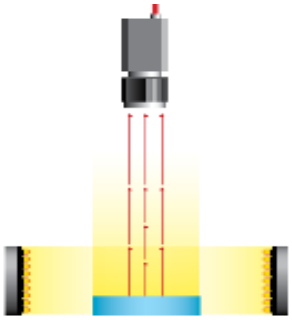


Figura 4: *Iluminación por campo oscuro*

Fuente: [8, p. 13]

- **Iluminación por contraste**

Para esta técnica de iluminación, el objeto de interés irá colocado entre la cámara y la fuente de luz. Es recomendado para objetos translúcidos con el fin detectar su silueta, manchas o grietas, aunque tiene como desventaja que no serán tomados en cuenta los detalles superficiales [8, p. 13].

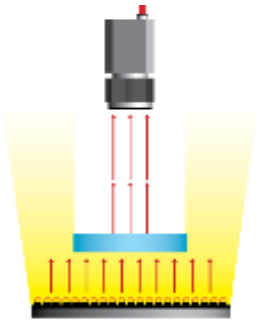


Figura 5: *Iluminación por contraste*

Fuente:[8, p. 13]

- **Iluminación difusa**

Para esta técnica, se utiliza un espejo semi transparente. Entonces una fuente de luz iluminará de manera lateral hacia el espejo, a la vez que desviará los rayos de luz al eje de la cámara. Generando así, una iluminación homogénea [8, p. 14].

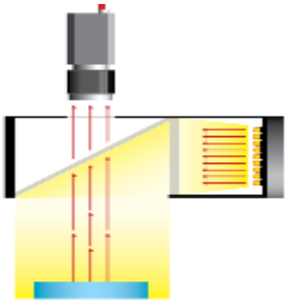


Figura 6: *Iluminación difusa*

Fuente: [8, p. 14]

- **Iluminación difusa tipo domo**

Se utiliza una cúpula esférica reflectante e iluminación coaxial. Con esto se aprovecha al máximo la iluminación difusa [8, p. 15].

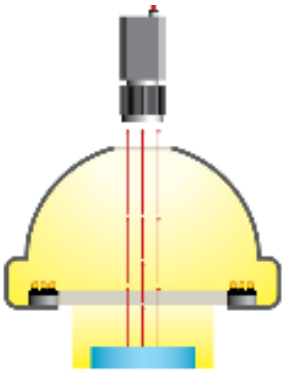


Figura 7: *Iluminación difusa tipo domo*

Fuente: [8, p. 15]

Filtrado

Las técnicas de filtrado más común que se suelen implementar en los sistemas de visión artificial es el filtrado espacial. Esta técnica consiste en realizar un recorrido en la imagen pixel a pixel mediante un kernel y realizar una operación matemática con los vecinos del pixel. Este conjunto de vecinos es denominado como ventanas y su tamaño puede ser de 3x3 o 5x5 [7, p. 42].

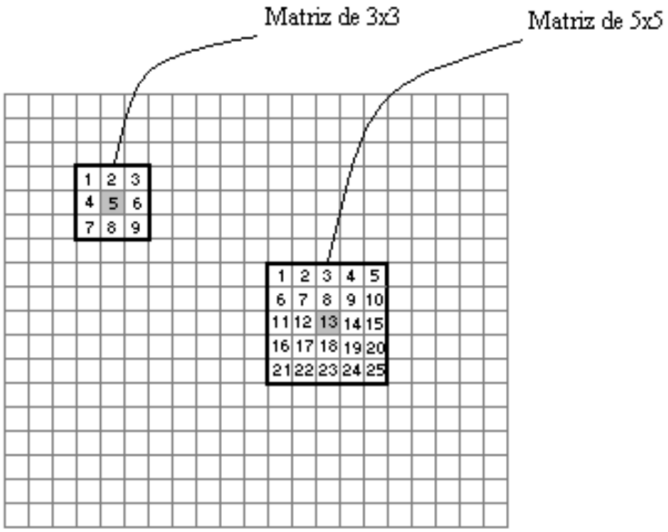


Figura 8: *Representación de una ventana 3x3 y 5x5*

Fuente: [7, p. 42]

- **Filtro pasa bajo**

Son utilizados para eliminar el ruido que está en el rango de la alta frecuencia dentro de una imagen. Para ello hay que definir la dimensión de la matriz (kernel) y el número de veces que se aplicará el filtro. Entre más iteraciones se haga, mejor será el filtrado, pero se tiende a perder la calidad de la imagen debido a la difuminación de la misma [7, p. 43].

Entre las matrices de convolución más comunes se tienen las siguientes.

$$h_a = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad h_b = \frac{1}{10} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad h_c = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- **Filtro de suavizado**

Esta técnica consiste en sustituir cada pixel por la media aritmética de sus vecinos y de él mismo. Por lo tanto, se toma un kernel de dimensiones NxN puntos. En donde el punto a sustituir irá en la posición central. El valor a sustituir será la media de la suma de todos los pixeles de la ventana. Se recomienda utilizar una ventana de dimensión N impar, para que el pixel central vaya en el centro [7, p. 43].

$$IMB(i,j) = \frac{\sum_{m=-\frac{N}{2}}^{\frac{N}{2}} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} IMA(i+m, j+n)}{\cdot N}$$

- **Filtro de moda**

Dentro de un conjunto de pixeles en una ventana consiste en calcular el valor que más se repita. Para un conjunto de pixeles con valores dispersos se tendrá una imagen dispareja. En cambio, si la imagen tiene un conjunto de pixeles similares, la imagen será más homogénea [7, p. 44].

Transformaciones morfológicas

Consiste en cambiar la forma y estructura de los objetos. Además, permiten separar objetos de interés dentro una imagen. Por lo general, se implementan estas técnicas en imágenes binarizadas [7, p. 55].

A continuación, se mostrarán las técnicas morfológicas más comunes.

- **Dilatación binaria**

La dilatación se denota $A \oplus B$ y se define como:

$$A \oplus B = \{x \in E^N \mid x = a + b \; para \; todo \; a \in A \; y \; b \in B\}$$

Donde A representa la imagen y B es el kernel formado por unos y ceros. Por lo tanto, la dilatación entre A y B es el conjunto de los valores de los desplazamientos de x tales que los valores del kernel B distintos de cero se solamente al menos una vez con los valores de A.

0	1	0
1	1	1
0	1	0

Figura 9: *Representación del kernel B*

Fuente: [7, p. 56]

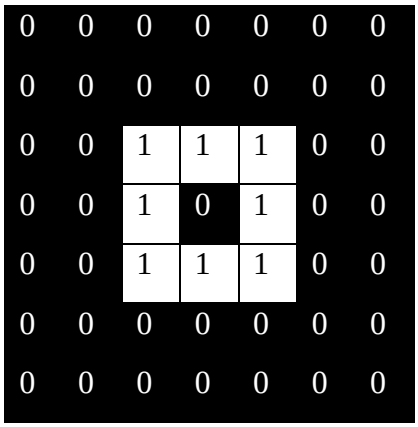


Figura 10: *Representación de la imagen A*

Fuente: [7, p. 56]

En el proceso de dilatación el kernel se va desplazando entre cada uno de los pixeles de la imagen A. Cuando al menos uno de los elementos de la matriz del kernel coinciden con un pixel del entorno. Entonces el pixel central donde se encuentra el kernel se pone a uno.

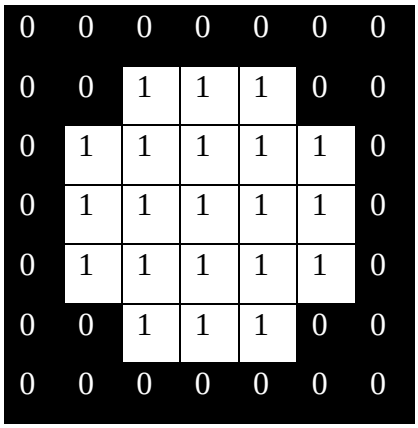


Figura 11: *Representación de la imagen dilatada $A \oplus B$*

Fuente: [7, p. 56]

Por lo tanto, al aplicar dilatación los bordes y contornos de los objetos tienen a aumentar de grosor. Su aplicación más común es para unir objetos separados después del filtrado [7, p. 56].

- **Erosión binaria**

La dilatación se denota $A \ominus B$ y se define como:

$$A \ominus B = \{x \in E^N \mid x = x + b \in A \; para \; todo \; b \in B\}$$

Se puede decir que la erosión es opuesta a la dilatación. Pero, eso no quiere decir que al aplicar erosión a una imagen y después aplicar dilatación de nuevo se va a obtener la imagen original.

Cuando la imagen es erosionada los bordes y contornos de los objetos se reducen. Se la utiliza para separar objetos que tienen contornos muy juntos.

Al igual que la dilatación, se utiliza un kernel B que irá recorriendo todos los pixeles de la imagen A [7, p. 57].

0	1	0
1	1	1
0	1	0

Figura 12: *Representación del kernel B*

Fuente: [7, p. 57]

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Figura 13: *Representación de la imagen A*

Fuente: [7, p. 57]

Cuando todos los puntos del kernel coincidan con los pixeles del contorno alrededor del pixel de interés, este se pondrá a uno, caso contrario se pondrá a cero.

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Figura 14: *Representación de la imagen A*

Fuente: [7, p. 57]

- **Apertura**

Para la apertura se utilizará las dos transformaciones morfológicas vistas anteriormente. Primero se aplica una erosión a la imagen, y después una dilatación. Aunque por intuición se pensaría que la imagen quedaría como al principio, esto no sucede. Ya que estas técnicas no son la inversa una de la otra [7, p. 58].

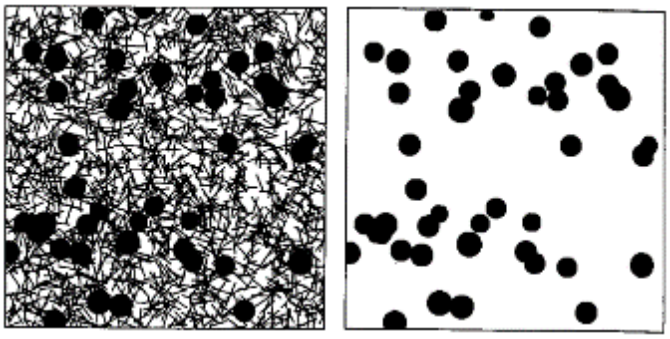


Figura 15: *De izquierda a derecha. Imagen con ruido e imagen después de aplicar la transformada de apertura*

Fuente: [7, p. 58]

- **Cierre**

Es el proceso contrario a la apertura. En donde se realiza primero una dilatación y después la erosión a la imagen [7, p. 58].

Técnicas para contar objetos

En la actualidad se cuentan con dos métodos para el conteo. Se pueden dividir en dos grupos.

- **Línea de interés (LOI)**

El procedimiento consiste en detectar un objeto en movimiento y realizar un seguimiento hasta que cruce una recta de referencia definida por el usuario [9, p. 2].

Figura 16: *Conteo por línea de interés*

Fuente: [9, p. 2]

- **Región de interés (ROI)**

Este proceso de conteo no requiere que el objeto este necesariamente en movimiento, sino que también puede contar objetos que se encuentran estáticos. Además, no requiere del seguimiento de los objetos, solo de su detección [9, p. 2].

Figura 17: *Conteo por región de interés*

Fuente: [9, p. 2]

A continuación, se muestra las ventajas y desventajas de cada estrategia de conteo.

Tabla 1: *Ventajas y desventajas sobre los métodos de conteo*

Método	Ventaja	Desventaja
ROI	• Conteo de objetos en movimiento y estáticos. • Puede contar objetos en oclusión. • Respuesta positiva ante multitudes de objetos.	• Requiere mayor tiempo de procesamiento. • Requiere el entrenamiento de un clasificador para determinar el aspecto de determinado objeto dentro de una escena.
LOI	• Conteo de objetos en movimiento. • Requiere menos procesamiento. • No requiere el entrenamiento de un modelo de detección.	• Los objetos estáticos no son contados. • Pésimo rendimiento ante la presencia de múltiples objetos. • Se ve afectado por la oclusión.

Fuente: [9, p. 2]

Seguidor de objetos

El seguimiento de objetos en una escena consiste en identificar a ese objeto dentro de los fotogramas posteriores también se lo denomina como tracking. Aunque parece un concepto sencillo, en la práctica abarca conceptos complejos.

A continuación, se muestra los principales conceptos relacionados con el rastreo de objetos [10, pp. 16-17].

- **Densidad de flujo óptico**

Este algoritmo permite identificar la dirección de vector un pixel en una imagen.

- **Caudal de flujo óptico**

Permite rastrear las características de los puntos principales de una imagen.

- **Filtro de Kalman**

Permite detectar la posición de un objeto en función de su movimiento previo.

- **Localización de objeto único**

Como su nombre lo indica, solo puede identificar un objeto a la vez. El objeto de interés es encerrado en un cuadro delimitador. Esta información de referencia es utilizada para rastrear al objeto en los fotogramas posteriores.

- **Localización de objetos múltiples**

Mediante un algoritmo de detección eficiente, se pueden rastrear varios objetos a la vez.

Algoritmos para el rastreo de objetos

- **Boosting Tracker**

Para la implementación de este algoritmo se necesita un entrenamiento previo mediante ejemplos positivos y negativos del objeto de interés. Internamente utiliza un detector de rostros (HAAR). También se puede seleccionar el objeto de interés mediante un cuadro delimitador. Este rastreador tiene una década de antigüedad, por lo tanto, no es utilizado actualmente debido a su poca eficiencia [10, p. 17].

- **TLD Tracker (Tracking, Learning and Detection)**

Este algoritmo contempla tres etapas. El rastreo a corto y largo plazo, el aprendizaje y finalmente la detección. Por ejemplo, cuando un objeto está siendo rastreado fotograma a fotograma. El detector toma en cuenta la información que se ha registrado hasta el momento, y en caso de corregir al rastreador este lo hace. Mientras tanto, en la etapa de aprendizaje se toma los errores del detector y es usada para evitar fallas en el futuro. Tiene como ventaja la posibilidad de rastrear objetos múltiples, aunque se presente oclusión. Aunque debido que puede dar falsos positivos no es muy utilizado [10, p. 19].

- **Medianflow Tracker**

El funcionamiento de este algoritmo consiste en rastrear al objeto tanto en los fotogramas previos como posteriores. El cálculo de la media de los errores permite minimizar este valor para seleccionar la trayectoria más confiable del objeto. Debido al informe de fallas se acopla perfectamente a escenarios donde el objeto de interés no presenta oclusiones y el movimiento es predecible. Aunque tiende a fallar cuando los objetos se mueven rápidamente [10, p. 20].

- **Goturn Tracker**

Este algoritmo está basado en redes neuronales convolucionales, por lo tanto, es más tolerable a cambios de iluminación pero, no es eficiente ante la oclusión [10, p. 21].

Sistema solar fotovoltaico

En la actualidad el aprovechamiento de energía solar en sectores alejados a una red eléctrica, es muy común observar el uso de paneles solares los cuales absorben dicha energía. Esta energía es consumida por diferentes cargas, pero para que esta energía solar se convierta en corriente eléctrica se utiliza algunos componentes. A continuación, se hablará de estos componentes que conforman un sistema solar fotovoltaico [11].

Componentes

Los componentes principales de un sistema solar se mostrarán en la tabla 2.

Tabla 2: *Componentes de un sistema solar fotovoltaico*

Componentes	Descripción
Placa o captador solar fotovoltaico	El uso de paneles solares en un sistema fotovoltaico ayuda a producir corriente constante y tensión continua.
Regulador	El regulador es un componente muy importante dentro de un sistema fotovoltaico, utilizado para la protección de la batería y control de flujo de la energía que circula.
Batería	Es un componente que complementa el sistema fotovoltaico, ya que es ahí donde se almacena la energía que es captada por los paneles solares.
Convertidor o inversor	El inversor es un componente primordial utilizado para convertir CC a AC
Carga	La carga en un sistema fotovoltaico son aquellos equipos que consumirán la energía de la batería, estos equipos pueden ser DC o AC.

Fuente: [11][12][13][14]

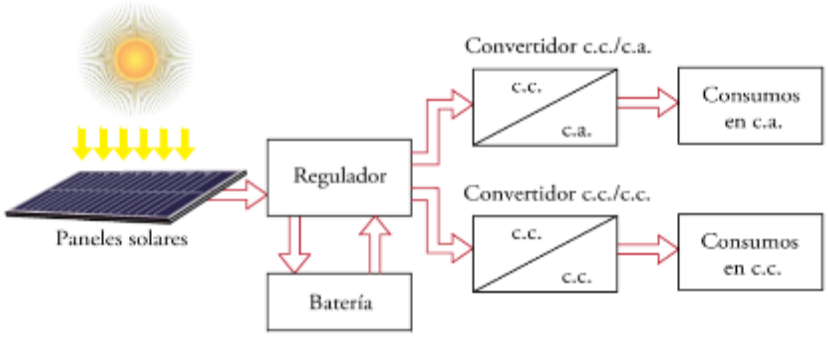


Figura 18*: Estructura de un sistema solar fotovoltaico*

Fuente: [11, p. 8]

Tipos de paneles

Es muy importante conocer que el componente principal de un sistema solar fotovoltaico es el panel solar, debido a que se encarga en captar y transformar la

energía solar a eléctrica. Es muy importante saber que un panel solar puede estar constituido entre 31 y 36 células solares, garantizando la carga efectiva de la batería [11].

A continuación, se presentará los diferentes tipos de paneles solares:

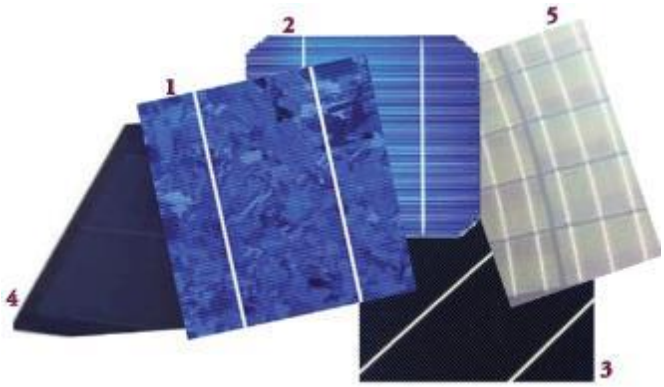


Figura 19: *Tipos de paneles solares*

Fuente: [11, p. 9]

1. Policristalino
2. Monocristalino
3. Monocristalino alta eficiencia
4. Silicio amorfo
5. Silicio amorfo semitransparente

Características físicas

Es muy importante saber las características físicas que brinda este componente de manera general, en la figura 20 se observa de manera grafica la estructura física de un panel solar [11].

- Cubierta exterior
- Encapsulante
- Parte posterior
- Marco soporte
- Garantía

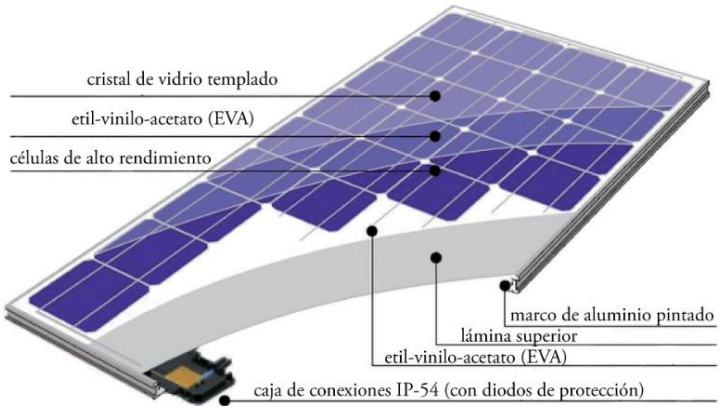


Figura 20: *Estructura física de un panel solar*

Fuente: [11, p. 13]

Cálculos

La fórmula utilizada para encontrar el voltaje que tiene cada célula del módulo solar se presenta en la siguiente ecuación [11, p. 14].

$$V_{cel} = \frac{V_{oc}}{36\ células}$$

Si $V_{oc} = 22.38$

El valor de tensión seria $V_{cel} = 0.62V$

Otro dato importante, es el cálculo de la eficiencia o también conocido como rendimiento máximo, para el cálculo del mismo se utiliza la siguiente ecuación [11, p. 14].

$$eficiencia = \frac{potencia\ máxima}{potencia\ radiación\ solar} * 100$$

Si la potencia máxima del panel solar es 135 y la potencia de radiación solar es 1000, el valor de la eficiencia será:

$$eficienia = 13{,}5\%$$

La fórmula para encontrar la potencia pico, es utilizada para saber cuánta potencia ocupará el sistema fotovoltaico[15].

$$P_p = \frac{E_g}{E_p}$$

Donde:

Pp = Potencia pico

Eg = Energía que debe generar o consumo

Ep = Energía que genera por cada vatio pico y está representada por $E_p = 0.9 * HSP$

0.9 = factor de rendimiento

HSP = Horas Sol Pico

Aplicación web

Evolución de las aplicaciones Web

Una aplicación Web con el pasar del tiempo ha venido evolucionando, y según el tipo de acceso la aplicación puede ser [16][17][18]:

- Estáticas
- Dinámicas
- Públicas
- Restringidas

Entre el año 1989-1997 apareció la estática o también denominada como Web 1.0 o Web de "solo lectura", es decir que no interactúa con el usuario, su desarrollo es con HTML y CSS. En el año 1997 se desarrolló otro tipo de Web denominada web dinámica o también como Web 1.5, son aquellas aplicaciones desarrolladas por algún lenguaje de programación como JavaScript y PHP, la comunicación se da entre el usuario y la aplicación es decir que es de "lectura/escritura". Su siguiente actualización es la web 2.0 o llamada colaborativa aparecerá en el año 2004, a partir de ese año hasta el año 2008 empezaron aparecer diferentes servicios como Facebook, Twiter, Youtube, Foros, etc [16][18][19].

La web 3.0 o también denominada como “semántica”, permite el uso de la nube o base de datos haciendo el sitio web más inteligente. De igual manera ocurre con la web 4.0 quien abarca la inteligencia artificial e implementación de asistentes de voz [18][20].

En la figura 21, se puede visualizar de manera resumida la evolución de las aplicaciones web.

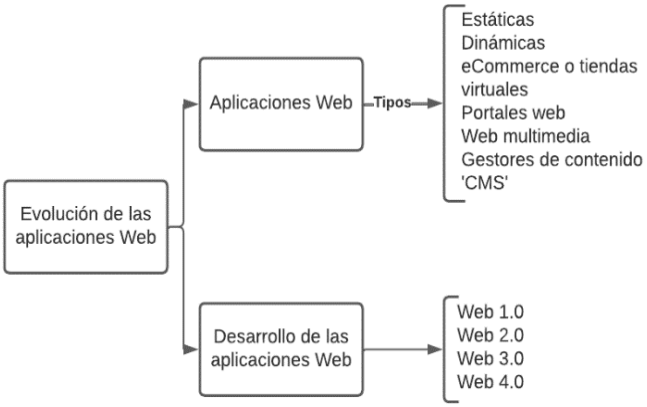


Figura 21: *Evolución de las aplicaciones Web*

Fuente: [16][18][19][20]

Arquitectura de la aplicación Web

La arquitectura básica de una aplicación web se divide en 3 partes. En la figura 22, se puede observar la arquitectura básica de la aplicación web [19][21].

- Uno o más clientes
- Una conexión de red
- Un servidor Web

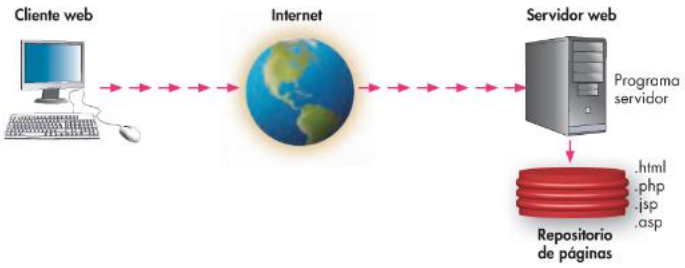


Figura 22: *Arquitectura básica de la aplicación web*

Fuente: [19]

Dentro del desarrollo de la aplicación Web, existen dos partes que caracterizan el funcionamiento del mismo. A continuación, se presentará estas 2 partes [22]:

- Front-end
- Back-end

FRONT-END

Front-end es la capa principal o parte frontal que se puede visualizar e interactuar de un sitio web, para que el Front-end pueda cumplir esas características puede hacer uso de [22]:

- **HTML**

HTML es un lenguaje de marcado que generalmente es utilizado para el desarrollo de una página web, cumple con funciones básicas y avanzadas para el desarrollo del mismo. En la actualidad su última versión es HTML5

- **CSS**

CSS es el encargado en la presentación de cualquier documento o página web creada en XML2, HTML, etc.

- **Bootstrap**

Bootstrap es una plantilla (framework) de código abierto que se utiliza para la creación de aplicaciones basado en CSS, HTML y JavaScript.

- **Angular JS**

Angular.js es una plantilla (framework) que se utiliza para la creación de aplicaciones basado en JavaScript de código abierto.

BACK-END

El Back-end es la segunda capa después del Front-end, se puede decir que es la parte no visible ya que es la parte lógica de la página web, dentro del Back-end se encuentra la comunicación con base de datos [22][23].

A continuación, se presentará una lista de los lenguajes y bases de datos que usualmente se encuentran dentro del Back-end.

- ASP.NET
- FrontPage
- Dreamweaver
- PHP
- Ruby
- Node.js
- Express
- MongoDB
- Firebase

Aplicación móvil

En la actualidad existen tres tipos de aplicaciones móviles, a continuación, se en listará e ilustrará cada uno de estos tipos [24, p. 2][25].

Tipos de aplicaciones móviles

- App nativa
- Aplicaciones web
- Aplicaciones híbridas

Figura 23: *Tipo de aplicaciones móviles*

Fuente: [24, p. 2]

Las aplicaciones web (WebApp) es desarrollada con HTML, Javascript y CSS, mientras que las App nativas son aquellas que se desarrollan para un determinado sistema operativo ya sea Androide o iOS utilizando un SDK o sus siglas en ingles" Software Development Kit", la siguiente aplicación son las hibridas, unión de Web App y App nativas, la característica principal de estas aplicaciones hibridas es que son capaces de ejecutarse en diferentes sistemas operativos móviles [24].

IOT

¿Qué es IoT?

IoT o en sus siglas en ingles "The Internet of Things" es el conjunto de dispositivos con inteligencia integrada que se comunican por medio de la red, esta red puede ser pública o privada. Los dispositivos que habitualmente se utilizan en IoT son sensores y actuadores, inteligentes [26][27].

A continuación, en la figura 24 se presentará la arquitectura general de la Tecnología IoT

Arquitectura IoT

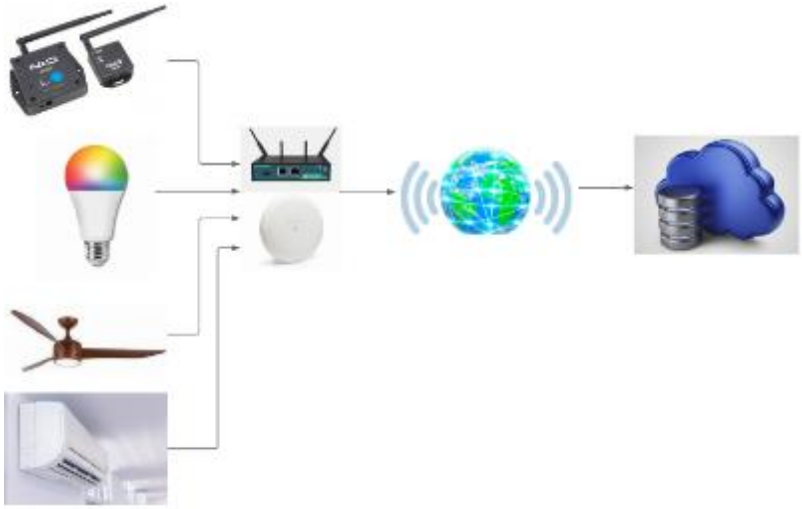


Figura 24: *Estructura de la plataforma IoT*

Fuente: Investigadores

La estructura de la plataforma IoT de la figura 24, se divide en cuatro etapas, a continuación, se describirá cada una de estas etapas.

- **Etapa 1 (Capa de percepción o de objetos)**

En la etapa 1 hace referencia a la capa física ya que se encuentran todos los dispositivos con inteligencia integrada, estos dispositivos pueden ser: Sensores, Focos inteligentes, Aire-acondicionados inteligentes, movimientos de un robot industrial, etc [28][29].

- **Etapa 2 (Capa de red o de abstracción)**

En la siguiente etapa se encuentra el método de comunicación, en este caso sería por medio de la compuerta Gateway. La función principal de una compuerta es transmitir los datos por medio de paquetes para que puedan ser leídos [30]. Es muy importante tener en cuenta que todo dispositivo inteligente necesita una conexión a internet.

Dentro de la capa de red, el encargado de enviar lo datos obtenidos por la capa 1 a la capa 3 es la capa de transporte garantizando su seguridad.

- **Etapa 3 (Capa de procesado)**

La siguiente etapa se encarga en el acceso a la nube ya sea para la lectura de datos o almacenamiento del mismo, dentro de esta capa, se encuentran tecnologías como módulos de procesamiento de Big Data y Cloud Computing [26].

- **Etapa 4 (Capa de aplicación)**

La última etapa se encarga en brindar el servicio de cada aplicación, por ejemplo: el funcionamiento de una lampara inteligente controlado por un móvil [29].

Protocolos que utiliza IoT

Los protocolos IoT ayudan con el intercambio de información. Es muy importante que los dispositivos que se utilizan tengan el mismo protocolo, para que el intercambio de información sea satisfactorio. A continuación, se presentara los protocolos más utilizados en IoT [31] [32].

- HTTP
- WebSocket
- XMPP
- CoAP
- MQTT

Aplicaciones IoT

IoT en la actualidad es muy utilizada en diversas áreas, en la tabla 3 se puede visualizar la aplicación de IoT en Áreas diferentes:

Tabla 3: *Aplicaciones IoT*

Áreas	Aplicaciones.
Hogar	El uso de la tecnología IoT en el área del hogar, es habitualmente el conjunto de dispositivos que permiten tener el control de la casa en tiempo real. Todos estos dispositivos son programados y controlados mediante un Smartphone
Industria y Comercio	IoT dentro de esta área puede aplicarse a diferentes necesidades como: • Publicidad personalizada. • Proveer el proceso de inventario. • Analizar el consumo de los clientes, etc.
Medio Ambiente	El uso de IoT dentro del área de medio ambiente, ayuda a mantener al tanto de la contaminación. Para el control del mismo puede hacer uso de sensores que ayuden a mantener las irregularidades y crímenes medioambientales.
Agricultura y Ganadería	De igual manera en la parte de la agricultura y ganadería, con ayuda de sensores se puede automatizar y monitorear el estado en que se encuentra cada planta en la parte de la agricultura y cada animal en la parte de la ganadería.
Salud	Los beneficios que presenta al utilizar IoT en el servicio de salud, ayuda con el control y prevención de enfermedades.

Fuente: [33]

Piscicultura

Este término hace referencia a la crianza exclusiva de peces, tales como truchas, tilapia, carpa, etc. Todo este proceso se lo realiza en ambientes controlados o semi controlados [34, p. 12].

Tabla 4: *Piscicultura según la densidad de carga y manejo*

Tipos	Extensiva	Semi intensiva	Intensiva
Descripción	Este tipo de cultivo se caracteriza por depender exclusivamente de alimentos naturales disponibles en el agua como plancton, organismos en suspensión, etc. Además, no se utilizan alimentos complementarios como el balanceado.	Se complementa el alimento natural que se encuentra en el agua, con balanceados bajos en proteínas. Este sistema de cultivo se implementa en especies omnívoras o planctívoros. El cultivo semi intensivo no es apropiado para truchas debido a su bajo contenido proteínico.	Las especies como las truchas necesitan balanceados ricos en proteínas, debido que son seres carnívoros. Por lo tanto, este tipo de cultivo hace el uso intensivo de alimentos altos en proteínas, sin ella la crianza de truchas se hace imposible.

Fuente: [34, p. 13]

Tabla 5: *Piscicultura según el número de especies*

Tipos	Monocultivo	Policultivo	Cultivo asociado
Descripción	Se cultiva una sola especie como la trucha, tilapia, etc.	Se cultiva dos o más especies diferentes en el mismo estanque. Por ejemplo, tilapia y camarón.	Con el fin de aprovechar al máximo el espacio de cultivo se pueden agregar especies no hidrobiológicos. Por ejemplo, crianza peces-patos.

Fuente: [34, p. 14]

Tabla 6: *Piscicultura según el número de especies*

Tipos	Comercial	De subsistencia
Descripción	Su producción genera ingresos económicos por su comercialización	La producción es a menor escala, y está destinado para el autoconsumo o para el intercambio con otros productos.

Fuente: [34, p. 14]

Densidad de siembra

La densidad de siembra se define como el número de alevines que se pueden colocar por metro cúbico. La cantidad de cultivo puede variar según la oxigenación del agua, temperatura, forma del estanque, tamaño de los ejemplares y el caudal.

Tabla 7: *Densidad de siembra por metro cúbico según la forma del estanque*

Tamaño del alevín	Cantidad máxima por metro cúbico	
	Estanque circular	Estanque rectangular
3 cm	7500	---
4 cm	4600	2300
5 cm	3400	1700

Fuente: [6, p. 13]

La densidad de siembra máxima en un estanque se puede aplicar si el agua cuenta con un sistema de aireación apropiado.

Métodos de conteo

En el área de la piscicultura, determinar el número de peces es fundamental para llevar un control y monitoreo de las especies que se están cultivando.

Como métodos de conteo de peces se pueden nombrar los más relevantes [35, pp. 3-4].

- **Conteo directo individual**

Este método consiste en transportar el alevín de un recipiente a otro de uno en uno. Aunque es el más preciso, su tiempo de ejecución es tardado.

- **Conteo por volumen de agua**

Este método se basa en el principio de Arquímedes. Por lo tanto, se relaciona la cantidad de agua de desplazada según el número de alevines.

- **Conteo por comparación visual**

Se toman dos recipientes similares. En uno se coloca una cantidad conocida de alevines, mientras en el otro recipiente se coloca la cantidad de alevines que se desea comparar para llegar a estimar la cantidad de la muestra.

- **Conteo por método gravimétrico**

Para este método se utiliza una balanza de preferencia electrónica para mayor precisión. Entonces un recipiente con un determinado número de alevines es pesado. El valor encontrado sirve de referencia para estimar el número de ejemplares de las muestras posteriores.

- **Conteo por máquina**

En áreas enfocadas a la comercialización de alevines, se utilizan sistemas de conteo más sofisticados, por ejemplo, con tecnología de led infrarrojo.

Alimentación

La trucha es una especie carnívora por excelencia, por lo tanto, necesita de una dieta adecuada para su cultivo. Una alimentación precisa garantiza un crecimiento uniforme.

Para una alimentación adecuada hay que tener en cuenta dos puntos importantes [36, pp. 51-52].

1. La selección del pellet adecuado para el cultivo está en función del pez más pequeño dentro de la población. Con esto se garantiza que todos coman y tengan un crecimiento uniforme.
2. Durante el suministro de alimento se debe procurar cubrir toda el área del estanque con el fin que todos los peces reciban el alimento al mismo tiempo.

Además, para mantener el alimento siempre fresco, hay que tomar en cuenta lo siguiente.

- Para evitar la proliferación de hongos en el alimento, este siempre deber almacenado en espacios libres de humedad y altas temperaturas.
- Un espacio libre de plagas como ratas e insectos evitará la contaminación del alimento.
- El almacenamiento del alimento durante cortos periodos de tiempo evitará la pérdida de nutrientes. Se recomienda un tiempo máximo de 3 meses.

Requerimientos nutricionales

Para un desarrollo óptimo del cultivo, se necesita un alimento balanceado. El alimento para peces contiene varios nutrientes como las proteínas que ayudarán en el crecimiento, mientras los lípidos le dará energía metabólica. Además, un buen balanceado contiene vitaminas (garantiza un buen crecimiento y previene enfermedades) y minerales (ayuda en la formación de dientes, huesos y sangre) adicionales.

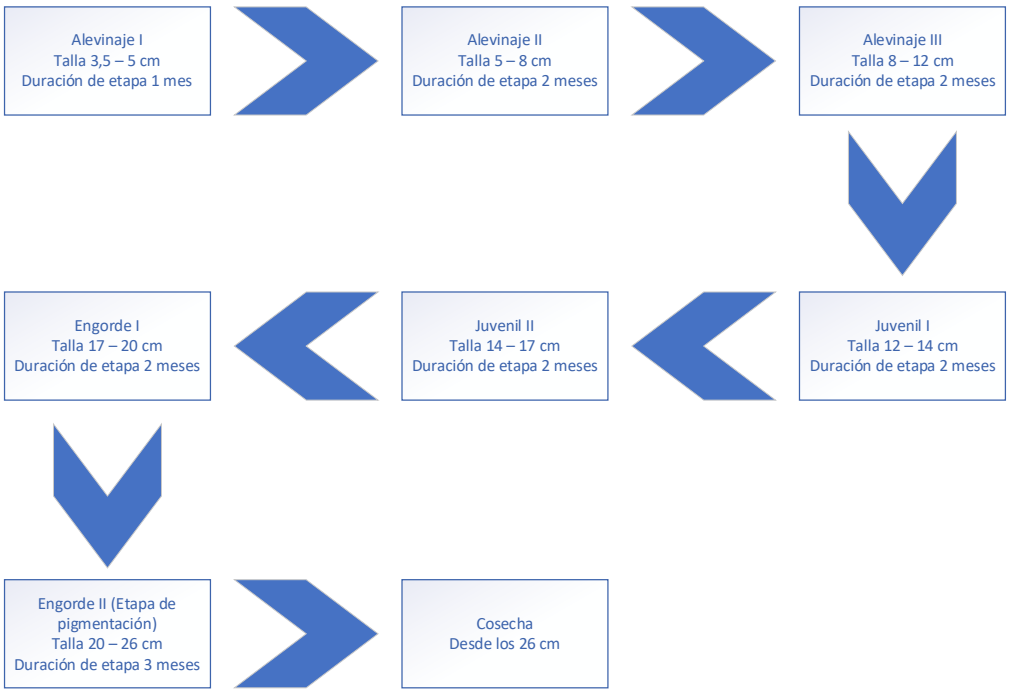


Figura 25: *Etapas de cultivo*

Fuente: [36, pp. 52-55]

Tabla 8: *Porcentaje nutricional durante cada etapa de la trucha*

Nutrientes	Alevines	Juveniles	Adultos	Reproductores
Proteínas (mínimo)	45	42	40	40
Carbohidratos (máximo)	22	24	25	25
Grasas (mínimo)	10	10	10	10
Minerales (máximo)	10	10	10	10
Humedad (máximo)	10	10	10	10
Fibra (máximo)	2	3	3	3
Calcio (mínimo)	1,5	1,5	1,5	1,5
Fosforo (mínimo)	1	1	1	1

Fuente: [37, p. 33]

Tamaño de los pellets según la etapa de la trucha

A continuación, se muestra los pellets y sus características más relevantes. Este tipo de alimento es el más común que se puede encontrar de manera comercial para cada etapa de la trucha.

Tabla 9: *Tipo de alimento durante cada etapa de la trucha*

Tipo de alimento	Peso unitario (gramos)		Tiempo de uso sugerido	Tamaño de partícula (mm)
	desde	hasta		
Trucha inicio I	Post - larvas	1	45 días	1,5 x 0,8 lento hundimiento
Trucha inicio II	1	5	55 días	1,5 x 2 lento hundimiento
Trucha crecimiento I	5	25	2 meses	2 x 3 flotante
Trucha crecimiento II	25	66,6	2 meses	3,5 x 4 flotante
Truchas engorde	66,6	Comercialización	4 meses	5 x 6 flotante
Truchas acabados pigmentado	100 - 130	Comercialización	45 a 60 días	5 x 5 flotante
Truchas reproductores	> 500	Fin de su ciclo de vida	2 a 4 años	9 x 5 flotante

Fuente: [36, p. 53]

Frecuencia de alimentación

Se puede definir como la ración dividida entre el número de veces que se alimentará al día el cultivo. Cada etapa implementa diferentes frecuencias de alimentación.

Tabla 10: *Frecuencia de alimentación según la etapa de la trucha*

Peso unitario (gramos)		Frecuencia de alimentación
desde	hasta	
Post - larvas	1	10 - 15
1	5	8 - 10
5	25	4 – 6
25	66,6	3 - 4
66,6	Comercialización	2 - 4
100 - 130	Comercialización	2 - 4
> 500	Fin de su ciclo de vida	2

Fuente: [36, p. 54]

Como se puede apreciar, a medida que la trucha va creciendo es alimentado con menos frecuencia durante el día.

Cálculo de ración

Determinar la cantidad adecuada de alimento permitirá no sobrealimentar y hacer mal uso del balanceado. Para encontrar la cantidad ideal a suministrar se hace uso de tablas de alimentación.

Tabla 11: *Tabla de alimentación para determinar la tasa alimenticia (%) en función de la temperatura del agua y longitud promedio del pez*

Longitud del pez (cm)	Temperatura ℃							
	8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
3	5,28	6,33	7,39	8,45	9,52	10,6	11,69	12,78
4	3,94	4,72	5,51	6,3	7,09	7,69	8,69	9,49
5	3,14	3,77	4,39	5,02	5,65	6,28	6,91	7,55
6	2,61	3,13	3,65	4,17	4,69	5,21	5,74	6,27
7	2,3	2,78	3,21	3,67	4,12	4,58	5,04	5,5
8	2,01	2,41	2,8	3,2	3,6	4	4,4	4,8
9	1,78	2,14	2,49	2,84	3,2	3,55	3,91	4,25
10	1,6	1,92	2,24	2,55	2,87	3,19	3,51	3,83

Fuente: [34, pp. 61-62]

Por lo tanto, para usar correctamente la tabla se necesita los siguientes datos

- Temperatura del agua
- Número de peces
- Talla promedio de los peces
- Biomasa total
- Tasa alimenticia (%)

El cálculo de ración diaria se puede hacer mediante la siguiente fórmula.

$$ración_{diaria} = Biomasa_{total} * tasa\ de\ alimentación\ (\%)$$

Para calcular la cantidad que hay que suministrar durante cada dosis de alimentación se divide la ración diaria entre la frecuencia de alimentación correspondiente para la etapa que se está cultivando.

$$Cantidad\ en\ cada\ dosis = \frac{ración_{diaria}}{frecuencia\ de\ alimentacón}$$

Para encontrar la temperatura del agua se recomienda llevar un registro diario en el horario de la mañana, medio día y tarde. Esto permitirá obtener el promedio de la temperatura diaria y observar su comportamiento [36, p. 12].

Tabla 12: *Formato recomendado para el registro de temperatura diaria*

Día	Horario de registro de temperatura ℃					Promedio diario
	8:00	10:00	12:00	14:00	16:00	
1						
2						
3						
.						
.						
.						
30						
Promedio Mensual						

Fuente: [36, p. 56]

Para entender de mejor manera el cálculo de la ración de alimento, a continuación, se planteará un ejercicio.

En un estanque se tienen 5000 alevines, con una longitud promedio de 4 cm. Su biomasa total es de 4300 gramos. Para el mes de junio se ha calculado una temperatura promedio mensual de 12 grados Celsius. Calcular la cantidad de alimento que se debe suministrar durante cada intervalo de alimentación.

Se tienen los siguientes datos.

- Número de peces = 5000
- Biomasa total = 4300 gramos
- Temperatura del agua = 12 ℃
- Talla promedio = 4 cm
- Frecuencia de alimentación = 8 (este valor corresponde a la etapa de alevinaje)

Desarrollo

Para calcular la cantidad de alimento, se necesita el porcentaje de la tasa de alimentación. Por lo tanto, con la temperatura y longitud promedio se procede a ubicar en que valor se cruzan estas dos variables en la tabla de alimentación.

Tabla 13: *Tabla de alimentación para determinar la tasa alimenticia (%) en función de la temperatura del agua y longitud promedio del pez*

Longitud del pez (cm)	Temperatura ℃							
	8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
3	5,28	6,33	7,39	8,45	9,52	10,6	11,69	12,78
4	3,94	4,72	5,51	6,3	7,09	7,69	8,69	9,49
5	3,14	3,77	4,39	5,02	5,65	6,28	6,91	7,55

Fuente: [36, pp. 61-62]

Según la tabla 13 la tasa de alimentación para este cultivo de peces es de 7,09 %.

$$Cantidad\ en\ cada\ dosis = \frac{Biomasa_{total} * tasa\ de\ alimentación\ (\%)}{frecuencia\ de\ alimentación}$$

$$Cantidad\ en\ cada\ dosis = \frac{4300 * 0{,}0709}{8} = 38{,}11\ gramos$$

Entonces, en conclusión, se puede decir que para un cultivo de 5000 alevines hay que suministrar 38,11 gramos 8 veces al día.

Horario de alimentación

Con el fin de establecer una rutina para acostumbrar al pez al ritmo de alimentación. Se recomienda alimentarlos en el horario de la mañana desde las 8:00 y antes del atardecer 17:00 [34, p. 54].

Condiciones ambientales

La trucha puede ser alimentado con un balanceado de excelente calidad. Pero esto no garantiza que se desarrolle adecuadamente o que el riesgo de mortandad disminuya. La alimentación solo es un factor de entre tantos que condiciona el desarrollo de los peces.

Las condiciones ambientales donde se cultiva la trucha juegan un papel importante en la producción de esta especie. Es tal su importancia que influye el apetito de los peces. A continuación, se mostrarán los factores más relevantes dentro de la piscicultura.

- **Oxígeno disuelto**

Como todo ser vivo, los peces también necesitan respirar, ellos mediante sus branquias captan el oxígeno disuelto en el agua. Las truchas son especies que consumen grandes cantidades de oxígeno. Esta demanda de oxígeno puede aumentar durante el proceso de alimentación y digestión. En las primeras etapas es donde llegan a consumir más oxígeno.

La temperatura y presión atmosférica también influyen en el oxígeno disuelto en agua. En consecuencia, la temperatura es inversamente proporcional al oxígeno. Es decir, que entre más alta sea la temperatura menos oxígeno habrá y viceversa. Por otro lado, la presión atmosférica es directamente proporcional a la cantidad de oxígeno [38, pp. 12-13].

Tabla 14: *Comportamiento de la trucha según la oxigenación del agua*

o^2 mg/litro	0 – 3	3,1 – 4,5	4,6 – 5,9	6 – 8,5
Condición	Muere	Sufre estrés.	Poco estrés. Crecimiento lento.	Óptimo desarrollo.

Fuente: [36, p. 12]

- **Temperatura**

Se puede decir que, dentro de la piscicultura enfocada a la trucha, la temperatura es una variable muy importante, ya que influye en el crecimiento y apetito de esta especie.

Tabla 15: *Comportamiento de la trucha según la temperatura del agua*

Temperatura ℃	1 – 3	4 – 8	9 – 14	15 – 17	18 - 20
Condición	Muere	Crecimiento lento.	Crecimiento óptimo. Buena incubación. Buena reproducción.	Velocidad de crecimiento disminuye.	Oxigenación del agua disminuye. Estrés.

Fuente: [36, p. 13]

Para temperaturas inferiores a 9 grados Celsius las truchas tendrán un crecimiento lento, que dentro de una producción comercial no es beneficioso. El mejor rango de temperatura oscila entre los 9 – 14 grados Celsius. Hay que mencionar que entre más aumente la temperatura el apetito del cultivo también lo hará, pero a partir de los 18

grados, el apetito tenderá a disminuir y el proceso de digestión será ineficaz [38, p. 16].

- **Potencial de hidrogeno (pH)**

Hace referencia a la acidez del agua. Además, ayuda en los procesos metabólicos de las truchas. La especie de los salmónidos, son sensibles a los cambios bruscos de pH. Por lo que, valores inferiores a 6,5 pueden provocar hemorragias en las branquias, elevando el porcentaje de mortandad. En cambio, un pH por encima de 9 genera un ambiente incompatible con la vida de la trucha [36, p. 13].

Tabla 16: *Comportamiento de la trucha según el pH*

pH	**4 – 5**	**5,1 – 6,5**	**6,6 – 7,9**	**8 – 10**
Condición	Mucho estrés. Crecimiento lento.	Estrés. Crecimiento lento.	Óptimo desarrollo.	Crecimiento lento. Muere

Fuente: [36, p. 13]

- **Transparencia del agua**

Este término hace referencia a la visibilidad a través del agua. Los materiales en suspensión, así como la presencia de organismos en la superficie puede provocar turbidez en el agua. Además, de reducir la actividad fotosintética, afecta a la visibilidad de los peces, provocando que no se alimenten correctamente [38, p. 17].

- **Dióxido de carbono**

Es producto de la respiración de los peces y de la descomposición de material orgánico. Su valor máximo permitido es 2ppm. Si se llega acumular en el ambiente del cultivo puede provocar que el oxígeno disuelto disminuya y los valores de pH se alteren [36, p. 13].

1.3 Objetivos

1.3.1 Objetivo general

Implementar un sistema dosificador automático de alimento para alevines de trucha, en función de su biomasa determinada por un sistema de visión artificial en la finca del señor Ortiz en el cantón Salcedo.

1.3.2 Objetivos específicos

• Analizar la alimentación de la trucha en etapa de alevinaje.

• Determinar las características y tecnologías de los dispositivos para el sistema de alimentación y visión artificial.

• Diseñar la estructura de los sistemas de dosificación y conteo de alevines.

• Diseñar los sistemas de control del alimentador automático y conteo de alevines.

• Desarrollar una plataforma IoT para el control del sistema de alimentación.

CAPÍTULO II

METODOLOGÍA

2.1 Materiales

Estructurales:

Acero inoxidable

Sistema dispensador de alimento:

- Sensor ultrasónico HC-SR04
- Celda de carga flexible
- ESP32
- Servomotor MG946R
- Motor DC 775 12V
- Motor a pasos NEMA 23
- Controladores para el driver y motor a pasos

Sistema fotovoltaico

- Regulador de carga
- Panel solar
- Batería 12v 7A

Sistema de visión artificial:

- Raspberry Pi 3
- Cámara pi camera v1.3
- Tiras led 12v
- Fuente de 12v
- LCD 16x2
- Pulsadores

2.2 Métodos

2.2.1 Modalidad de la investigación

El proyecto de titulación que se desarrollo está dentro de la investigación aplicada, debido que se usó el conocimiento adquirido durante la formación académica para aplicar en el sistema automatizado de alimentación y conteo de truchas(alevines) en la finca del señor Ortiz del cantón Salcedo.

El desarrollo del trabajo de titulación se realizó por medio de la investigación bibliográfica, debido a que se extrajo información relevante de: libros, artículos de revistas, revistas indexadas y base de datos de diferentes Universidades, con el único objetivo de tener una guía para el encaminamiento correcto del desarrollo del proyecto de investigación.

El trabajo de titulación se precisa como investigación de campo, debido a que se implementó un sistema dosificador automático en la finca del señor Ortiz, mejorando la alimentación y bienestar de la trucha.

El presente proyecto de titulación se define como investigación experimental, debido a que se realizó diversas pruebas para comprobar que el sistema automatizado funcione de manera adecuada. Además, se colocó la posición correcta del disco centrifugó para que disperse la comida de la trucha de manera distribuida, de igual manera, se ubicó en la posición correcta el sistema de conteo por visión artificial, por lo que este proceso tuvo varias etapas de prueba y error.

2.2.2 Recolección de información

La recolección de Información del presente trabajo de titulación, se utilizó diferentes artículos, libros, artículos académicos y base de datos de repositorios de las diferentes Universidades del país, que estuvieron relacionados con el tema de investigación.

2.2.3 Procesamiento y Análisis de Datos

Para el desarrollo del procesamiento y análisis de datos se llevó a cabo los pasos descritos a continuación:

- Interpretación y optimización de la información.
- Análisis de los dispositivos que se van a emplear.

- Estudio de la APP para la automatización del sistema
- Presentación de los resultados de acuerdo con los objetivos planteados

2.2.4 Desarrollo del Proyecto

Para cumplir con los objetivos planteados se realizó las siguientes actividades:

- Análisis de los tipos de alimento para alevines de trucha existentes en el mercado
- Identificación de la cantidad de raciones de alimento
- Análisis de la frecuencia de alimentación
- Identificación de los factores ambientales que afectan la dosificación de alimento
- Análisis de las tarjetas de desarrollo compatibles con el proyecto
- Selección de los componentes de sensorización del sistema
- Identificación del tipo de cámara adecuada para la adquisición de imágenes
- Análisis de los actuadores necesarios para el proyecto
- Análisis de los parámetros físicos de las estructuras
- Selección de los materiales compatibles con los requerimientos del proyecto
- Diseño mediante software de las estructuras de los sistemas
- Configuración de las tarjetas de control del dosificador de alimento
- Diseño de algoritmos del sistema de visión artificial de conteo de alevines
- Configuración de la base de datos del sistema de alimentación
- Desarrollo de una interfaz web para el sistema de alimentación
- Diseño de una aplicación móvil
- Implementación de los sistemas diseñados

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1 Análisis y discusión de los resultados

3.1.1 Desarrollo de la propuesta

Diseño del contador

Consideraciones tomadas en cuenta para el diseño del contador

1. La construcción y diseño deben ser simples.
2. Debe ser compacto y ligero para su fácil transporte.
3. Los componentes deben ser reemplazables en caso de daño.
4. El material de la estructura no debe ser agresivo con los alevines.
5. Los peces se deben deslizar lo más vertical posible ante la cámara para determinar la biomasa correcta.

Selección del método de conteo de alevines

El método de conteo determina como serán manipulados los alevines para que sean contados.

Tabla 17: *Tabla comparativa para la estructura del contador*

Características	Pendiente con flujo de agua	Pendiente con caída por gravedad
El alevín puede nadar en contracorriente	Debido al flujo de agua, el alevín puede retornar al lugar de inicio donde fue lanzado.	Debido a la pendiente y sin tener un flujo de agua constante, el alevín se puede deslizar hasta la salida del sistema.
Permite regular el flujo de agua	El flujo de agua puede ser regulado.	El flujo de agua depende de la cantidad que entre junto con el alevín.
Sistemas adicionales	• Bomba sumergible • Bomba de agua	No requiere sistemas adicionales.
Precisión durante el conteo	Debido que hay la posibilidad de que el alevín regrese al inicio de lanzamiento puede provocar falsos positivos.	El alevín al deslizarse por gravedad no hay probabilidad que retorne por lo tanto los errores serian minimizados.

Fuente: Investigadores

Justificación

La pendiente mediante flujo de agua es una buena opción para el contador, pero los alevines son animales que están acostumbrados a desarrollarse en ambientes con corrientes de agua por lo que, al tratar de imitar su hábitat, durante el proceso de conteo pueden retornar al lugar de inicio donde fueron lanzados provocando imprecisión en los resultados finales. Por lo tanto, se ha implementado el sistema de conteo de pendiente de caída por gravedad porque al no tener un flujo de agua constante los alevines no tienen otra opción que seguir hasta la salida de la estructura del sistema.

Desarrollo

Para el proceso de conteo y determinación de biomasa se necesitó que el alevín pase de manera vertical ante la cámara. La razón de esta condición es porque la biomasa del pez está relacionada con su talla (alto del recuadro delimitador), por lo que si se desliza en posiciones diferentes a la vertical la biomasa será incorrecta.

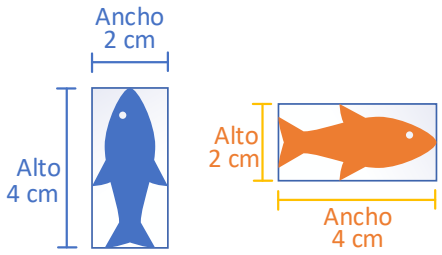


Figura 26: *De izquierda a derecha. Posición vertical del pez y pez en posición horizontal*

Fuente: Investigadores

Por ejemplo, en la figura 26, se puede apreciar el pez en posición vertical, esta es la posición ideal para medir la biomasa. Por el contrario, si el pez está en posición horizontal el ancho se convierte en alto, por lo tanto, la biomasa calculada será incorrecta. A modo de ejemplo se puede decir que el pez de la figura superior en posición vertical tiene un tamaño de 4 centímetros de alto que, según las tablas relacionales entre talla (alto) y peso, corresponde a 0,86 gramos. Por el contrario, cuando el pez está horizontal su alto será de 2 centímetros y tendrá un peso de 0,11 gramos. Siendo este último valor incorrecto.

Para hacer que los alevines pasen de manera vertical en todo momento, se colocó canales.

Tabla 18: *Anchura (en vertical) de una muestra de 10 alevines*

# Alevín	1	2	3	4	5	6	7	8	9	10
Ancho (cm)	0,7	0,6	0,6	0,7	0,8	0,8	0,9	0,7	0,7	0,6

Fuente: Investigadores

De la muestra tomada, se puede apreciar que el mayor valor es 0,9 centímetros. Para el ancho de los canales de tomará el mayor valor más el 70% del mismo para mantener un margen de seguridad. Por lo tanto, el ancho de los canales será de 1,53 cm. Para el alto de los canales se tomará el mismo valor.

Para definir el número de canales, hay que tener en cuenta que, a medida que el objeto de interés se aleja del centro de la cámara y se mueve más hacia los espacios laterales, sus dimensiones empiezan a cambiar ligeramente y también la altura de los canales no permitirá detectar correctamente al alevín.

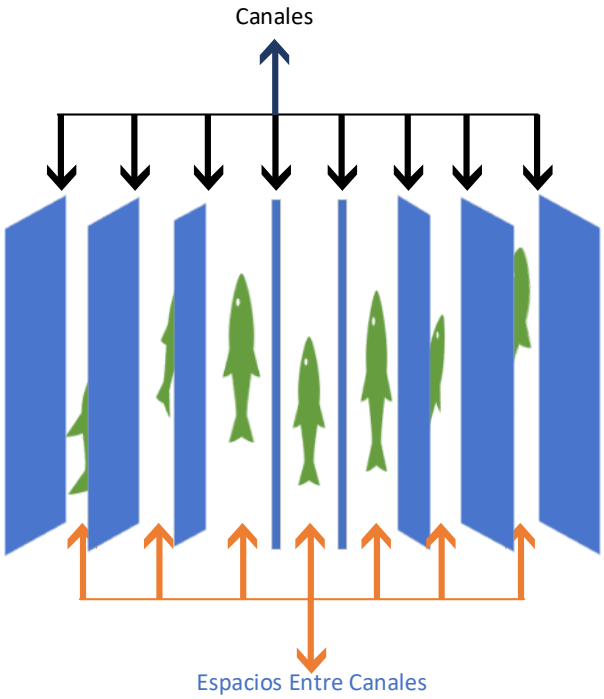


Figura 27: *Representación de la pérdida de visión en los espacios laterales*

Fuente: Investigadores

En la figura 27, se ha representado como los peces son obstruidos por la altura de los canales por lo que entre más canales se coloquen menos visibilidad tendrá la cámara. Por lo tanto, se dejó solamente 5 canales con la finalidad de capturar la imagen correctamente.

Para la longitud de la rampa deslizante, se necesita una distancia corta para que los alevines no pasen mucho tiempo fuera del agua. Por lo que, la pendiente tendrá una longitud de 50 centímetros.

Con el objetivo de no permitir el ingreso de la luz exterior y solo depender de la iluminación interior para mantener un ambiente controlado se ha optado por una altura de la estructura de 15 centímetros. Además, esto permite un sistema compacto.

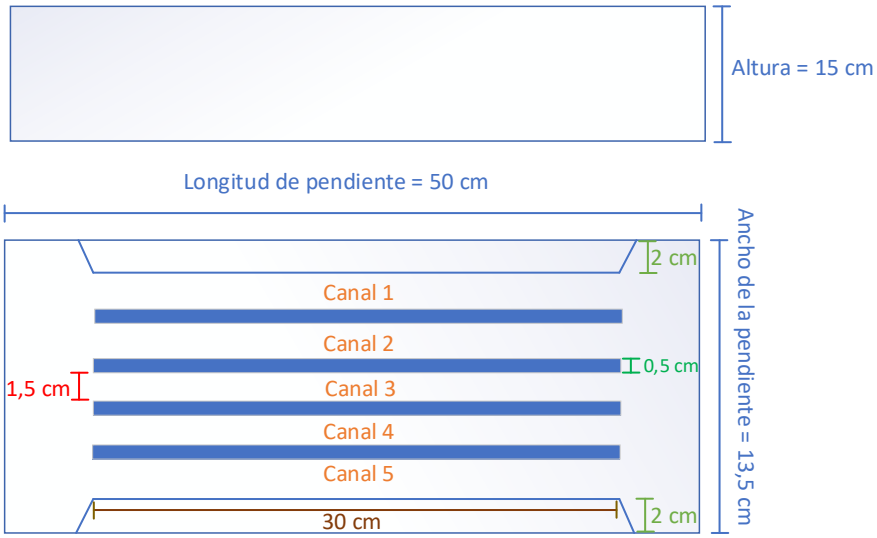


Figura 28: *De arriba hacia abajo. Vista lateral izquierda y vista superior interior del contador*

Fuente: Investigadores

En resumen, la pendiente de la estructura tendrá las siguientes dimensiones:

$$Largo_{pendiente} = 50\ cm$$

$$Ancho_{pendiente} = Espacio\_entre_{canales} + Ancho_{canales} + Espacios_{laterales}$$

$$Ancho_{pendiente} = 1{,}5 * 5 + 0{,}5 * 4 + 2 * 2 = 13{,}5\ cm$$

$$Altura_{pendiente} = 15\ cm$$

Selección de la cámara

La cámara será la encargada de capturar las imágenes que serán utilizadas por el sistema de visión artificial. Su posición será perpendicular al plano de la escena para no deformar la imagen. Además, se encuentra ubicado sobre el canal central para dejar espacios simétricos a cada lado de la escena de interés.

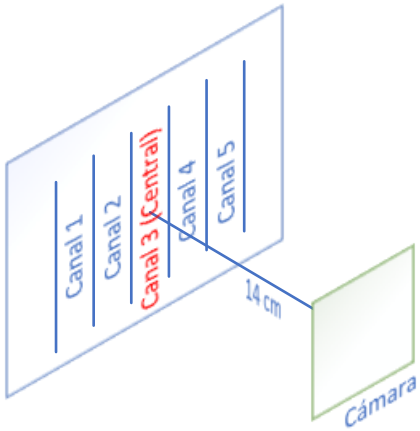


Figura 29: *Posición de la cámara*

Fuente: Investigadores

Para calcular el ángulo necesario para abarcar el ancho de la escena de 13,5 centímetros, se hará uso de trigonometría.

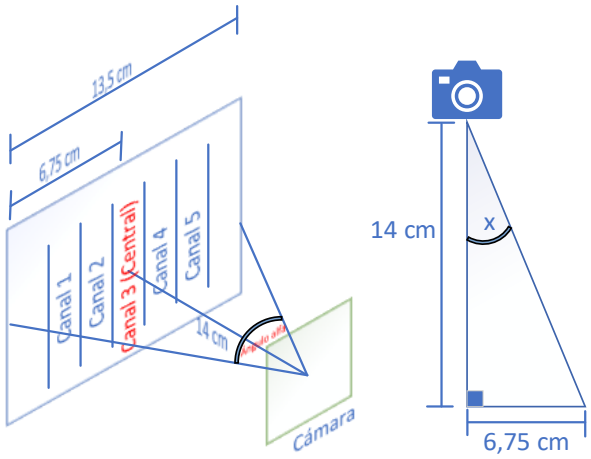


Figura 30: *Triángulo rectángulo formado entre la cámara y la escena*

Fuente: Investigadores

Por lo tanto, se tiene lo siguiente

$$\alpha = 2 * tan^{-1}\left(\frac{cateto\ opuesto}{cateto\ adyacente}\right)$$

$$\alpha = 2 * tan^{-1}\left(\frac{6{,}75}{14}\right) = 51{,}48^{\circ}$$

Entonces, para cubrir una escena de 13,5 centímetros se necesita que la cámara tenga un ángulo de visión horizontal de al menos $51{,}48^{\circ}$

Tabla 19: *Tabla comparativa de las cámaras para el sistema de visión artificial*

Características	Pi CAMERA V1.3	Cámara Ojo de Pez Para Raspberry PI	ARDUCAM Sony IMX219
Representación gráfica			
Resolución	5 MPX	5 MPX	8 MPX
Ángulo de visión horizontal	54˚	170˚	70˚
Distancia focal	3,6 mm	1,7 mm	25 mm
Tamaño del sensor	$1/4''$	$1/4''$	$1/4''$
Resolución de video	1080p a 30 FPS 720p a 60 FPS 480p a 90 FPS	1080p a 30 FPS 720p a 60 FPS	1080p a 30 FPS 720p a 60 FPS 480p a 90 FPS
Conector	CSI	CSI	CSI

Fuente: Investigadores

Según la tabla comparativa todas entran en el rango de posibles opciones porque tienen el ángulo de visión mínimo requerido.

Por lo tanto, se tuvo que determinar si todas las cámaras logran abarcar el ancho de la escena de 13,5 centímetros. Para ello se hizo uso de la siguiente fórmula.

$$ancho_{escena} = \frac{S * D}{f}$$

Donde:

$$S = tamaño\ del\ sensor\ en\ mm$$

$$D = distancia\ entre\ la\ cámara\ y\ la\ escena\ en\ mm$$

$$f = distancia\ focal\ en\ m$$

Pi CAMERA V1.3

$$ancho_{escena} = \frac{6{,}35mm * 140mm}{3{,}6mm} = 24{,}69\ cm$$

Cámara Ojo de Pez Para Raspberry PI

$$ancho_{escena} = \frac{6{,}35mm * 140mm}{1{,}7mm} = 52{,}29\ cm$$

ARDUCAM Sony IMX219

$$ancho_{escena} = \frac{6{,}35mm * 140mm}{25mm} = 3{,}56\ cm$$

Justificación

Según los cálculos, la cámara ARDUCAM Sony IMX219 queda descartada porque solo tiene un campo de visión horizontal de 3,56 centímetros, muy por debajo del ancho de la escena de 13,5 centímetros. Por otro lado, la cámara de raspberry con lente ojo de pez puede abarcar casi cuatro veces el ancho de la escena, el problema viene por el gran angular de 170 grados que provocará deformación en los laterales de la imagen capturada. Por lo tanto, como mejor opción, se tiene la Pi CAMERA V1.3 que está dentro de los límites para capturar el ancho de la escena y puede abarcar un ángulo de visión horizontal de 54°, suficiente para cubrir el ángulo calculado previamente.

Selección de la tarjeta de desarrollo para el contador

Esta tarjeta será la encargada de realizar todas las funciones relacionadas con el procesamiento de video en tiempo real y la interpretación de datos para el conteo y determinación de biomasa de los alevines.

En el mercado se ha podido encontrar las siguientes SBC.

Tabla 20: *Tabla comparativa de los SBC*

Características	Banana Pi M1	Raspberry Pi 3B	Orange Pi PC
Representación gráfica			
Procesador	Allwinner A20 Dual-core Cortex-A7 1.0GHz CPU	Broadcom Quad Core BCM2837, Cortex-A53 (ARMv8) 1.2 GHz	Quad-core Cortex-A7 1.6 GHz
GPU	Mali-400 MP2 Dual Core	Broadcom VideoCore IV Dual Core 400MHz	Mali-400 MP2 Dual Core
RAM	1GB DDR3	1GB LPDDR2	1GB DDR3
Almacenamiento	Tarjeta microSD hasta 32 GB	Tarjeta microSD hasta 32 GB	Tarjeta TF hasta 32 GB
Conectividad inalámbrica	No disponible, únicamente por cable de red	IEEE 802.11.b/g/n Bluetooth 4.1	No disponible, únicamente por cable de red
GPIO	26 pines	40 pines	40 pines
Sistema operativo	Android, Ubuntu y Raspbian	Raspbian	Android Ubuntu
Voltaje de alimentación	5 VDC	5 VDC	5 VDC
Corriente	2A	2.5A	2A
Salida de video	HDMI, CVBS, LVDS/RGB	HDMI	HDMI
Interfaz para cámara y video	CSI soportado DSI soportado	CSI soportado DSI soportado	CSI soportado

Fuente: Investigadores

Justificación

Las tarjetas tienen características casi similares. En cuanto al sistema operativo que soportan todas pueden correr distros basadas en Linux. La conectividad inalámbrica es una característica en donde Raspberry se destaca, las demás tarjetas únicamente se pueden conectar por cable de red, siendo una gran desventaja. Tanto Banana como Orange tienen se destacan por tener una mejor velocidad de transferencia de datos por sus memorias DDR3 en comparación con Raspberry, pero se quedan cortos en el procesador. Raspberry por otra parte cuenta con una CPU Cortex-A53 de cuatro

núcleos, a pesar que Orange tiene un procesador casi similar, Cortex-A7 tiene menos rendimiento. Por lo tanto, se ha usado la SBC Rasberry debido a que tiene un mejor procesador y es compatible con tecnologías inalámbricas.

Selección del tipo de iluminación

La correcta iluminación hará que el área de interés dentro del sistema de visión artificial no requiera del preprocesamiento de imágenes para la corrección y filtrado aligerando la carga de trabajo.

Tabla 21: *Tabla comparativa de las técnicas de iluminación*

Tipo de iluminación	Descripción	Ventaja	Desventaja
Frontal	La cámara y la iluminación, se posicionan sobre el objeto.	Iluminación directa sobre el objeto de interés reduciendo sombras.	Produce brillos sobre superficies reflectantes.
Lateral	La iluminación se ubica de manera paralela al objeto a iluminar y perpendicular a la cámara.	Resalta los detalles del objeto.	Se producen sombras o áreas quemadas.
Difusa	Mediante la iluminación indirecta se consigue una luz suave.	Reducción de brillos. Reducción de sombras.	La superficie se difumina. No es adecuada para espacios reducidos. Requiere de un sistema de espejos para generar una iluminación uniforme.
Contraste	El objeto de interés se coloca entre la cámara y el sistema de iluminación.	Los bordes quedan bien definidos.	Los detalles de la superficie no son eliminados.

Fuente: Investigadores

Justificación

La iluminación por contraste ayuda a definir los bordes de los objetos de interés, que para el sistema de conteo y biomasa son suficientes pero el problema se complica en el diseño y construcción al tratar de acoplar la luz en la parte inferior del contador, por lo tanto, queda descartada. La iluminación difusa introduce defectos en la imagen por lo que hay que implementar filtros haciendo que el sistema se cargue de una tarea adicional ralentizando los otros procesos. Además, no es adecuada para espacios reducidos por lo que no se acopla con la estructura compacta que se necesita. Las

técnicas restantes tienen desventajas con respecto a las sombras y exceso de brillos. Por lo tanto, hay que combinar ambas técnicas para complementarlas. Así, para evitar el exceso de brillos se coloca la iluminación frontal separado de la cámara y para resaltar los detalles del objeto se utiliza la iluminación lateral.

Diseño del dispensador de alimento

Consideraciones tomadas en cuenta en el diseño del dispensador

1. La construcción y diseño deben ser simples.
2. El alimento para alevines no debe estar almacenado durante largos periodos de tiempo.
3. Los componentes del dispensador deben ser fáciles de reemplazar en caso de sufrir daños
4. Los componentes deben trabajar con voltajes inferiores o iguales a 12 VDC para que sean compatibles con sistemas de generación de energía autónomos.

Desarrollo

Selección de material del dispensador

La selección del material es muy importante para mantener el balanceado fresco. También hay que tomar en cuenta si el material es resistente a los golpes y factores ambientales. Los materiales más comunes que se pueden encontrar son los siguientes.

Tabla 22: *Tabla comparativa de los materiales para la estructura del dispensador*

Características	Acero inoxidable	Policarbonato compacto	PVC compacto
Resistencia al agua	Material completamente inmune al agua.	Material impermeable.	Material impermeable.
Descomposición del material con el paso del tiempo	Dependiendo del tipo de acero no sufren cambios físicos en su estructura.	En contacto con sustancias químicas pueden llegar a perder sus propiedades físicas.	Con el paso del tiempo puede llegar a volverse amarillento.
Método de adhesión	• Soldadura • Adhesivo epoxi	• Pegamento especializado • Tornillos	• Pegamento especializado • Tornillos
Resistencia a golpes	Alta resistencia ante daños físicos.	Resistencia media (su calidad puede mejorar según los métodos de fabricación).	Material delicado que puede llegar a romperse (su calidad puede mejorar según los métodos de fabricación).
Costo	Dependiendo del espesor de la lámina y del tipo, su costo oscila desde lo económico hasta altos precios.	Más económico con respecto a otros materiales.	Más económico con respecto a otros materiales.
Limpieza	El pulido electroquímico permite que la superficie se limpie fácilmente.	Al tener una superficie lisa se puede limpiar su superficie fácilmente.	Al ser un producto plástico su superficie puede llegar a ser manchada.

Fuente: Investigadores

Justificación

Mediante la tabla comparativa, se llegó a la conclusión que el mejor material para la estructura es el acero inoxidable. Debido que dentro del dispensador se va almacenar balanceado, por lo que se requiere un material que no se descomponga con el paso del tiempo. El acero inoxidable permite acoplar sus partes mediante soldadura en contraposición con los demás materiales que requieren de pegamento que con el tiempo pueden llegar a contaminar el alimento. La estructura también requiere de resistencia ante daños físicos por lo que el acero puede satisfacer esta necesidad. En cuanto al método de limpieza se puede hacer fácilmente sin llegar a usar productos químicos.

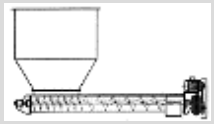
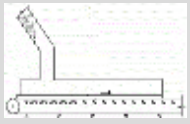
Se pueden encontrar diferentes tipos de aceros inoxidables. El material más común dentro de la industria alimenticia y más tolerante ante la corrosión es el acero tipo 304, por lo que puede ser implementado en el proyecto.

El mismo análisis se aplicó para seleccionar el material para la estructura del contador. Por ello, se ha optado por utilizar el acero tipo 304 por su resistencia tanto a daños físicos como a la corrosión. Además, esta tiene una superficie lisa que no generará daños en los alevines.

Selección del mecanismo de dosificación

Los mecanismos de dosificación son dispositivos encargados de expedir sustancias de diferentes naturalezas, transportándolas desde una entrada hasta una salida.

Tabla 23: *Tabla comparativa de los mecanismos de dosificación para el dispensador de alimento*

Características	Mecanismo de tornillo Sinfín	Mecanismo de compuerta rotativa	Mecanismo de banda transportadora
Representación gráfica			
Precisión de dispensado	Buena	Regular	Buena
Complejidad de construcción	Fácil construcción, su elemento principal es el tornillo.	Construcción moderada del mecanismo rotativo.	Construcción moderada debido a la banda transportadora y sus sistemas adicionales.
Mecanismos adicionales	• Motor • Sistema de engranajes	• Motor • Sistema de engranajes	• Motor • Banda rodante • Sistema de rodamiento • Compuerta reguladora de descarga
Distancia de transporte	Hasta 50 metros.	In situ.	Más de 50 metros.
Tipo de material que puede transportar	Materia polvoso y granulado.	Material polvoso y granulado.	Materiales sólidos y polvos.
Alteración del producto que trasporta	Si el material a transportar es frágil puede ser alterado.	El material no sufre alteraciones.	La cinta transportadora no provoca alteraciones a los materiales.

Fuente:[39, pp. 24-27]

Justificación

Mediante la tabla comparativa, se puedo definir que el mecanismo de tornillo sinfín es la adecuada para el sistema del dispensador de alimento porque su construcción es sencilla en comparación a otros sistemas. También su precisión es aceptable y al no tener mecanismos adicionales no requiere trabajos de mantenimiento constantes. Además, el tipo de material a transportar es del tipo granuloso, el mismo que no entra en la categoría de frágil por lo tanto no hay riesgo que sufra alteraciones durante su transporte.

Diseño de la tolva

Para determinar la capacidad que tendrá la tolva del dispensador, se tomó en cuenta la densidad de siembra de alevines que puede soportar el estanque.

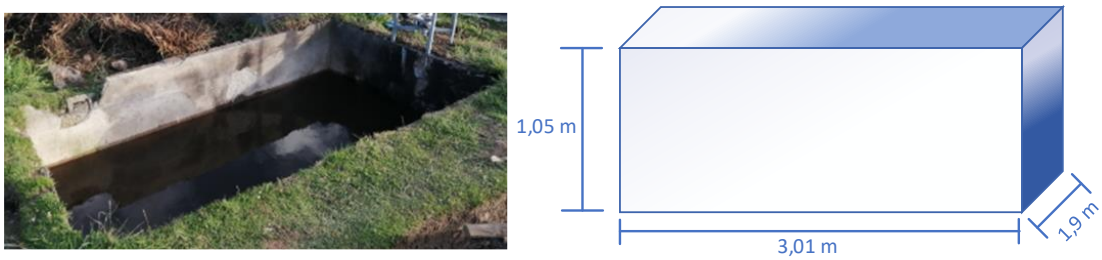


Figura 31: *De izquierda a derecha. Fotografía del estanque y sus dimensiones*

Fuente: Investigadores

Para el desarrollo del proyecto, se tiene a disposición un estanque rectangular de 1,05 metros de alto, 3,01 metros de largo y 1,9 metros de ancho. El estanque se llenó hasta los 80 centímetros para evitar la fuga de agua.

Por lo tanto, el volumen total de estanque será:

$$V_{estanque} = largo * ancho * alto$$

$$V_{estanque} = 3{,}01m * 1{,}9m * 0{,}80m = 4{,}58m^3$$

Tabla 24: *Densidad de siembra de alevines*

Longitud alevines	Número máximo por metro cúbico	
	Estanque Circular	Estanque Rectangular
3 cm	7500	--------
4 cm	4600	2300
5 cm	3400	1700

Fuente: [6, p. 13]

El tamaño de venta promedio de los alevines de trucha está entre los 3 y los 4 centímetros. Por lo tanto, se asumió un valor de 4 centímetros para el cálculo de la densidad de siembra.

$$Densidad_{siembra} = V_{estanque} * Número_{alevines}$$

$$Densidad_{siembra} = 4{,}58 * 2300 = 10534\ alevines$$

Para estimar el porcentaje de balanceado se ha recurrido a tablas de alimentación y se ha tomado el valor más alto para el cálculo, para mantener un margen de seguridad ya que, de lo contrario, si se selecciona un valor mínimo el alimento almacenado en la tolva se puede terminar antes de lo esperado. Por lo tanto, según la tabla alimenticia, para un tamaño promedio de 4 centímetros y con un valor máximo de 9.49 (porcentaje de alimentación a una temperatura 15 grados Celsius) y teniendo en cuenta que el peso promedio de un alevín de 4 centímetros es de 0,86 gramos. Además, se evitó el alimento acumulado mediante largos periodos de tiempo, se ha definido un periodo de almacenamiento de un mes, para que el llenado de la tolva sea cada 30 días y garantizando que el balanceado siempre este fresco.

Con los datos planteados anteriormente se calculó la capacidad de la tolva.

$$Biomasa_{alevines} = 10534 * 0{,}86 = 9059{,}24\ gramos$$

$$Alimento_{diario} = Biomasa_{alevines} * tasa\_alimentación_{\%}$$

$$Alimento_{diario} = 9059{,}24 * 9{,}49\% = 859{,}72\ gramos$$

Por lo tanto, para calcular el alimento que se necesita para un mes simplemente se multiplica por 30 días, y se tendrá un valor total de 25,79 Kg. Este valor encontrado, permitió calcular las dimensiones que debe tener la tolva para almacenar dicha capacidad.

Para el cálculo del volumen de la tolva hay que determinar primero la densidad aparente del balanceado. Para ello, hay que encontrar el volumen que ocupa una libra del alimento de peces.

$$Densidad_{aparente} = \frac{Masa}{Volumen}$$

$$Densidad_{aparente} = \frac{0{,}454\ kg}{675 cm^3} = 0{,}000673 \frac{kg}{cm^3}$$

Con la densidad que se calculó, se puede determinar el volumen que ocupará una masa de 25,79 kg.

$$Volumen = \frac{Masa}{Densidad_{aparente}}$$

$$Volumen = \frac{25{,}79\ kg}{0{,}000673 \frac{kg}{cm^3}} = 38320{,}95 cm^3$$

Por lo tanto, mediante prueba y error hay que encontrar las dimensiones que satisfaga dicho volumen. El diseño de la tolva está compuesto de dos figuras geométricas, un cubo y un prisma triangular.

$$Volumen_{tolva} = Volumen_{cubo} + Volumen_{prisma}$$

$$Volumen_{tolva} = (lado\ del\ cubo)^3 + (Area\ de\ la\ base\ del\ prisma) x (altura\ del\ prisma)$$

$$Volumen_{tolva} = (32\ cm)^3 + \left(\frac{32cm * 15cm}{2}\right) * 32cm$$

$$Volumen_{tolva} = 32768 cm^3 + 7680 cm^3$$

$$Volumen_{tolva} = 40448\ c^3$$

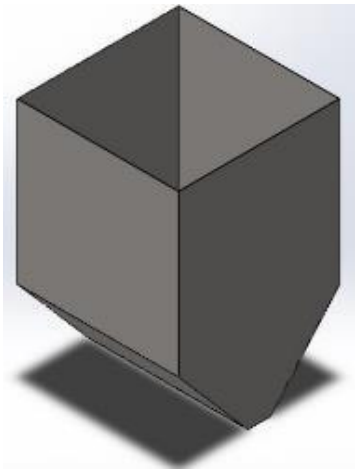


Figura 32: *Tolva del dispensador formado por un cubo y un prisma triangular*

Fuente: Investigadores

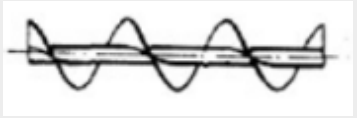
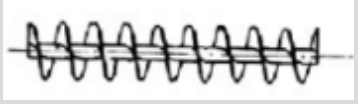
Diseño del sinfín

Como eje central del sinfín se ha utilizado una varilla de hierro dulce de 8 milímetros de diámetro y 320 milímetros de largo.

El tipo de material que se va a transportar es granulado y cada partícula mide entre 2 y 3 milímetros. Por lo tanto, para el cálculo del diámetro del tornillo se hizo un promedio entre los tamaños del material granulado que es 2.5 milímetros y se multiplico por 12.

$$Diámetro_{tornillo} = 2{,}5mm * 12 = 30mm$$

Tabla 25: *Tipo de hélices y sus aplicaciones*

Tipo de hélice	Aplicación	Representación
Hélice de igual paso que el diámetro del tornillo	Transporte de sólidos	
Hélice de gran paso, entre 1.5 a 2 veces el diámetro del tornillo	Transporte de productos que tienen buen flujo	
Hélice de paso corto. Mide 0,5 veces el diámetro del tornillo	Utilizado en aplicaciones con inclinación de hasta 20-25 grados y para el transporte de materiales con buen flujo.	

Fuente: [40]

Para el paso del tornillo, se hizo uso de la hélice de paso corto.

$$Paso_{hélice} = 0{,}5 * Diámetro_{tornillo}$$

$$Paso_{hélice} = 0{,}5 * 30mm = 15\ mm$$

Tabla 26: *Velocidad de giro según el tipo de material*

Tipo de material	RPM
Materiales pesados	Velocidad de giro (n) ≈ 50 rpm
Materiales ligeros	Velocidad de giro (n) < 150 rpm

Fuente: [40]

El área del relleno del canalón, se calculó mediante la siguiente fórmula:

$$S = \lambda \frac{\pi * D^2}{4}$$

Donde:

$S = área\ del\ relleno\ del\ transportador,\ en\ m^2$

$D = diámetro\ del\ canalón\ del\ transportador, en\ m$

$\lambda = coeficiente\ del\ material\ a\ transportar$

Para el tipo de carga ligera y poco abrasiva el valor de λ es 0,32

$$S = 0{,}32 \frac{\pi * 0{,}035^2}{4} = 3{,}08x10^{-4}m^2$$

La velocidad del desplazamiento del transportador está definida como

$$v = \frac{p * n}{60}$$

Donde:

$v = velocidad\ del\ transportador, en\ {}^{m}/_{s}$

$p = paso\ del\ tornillo, en\ m$

$n = velocidad\ de\ giro\ del\ tornillo, en\ rpm$

$$v = \frac{0{,}015 * 149}{60} = 0{,}037 \quad {}/_{s}$$

Para calcular la capacidad de transporte del sinfín se aplica la siguiente formula

$$Q = 3600 * S * v * \rho * i$$

Donde:

$Q = flujo\ del\ material, en\ {}^{t}/_{h}$

$S = área\ del\ relleno\ del\ transportador,\ en\ m^2$

$v = velocidad\ del\ transportador, en\ {}^{m}/_{s}$

$\rho = densidad\ material\ transportado, en\ {}^{t}/_{m^3}$

$i = coeficiente\ disminución\ de\ flujo\ por\ la\ inclinación\ del\ transportador$

Tabla 27: *Coeficientes de disminución de flujo del material*

Inclinación del canalón	0^o	5^o	10^o	15^o	20^o
i	1	0,9	0,8	0,7	0,6

Fuente: [40]

$$Q = 3600 * 3{,}08x10^{-4} m^2 * 0{,}037\ {}^{m}/_{s} * 0{,}673\ {}^{t}/_{m^3} * 0{,}6$$

$$Q = 0{,}017\ {}^{t}/_{h}$$

Para encontrar la potencia del motor necesaria se ha aplicado la siguiente fórmula.

$$P = P_H + P_N + P_i = C_o * \frac{Q * L}{367} + \frac{D * L}{20} + \frac{Q * H}{367} = \frac{Q * (C_o * L + H)}{367} + \frac{D * L}{20}$$

Donde:

$P = Potencia\ de\ accionamiento\ del\ transportador, en\ kW$

$P_H = Potencia\ para\ el\ desplazamiento\ horizontal\ del\ material, en\ kW$

$P_N = Potencia\ necesaria\ para\ el\ movimiento\ del\ sinfín\ en\ vacío, en\ kW$

$P_i = Potencia\ necesaria\ para\ mover\ el\ sinfín\ inclinado, en\ kW$

$C_o = Coeficiente\ de\ resistencia\ del\ material$

$Q = Flujo\ del\ material, en\ {}^{t}/_{h}$

$L = Longitud\ del\ sinfín, en\ m$

$D = Diámetro\ del\ canalón\ del\ transportador, en\ m$

$H = Altura\ del\ sinfín, con\ respecto\ a\ la\ horizontal, en\ m$

Tabla 28: *Valor de C_o según el material*

Tipo de material	Valor de C_o
Productos granulosos	1,2
Sosa	1,6
Sal de roca	2,5
Arena, cemento	4

Fuente: [40]

$$P = \frac{Q * (C_o * L + H)}{367} + \frac{D * L}{20}$$

$$P = \frac{0{,}017 * (1{,}2 * 0{,}32 + 0{,}03)}{367} + \frac{0{,}035 * 0{,}32}{20}$$

$$P = 1{,}92x10^{-5} + 5{,}6x10^{-4} = 5{,}79x10^{-4} kW$$

$$P = 7{,}76x10^{-4} HP$$

Para el cálculo del torque necesario para el motor se ha aplicado la siguiente fórmula

$$Torque_{(Nm)} = \frac{7127 * P_{HP}}{N_{(rpm)}}$$

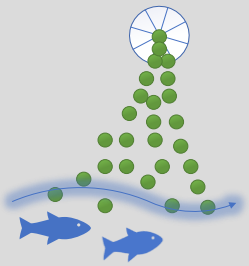
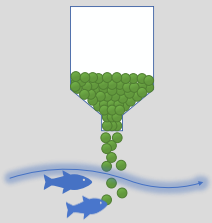
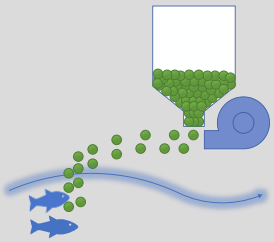
$$Torque_{(Nm)} = \frac{7127 * 7{,}76 * 10^{-4}}{149}$$

$$Torque_{(Nm)} = 0{,}037\ Nm$$

Selección del mecanismo de dispersión de alimento

El mecanismo de dispersión permite distribuir una determinada cantidad de material en un área en específico.

Tabla 29: *Tabla comparativa de los sistemas de dispersión*

Características	Dispersión por disco centrífugo	Distribución por gravedad	Dispersión por corriente de aire
Representación gráfica			
Capacidad de dispersión	Puede dispersar a largas distancias y cubrir extensas áreas según el diseño del disco y su ubicación.	Dispersión limitada debido que el material cae en el mismo sitio.	Puede dispersar a largas distancias, pero el área que cubre es limitada.
Tipo de material que puede dispersar	Preferentemente granulado.	Material de cualquier tipo.	Gránula o micro granulado.
Distribución	Uniforme	Uniforme	Muy Uniforme
Capacidad para regular el ancho de dispersión	Si, mediante la posición de caída del material sobre el disco y el ángulo de inclinación se puede regular el área de dispersión.	No, el área de dispersión no es modificable.	No, el área no es modificable.
Directividad	Se puede colocar un deflector para que disperse en una sola dirección.	Única dirección de dispersión vertical.	Debido al flujo de aire puede dispersar en una sola dirección.
Complejidad de construcción	Moderada, requiere definir los parámetros de construcción del disco.	Sencilla, no requiere ningún sistema adicional para dispersar el alimento.	Moderada, requiere la regulación de un sistema que genere un flujo de aire.

Fuente: [41]

Justificación

Mediante la tabla comparativa se determinó que el sistema de dispersión adecuado es la dispersión a través de un disco centrífugo, debido que tiene mayor distancia de alcance y se puede regular el ancho de dispersión, en comparación con los demás sistemas que no cuenta la característica para regular su anchura de distribución del material. Otra característica relevante es la dirección de dispersión, todos los sistemas

tienen directividad ajustable, a excepción del distribuidor por gravedad, para el centrífugo solo hay que añadir un deflector para agregar dicha característica.

Diseño de la centrífuga

Para el diseño del sistema centrífugo del dispensador, se ha basado en el principio de las abonadoras de disco centrífugo.

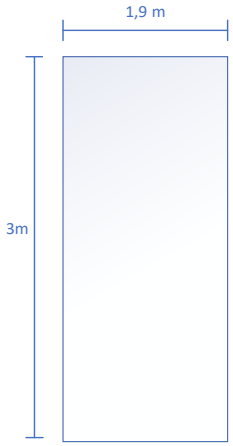


Figura 33: *Dimensiones del estanque de alevines*

Fuente: Investigadores

Para ello, hay que definir las características que tendrá el disco como el largo de las aspas, el ángulo que deben tener con respecto al radio del disco, el diámetro, etc. Estos parámetros varían en función de las dimensiones del estanque.

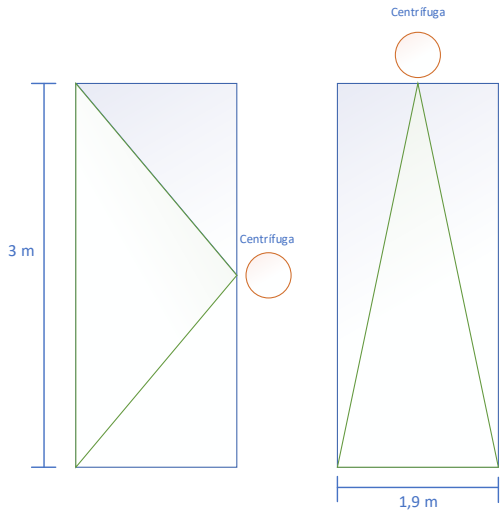


Figura 34: *De izquierda a derecha. Ubicación Lateral y superior de la centrífuga*

Fuente: Investigadores

Para la ubicación de la centrífuga se ha calculado el área que abarca en cada posición.

Para la ubicación lateral, se tiene un área de:

$$\text{Área}_{lateral} = \frac{bxh}{2} = \frac{3 * 1{,}9}{2} = 2{,}85m^2$$

Para la ubicación superior, se tiene un área de:

$$\text{Área}_{superior} = \frac{bxh}{2} = \frac{1{,}9 * 3}{2} = 2{,}85m^2$$

Tanto la ubicación lateral como superior tienen una misma área de dispersión. Pero, gráficamente se puede apreciar que la ubicación en la posición superior tiene espacios laterales más estrechos con respecto a la ubicación lateral.

Por lo tanto, se ha determinado que la mejor ubicación del dispensador es en la parte superior del estanque.

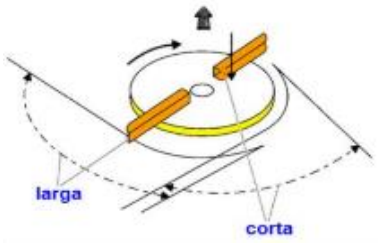


Figura 35: *Longitud de las aspas del disco*

Fuente: [41]

La anchura de proyección está en función de la longitud de las aspas. Para una proyección ancha se utilizan aspas largas, mientras que para una proyección más corta las aspas deben ser cortas.

En este caso, se necesita una proyección más corta, por lo tanto, para el disco se usarán aspas cortas.

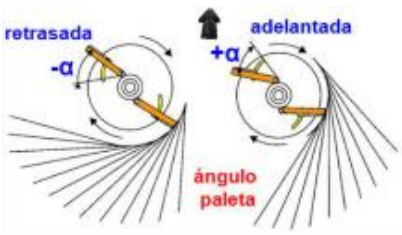


Figura 36: *Ángulo formado entre las aspas y el radio del disco*

Fuente: [41]

Otro aspecto que determina el ancho de dispersión, es el ángulo que forma el aspa con el radio del disco. Cuando el aspa se encuentre retrasado con respecto al radio del disco se tendrá una mayor proyección. Por el contrario, si el aspa está adelantada, la proyección será más corta. Para entender el término atrasado y adelantado con respecto al ángulo de las aspas, hay que imaginar un reloj analógico, el horero a las 12:00 representa el punto de referencia del radio, mientras tanto el minutero representa las aspas y el ángulo formado entre horero y minutero es el ángulo α. Entonces, cuando el minutero esté en 11 significa que el aspa esta adelantada, por el contrario, si está en 1, el aspa está retrasada.

Por lo tanto, para el disco del dispensador se colocarán sus aspas adelantas un ángulo $\alpha = 30°$ con respecto al radio.

Para determinar el diámetro del disco, se hizo uso del movimiento circular uniforme y del tiro semi parabólico. Para facilitar los cálculos, se aplicó el análisis a un grano del alimento del alevín.

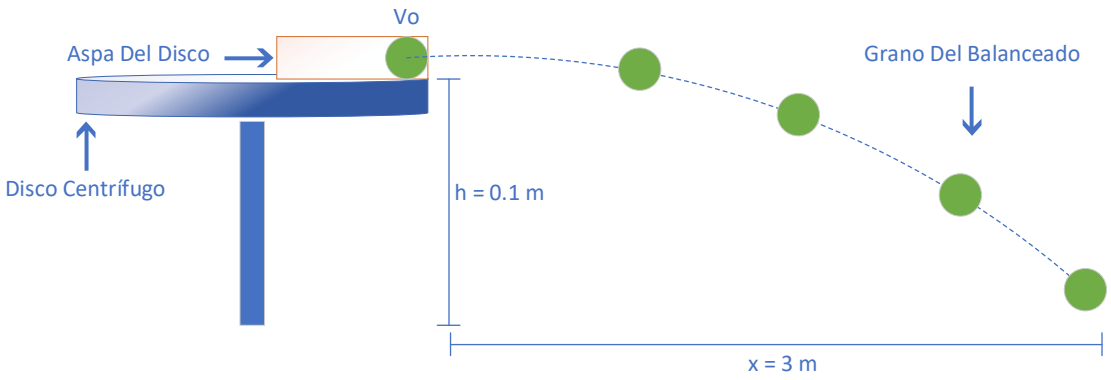


Figura 37: *Lanzamiento semi parabólico de un grano*

Fuente: Investigadores

$$t = \sqrt{\frac{2h}{g}} = \sqrt{\frac{2 * 0{,}1}{9{,}8}} = 0{,}14\ segundos$$

$$V_o = \frac{x}{t} = \frac{3}{0{,}14} = 21{,}43\ {}^{m}/_{s}$$

Por lo tanto, para ser lanzado hasta una distancia de 3 metros en condiciones ideales se necesita una velocidad lineal inicial de al menos 21,43 $/_s$

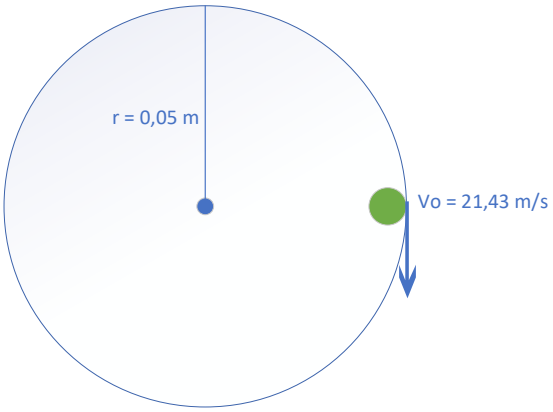


Figura 38: *Disco de dispersión*

Fuente: Investigadores

Para el disco de dispersión, se ha propuesto un tamaño de 5 centímetros de radio.

$$\omega = \frac{v_{tangencial}}{radio} = \frac{21{,}43}{0{,}05} = 428{,}6\ {}^{rad}/_{s}$$

$$Velocida\ de\ giro\ disco = 4092{,}83\ R.P.M$$

Por lo tanto, en condiciones ideales para que el disco de dispersión con un radio de 5 centímetros lance una partícula hasta una distancia de 3 metros, se necesita al menos 4092,83 R.P.M (revoluciones por minuto)

Diseño del recipiente de carga

Para el recipiente de carga se ha colocado un envase plástico con la finalidad de no agregar pesos adicionales sobre la celda de carga.

Como se había definido en la sección del diseño de la tolva. La densidad del alimento es $0{,}673\frac{g}{cm^3}$ y la cantidad máxima de balanceado diario es $859{,}72\ gramos$, esta cantidad de divide entre la frecuencia de alimentación, que para los alevines es 10 veces al día máximo. Entonces durante cada dosis hay que dispersar 85,97 gramos. Por lo tanto, el recipiente debe ser capaz de albergar al menos, 85,97 gramos.

El envase que se ha utilizado tiene la capacidad de almacenar un volumen de 600 cm^3.

$$Capacidad_{recipiente} = densidad_{alimento} * volumen_{envase}$$

$$Capacidad_{recipiente} = 0{,}673\frac{g}{cm^3} * 600cm^3 = 403{,}8\ g$$

Por lo tanto, el envase es capaz de almacenar hasta 403,8 gramos, capacidad suficiente para poder guardar cada dosis de alimento.

Diseño de la compuerta de descarga

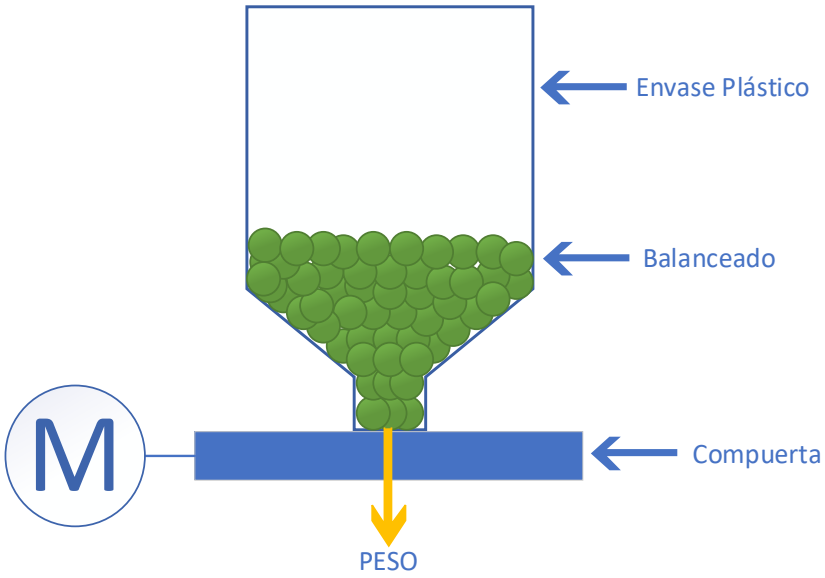


Figura 39: *Representación gráfica de la compuerta de descarga*

Fuente: Investigadores

Para el análisis de la compuerta de descarga, se puede apreciar gráficamente que la única fuerza que se ejerce sobre la compuerta es el peso del alimento.

La compuerta está acoplada hacia un actuador, cuyo eje de rotación es únicamente en el eje horizontal. Además, se puede apreciar que ninguna fuerza perpendicular al eje de rotación se opone al movimiento de la compuerta. Por lo tanto, no se puede hacer un cálculo del torque necesario para el actuador. Además, se toma en cuenta que la compuerta debe soportar solo una carga máxima de 85,97 gramos durante cada dosis que no genera un peso excesivo que requiera un análisis más avanzado. Por ello, la implementación de un actuador comercial con un buen torque será suficiente.

Selección de actuador para la compuerta

El actuador es el encargado de abrir la compuerta una vez que se haya llegado hasta el peso requerido.

Tabla 30: *Tabla comparativa de los actuadores*

Características	SG90	TOWER PRO MG995	MG946R
Representación gráfica			
Voltaje de operación	4,8 – 6 VDC	4,8 – 6,6 VDC	4,8 – 7,2 VDC
Torque	1,2 kg/cm	9,4 kg/cm	10,5 Kg/cm
Ángulo de rotación	$0^{\circ} hasta\ 180^{\circ}$	360° rotación continua	$0^{\circ} hasta\ 180^{\circ}$
Sistema de engranajes	Plástico	Metal	Metal
Temperatura de operación	$-30℃\ a + 60℃$	$0℃\ a + 55℃$	$0℃\ a + 55℃$
Corriente (sin carga)	10 mA	170 mA	170 mA

Fuente: Investigadores

Justificación

Se ha seleccionado el servomotor MG946R porque tiene un mejor torque en comparación a los demás componentes. Además, su sistema de engranajes de metal permite realizar trabajos más pesados sin riesgo de dañar el sistema. Por otra parte, el TOWER PRO MG995 al no tener un sistema de control de bucle cerrado debido a su rotación continua de 360 grados no se puede colocar en una posición angular en específico por lo que queda descartado a pesar de tener un torque casi cercano al MG946R.

Selección del actuador para el tornillo sinfín

Actuador encargado de hacer girar el tornillo de sinfín:

Tabla 31: *Tabla comparativa de los actuadores para el sinfín*

Características	NEMA 23	NEMA 17	NEMA 8
Representación gráfica			
Voltaje de operación	12 – 24 VDC	12 VDC	4,8 VDC
Torque	0,5 – 3 Nm	0,2 – 1 Nm	0,01 – 0,04 Nm
Temperatura de operación	$-10℃\ a + 50℃$	$-10℃\ a + 40℃$	$-10℃\ a + 50℃$
Diámetro del eje	8 mm	5 mm	4 mm
Corriente/fase	2800 mA	1800 mA	200 mA
Número de fases	2	4	2

Fuente: Investigadores

Justificación

Según el torque calculado para el motor del tornillo sinfín solo sería necesario un torque de 0,037 Nm, el motor a pasos NEMA 8 cumple con los requisitos, pero, el diámetro del sinfín era de 8mm por lo que no se podrá acoplar correctamente. NEMA 23 a parte de tener un diámetro inferior al requerido va tener un alto consumo debido a sus cuatro fases. El NEMA 23 cumple con los requisitos del diámetro y del torque adecuado, por lo que se implementó en el sistema del dispensador.

Selección del actuador para el disco centrifugo

Este actuador generará las revoluciones necesarias para lanzar el alimento hasta una determinada distancia. En el mercado se ha encontrado los siguientes modelos.

Tabla 32: *Tabla comparativa de los actuadores para el disco centrífugo*

Características	Motor DC 775	Motor DC con caja reductora	Motor DC 540
Representación gráfica			
Voltaje de operación	6 -38 VDC	6 -12 – 24 VDC	12 – 24 VDC
Torque	0,029 – 0,127 Nm	0,049 – 2,941 Nm	0,294 – 0,392 Nm
Temperatura de operación	$-10℃\ a + 60℃$	$-20℃\ a + 50℃$	$-10℃\ a + 60℃$
Corriente de consumo	450 mA – 3000 mA	500 mA – 2500 mA	300 mA – 600 mA
R.P.M	4000 – 28000 RPM	1 – 500 RPM	1 – 928 RPM

Fuente: Investigadores

Justificación

Durante los cálculos se determinó que se necesita al menos 4092,83 R.P.M (revoluciones por minuto) para lanzar el alimento hasta 3 metros. Todos los motores cumplen con el rango de voltaje. Además, el disco centrifugo al estar construido en un material ligero, el motor no tendrá dificultad para moverse con él. Por lo tanto, el motor adecuado para el dispensador es el modelo 775, porque tiene entre 4000 y 28000 rpm.

Selección de la compuerta de descarga

La compuerta de descarga es la encarga de liberar el material que almacenó durante el proceso de transporte del material.

Tabla 33: *Tabla comparativa de válvulas de descarga*

Características	Válvula Rotativa	Compuerta giratoria	Válvula mariposa
Representación gráfica			
Tipo de material que soporta	Materiales polvosos y granulosos.	Solo materiales granulosos.	Solo fluidos.
Tipo de aplicación	Nivel industrial.	Aplicaciones domésticas.	Nivel industrial.
Costo	Elevado	Económico	Moderado
Método de adquisición	Bajo pedido con el proveedor.	Disponible en cualquier tienda electrónica.	Bajo pedido con el proveedor.
Peso	50000 gramos	2 gramos	1400 gramos
Dimensiones	Largo 67,2 cm Alto 40 cm Ancho 42 cm	Largo 2 cm Alto 2,5 cm Ancho 0,5 cm	Largo 10 cm Alto 15 cm Ancho 3cm

Fuente: Investigadores

Justificación

Por obvias razones, la única compuerta que se puede implementar en el dispensador es una compuerta giratoria en el sentido horizontal construido con un servomotor comercial. La compuerta de mariposa y similares como la electroválvula quedan descartadas debido que están diseñadas solo para materiales fluidos y están enfocados a nivel industrial, por lo tanto, no son compatibles para proyectos de pequeñas dimensiones. Por otra parte, la compuerta rotativa tampoco es compatible debido que está enfocado para niveles industriales y sus dimensiones y peso son exageradas para el proyecto. Los tipos de válvulas presentadas a excepción de la compuerta giratoria, solo se las puede encontrar en tamaños industriales para tareas pequeñas no hay válvulas similares en el mercado, por lo tanto, la selección de componentes queda muy limitada.

Selección de sensor de peso

El sensor de peso, es el encargado de pesar la cantidad adecuada para cada dosis. De manera comercial se tienen varios sensores que funcionan mejor o peor según el área de aplicación.

Tabla 34: *Tabla comparativa de los sensores de carga*

Características	Celda de carga de flexión	Celda de carga de compresión	Sensor de fuerza resistivo FSR 402
Representación gráfica			
Empotrable	Si, cuenta con orificios a cada extremo para ser fijados mediante tornillos.	No cuenta con puntos de sujeción.	No cuenta con puntos de sujeción.
Carga máxima	Hasta 5 Kg	Hasta 50 Kg	Hasta 10 Kg
No linealidad	+/- 0,05 %	+/- 0,03 %	Sensor no lineal
No repetividad	+/- 0,03 %	+/- 0,03 %	+/- 2 %
Histéresis	+/- 0,05 %	+/- 0,03 %	+ 10%
Módulos adicionales	Amplificador HX711	Amplificador HX711	El fabricante recomienda configuraciones mediante amplificadores operacionales.
Dimensiones	1,27 x 1,27 x 7,5 cm	3,4 x 3,4 cm	Diámetro del círculo 1,3 cm Largo 5,6 cm
Temperatura de operación	$-25℃\ a\ +85℃$	$0℃\ a\ +50℃$	$-30℃\ a\ +70℃$
Tensión de operación	5 - 10 VDC	5 - 10 VDC	En función del amplificador operacional.

Fuente: Investigadores

Justificación

El sensor que brinda mejor estabilidad es el sensor de carga de flexión debido que tiene puntos de sujeción que permite mantener al sensor fijo en la estructura del dispensador por un lado y por el otro al recipiente encargado de almacenar el alimento que ha

transportado el sinfín. En cuanto a la precisión se la puede analizar mediante los parámetros de no linealidad que muestra el valor máximo de desviación con respecto a la curva característica del sensor. Para el sensor de flexión apenas hay una tolerancia de diferencia del +/- 0,02 % con respecto al sensor de compresión, por lo tanto, no influirá en el resultado final.

Selección de sensor de distancia

Sensor encargado de determinar la distancia a la que se encuentra determinado objeto u obstáculo. En el mercado se tienen algunos modelos de sensores disponibles.

Tabla 35: *Tabla comparativa de los sensores de distancia*

Características	HC-SR04	SHARP GP2Y0A21YK0F	SHARP GP2Y0D805Z0F
Representación gráfica			
Tipo de tecnología	Ultrasónico	Óptico	Óptico
Distancia de medición	2 cm hasta 400 cm	10 cm hasta 80 cm	0,5 cm hasta 5 cm
Ángulo de detección	15°	35°	20°
Velocidad de respuesta	Media	Alta	Alta
Es afectado por el polvo	Inmune	Afectado	Afectado
Influencia del material y color en la detección	Inmune	Afectado	Afectado
Señal de salida	Señal digital	Señal analógica	Señal digital
Voltaje de operación	5 VDC	4,5 – 5 VDC	2,7 – 6,2 VDC
Temperatura de funcionamiento	$-20℃\ a + 70℃$	$-10℃\ a + 60℃$	$-10℃\ a + 60℃$
Corriente de consumo	15 mA	30 mA	5 mA

Fuente: Investigadores

Justificación

El sensor que mejor se adapta al proyecto es el HC-SR04 debido a que tiene un rango de detección más amplio desde los 2 centímetros hasta 400 centímetros, en contraste con el sensor GP2Y0A21YK0F que su rango empieza desde los 10 cm, en cambio el sensor GP2Y0D805Z0F a parte de tener un corto alcance, su señal de salida digital solo permite dos estados 1 (se ha detectado un objeto) y 0 (ningún objeto detectado) por lo que queda totalmente descartado. Además, los sensores ópticos son afectados por el polvo, el tipo de material y su color provocando que la señal de rebote no retorne adecuadamente hasta el sensor.

Selección de tarjeta de desarrollo dispensador

Esta tarjeta electrónica tiene como finalidad ser un sistema de control, para leer los datos de los sensores y desde la nube y en función a ello, ejecutar determinadas tareas.

Tabla 36: *Tabla comparativa de las tarjetas de desarrollo*

Características	NODEMCU V2	ESP-WROOM-32	WEMOS D1 MINI
Representación gráfica			
Procesador	Xtensa Single Core 32-bit L106	Xtensa Dual Core 32-bit LX6	Tensilica Xtensa LX3 (32 bit)
Frecuencia de reloj	80 MHz	160MHz	80MHz/160MHz
Voltaje de operación	3,3 VDC	3,3 VDC	5 VDC
Memoria flash	4 MB	16 MB	4 MB
E/S Digitales	11	18	11
E/S Analógicas	1	18	1
Temperatura de operación	$-40℃\ a + 125℃$	$-40℃\ a + 125℃$	$-40℃\ a + 125℃$
Conectividad	802.11 b/g/n	802.11 b/g/n y Bluetooth	802.11 b/g/n
RTC	NO	SI	NO
Pines PWM	8	16	9
Consumo	70 mA	240 mA	70 mA

Fuente: Investigadores

Justificación

Como se puede observar en el cuadro comparativo todas tienen casi las mismas características en cuanto a la conectividad, procesador, memorias, etc. Para el uso en el proyecto se podría usar cualquiera, pero, el dispensador requiere al menos 11 pines. El modelo Nodemcu podría ser usado perfectamente, pero hay que tener en cuenta que no todos los pines son utilizables, en realidad de los 11 pines solo 9 (D0-D8) se los puede usar sin tener comportamientos inesperados. Los 2 pines restantes corresponden a TX y RX, que pueden ser usados únicamente como entrada y salida respectivamente, pero durante cada reinicio su comportamiento será inestable. Por lo tanto, el modelo que mejor se ajusta al proyecto es la ESP-WROOM-32 que posee 18 pines utilizables teóricamente, pero en realidad son 16 que se pueden usar de manera estable, cantidad suficiente para desarrollar el proyecto. Además, se puede rescatar la implementación de un módulo RTC interno que puede ser utilizado para almacenar el tiempo.

Implementación de la placa del sistema

Para la placa electrónica se ha optado por un diseño mono bloque con la finalidad de facilitar la conexión entre módulos mediante las pistas de cobre.

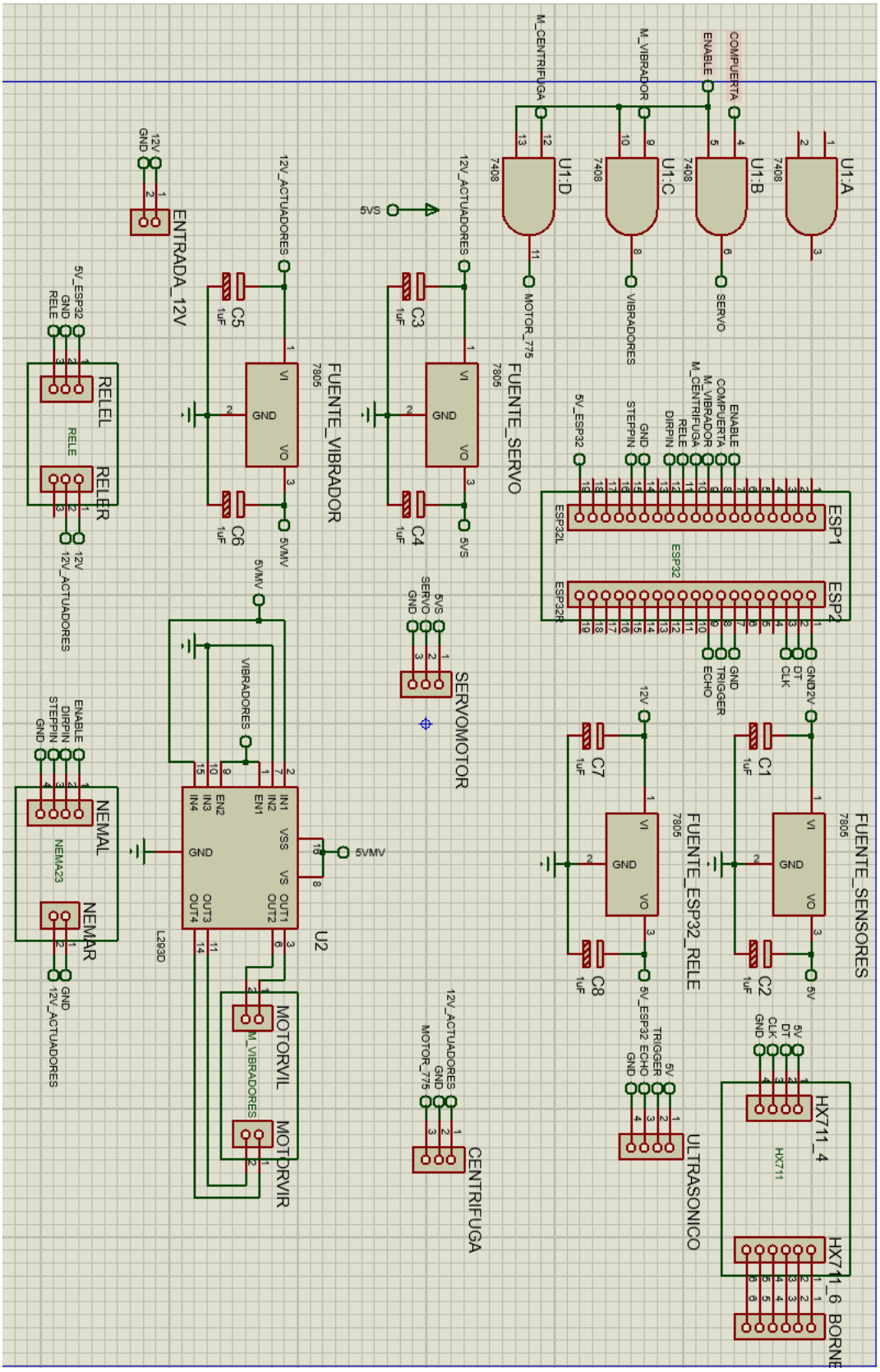


Figura 40: *Esquema completo del sistema del dispensador*

Fuente: Investigadores

Antes de empezar con el diseño del PCB, el circuito debe ser probado en un protoboard para verificar si su comportamiento responde a las necesidades planteadas. Caso contrario, realizar las correcciones pertinentes.

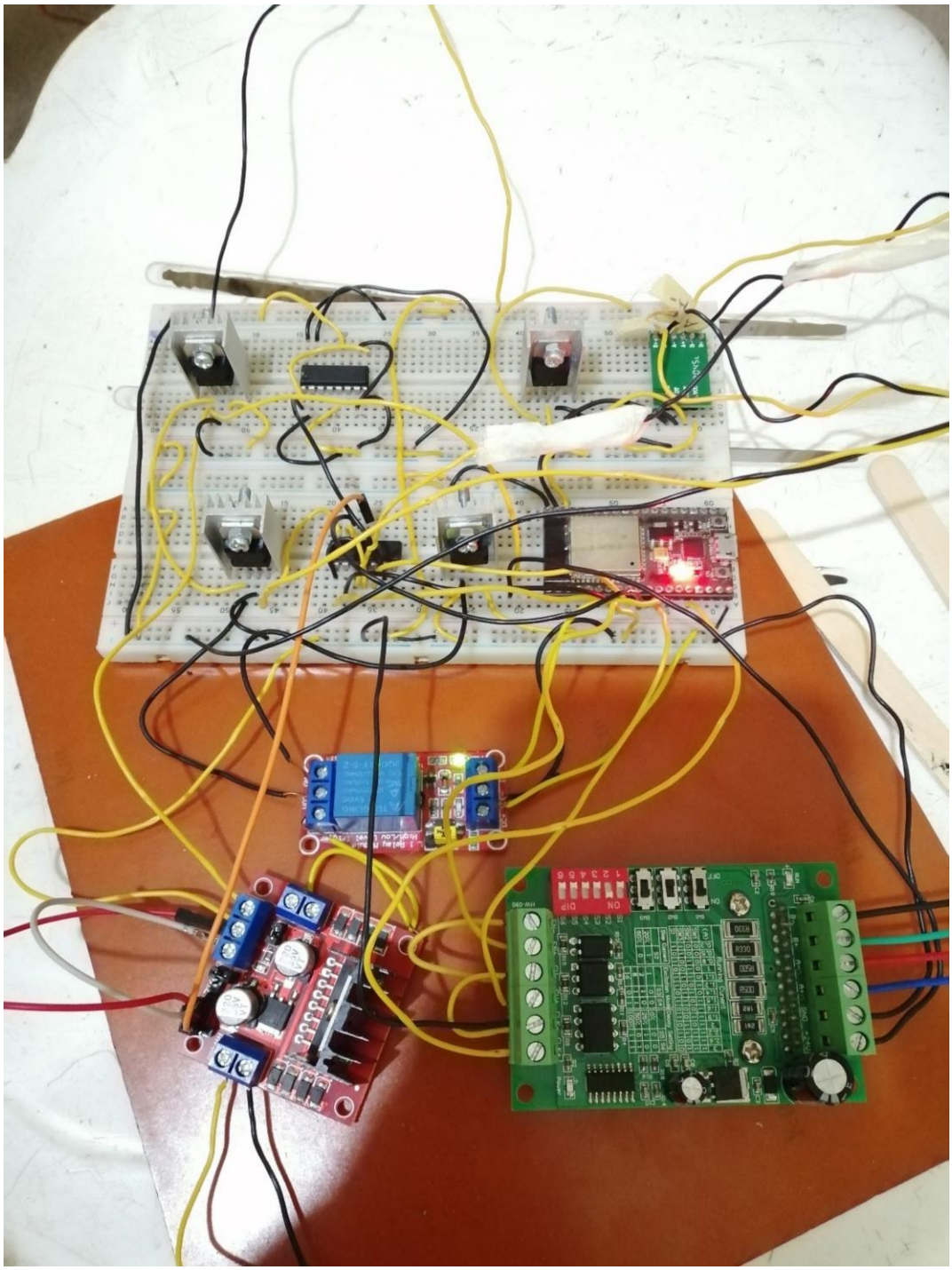


Figura 41: *Circuito en modo de prueba*

Fuente: Investigadores

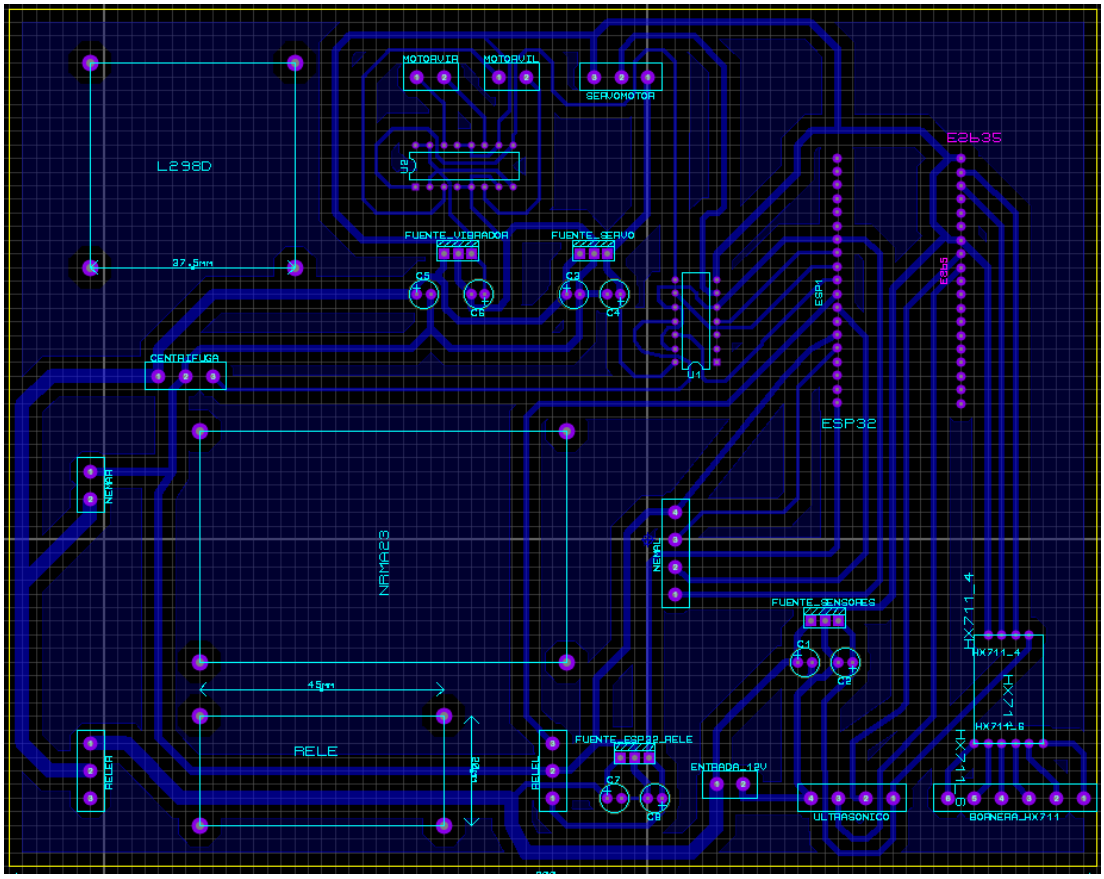


Figura 42: *Diseño PCB del sistema del dispensador en Proteus*

Fuente: Investigadores

Se han colocado los componentes y módulos en regiones estratégicas con la finalidad de crear rutas directas para la conexión de los actuadores. Además, se ha tomado en cuenta que las pistas de los componentes no entren en conflicto entre ellas para evitar el diseño a doble cara.

Pasos para el diseño de la placa electrónica

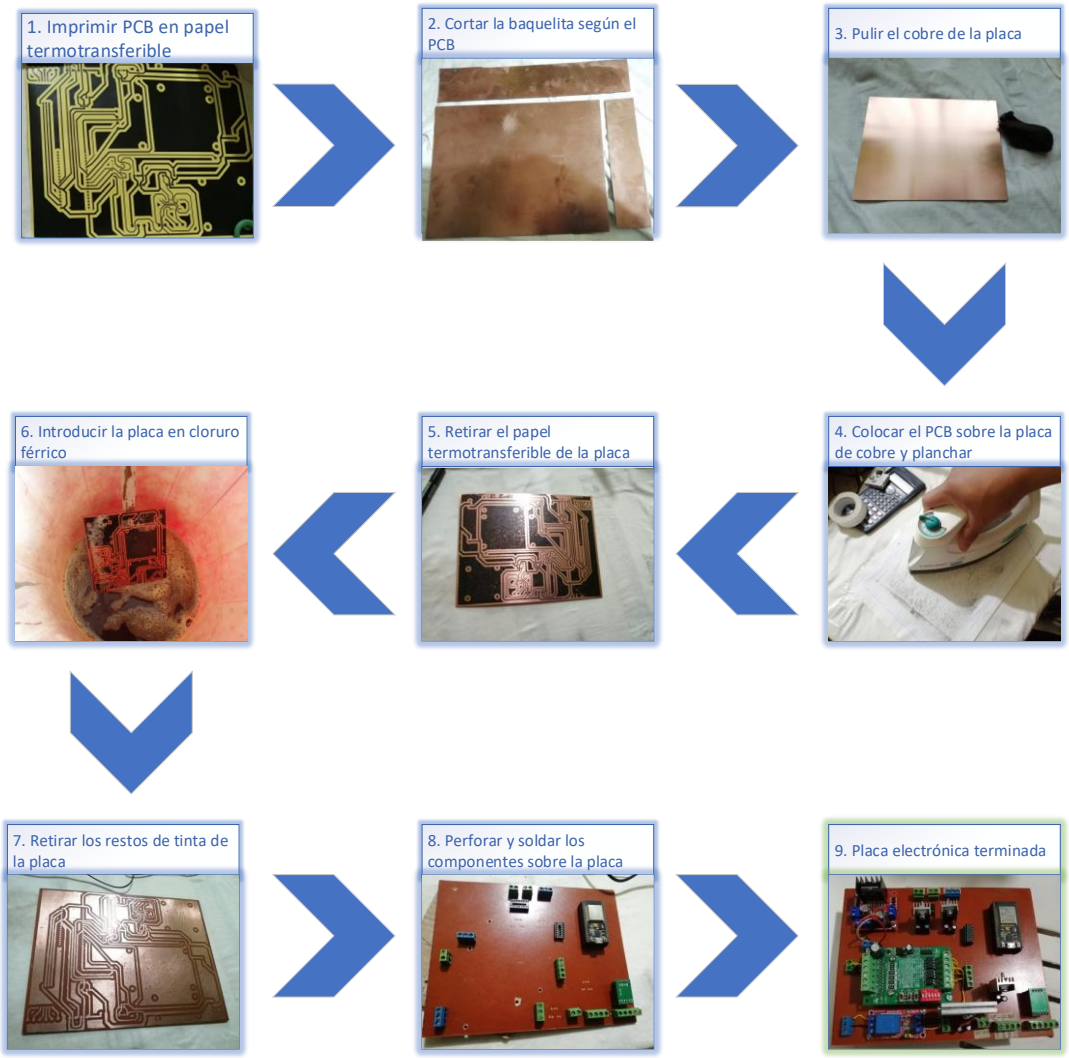


Figura 43: *Etapa para el diseño de la placa electrónica*

Fuente: Investigadores

Como se puede apreciar en la figura superior, como primer paso para desarrollar la placa, el PCB (previamente desarrollado en Proteus) debe ser impreso en papel termotransferible para que se pueda traspasar fácilmente al aplicar calor. En segundo lugar, la baquelita ha sido redimensionada para eliminar los excesos. Esta nueva placa es pulida con el fin de eliminar impurezas de la superficie y la tinta se adhiera con mayor facilidad. Tanto la placa como el papel impreso se deben colocar cara a cara para empezar con la transferencia de calor a través de plancha, este proceso durará entre 5 a 7 minutos. Para facilitar el retirado del papel sobre la placa hay que humedecerlo con agua fría durante 3 minutos. En caso de que se hayan perdido pistas, se las puede recuperar mediante el uso de un marcador permanente punta fina. Esta

placa se la introduce en cloruro férrico y se la remueve constantemente hasta que el cobre que no está cubierto por la tinta se desprenda completamente. Hasta este punto se ha llegado hasta el séptimo paso, donde hay que retirar los restos de tinta con la finalidad de comprobar la continuidad de las pistas mediante un multímetro para determinar si hay conexión o cortos entre las pistas. Como penúltimo paso viene el perforado de la placa y el montaje de los componentes para aplicar soldadura mediante cautín y estaño. Finalmente, se obtiene la placa con todos los módulos conectados para verificar su funcionamiento.

El dispensador trabaja con sistemas de 5V y 12V. Como fuente de alimentación general se está utilizando una batería de 12V, por lo tanto, para alimentar los circuitos de 5V se ha implementado reguladores de voltaje. A continuación, se muestra una tabla de los componentes y sus respectivos voltajes.

Tabla 37: *Componentes*

Componente	Voltaje de alimentación
ESP32	5V
RELÉ	5V
Sensor HX711	5V
Sensor HC-SR04	5V
Servomotor MG 946R	5V
Driver L293D	5V
Driver L298N	12V
Driver TB6560	12V

Fuente: Investigadores

En el circuito se están utilizando reguladores lineales 7805, por lo tanto, se tendrá pérdida de potencia. Para el análisis, se puede representar al regulador 7805 como una resistencia en serie con las cargas que alimenta. El voltaje que cae en el regulador se puede encontrar de la siguiente manera. $V_{regulador} = (Voltaje_{entrada} - 5v)$. Como el sistema está siendo alimentado con una batería de 12v, habrá una caída de 7v en el regulador. Por lo tanto, la potencia del circuito será tomada desde la entrada del regulador para los cálculos de la batería del sistema.

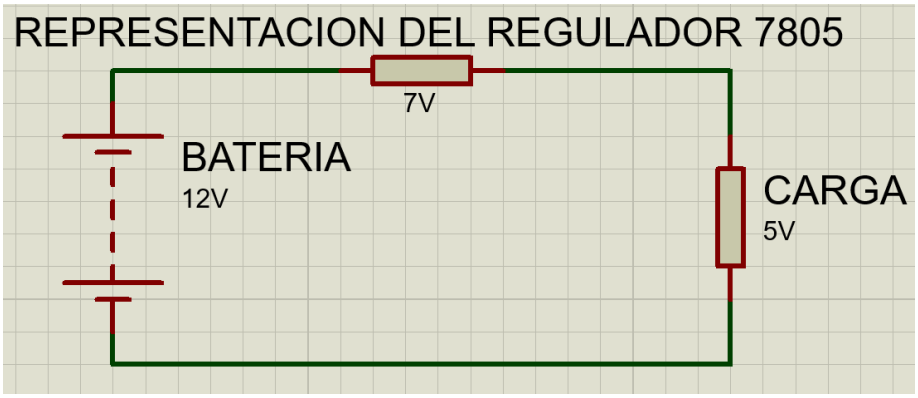


Figura 44: *Representación del regulador lineal en el circuito*

Fuente: Investigadores

El sistema completo del dispensador consta de 4 circuitos con sus respectivos reguladores lineales y 2 circuitos que funcionan directamente con la fuente de 12V.

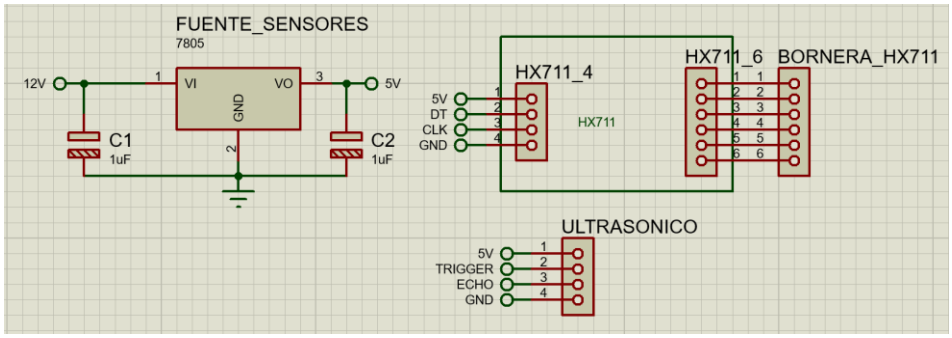


Figura 45: *Circuito de los sensores (7805 + Sensor HX711 + Sensor HC-SR04)*

Fuente: Investigadores

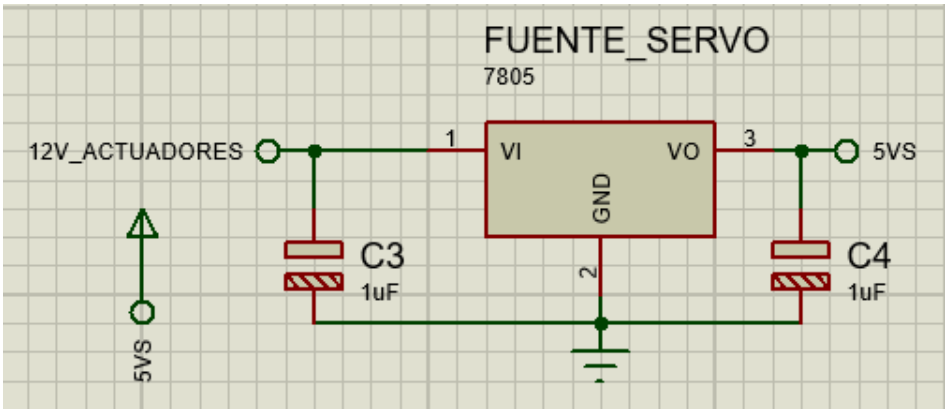


Figura 46: *Circuito para el servomotor (7805 + MG 946R)*

Fuente: Investigadores

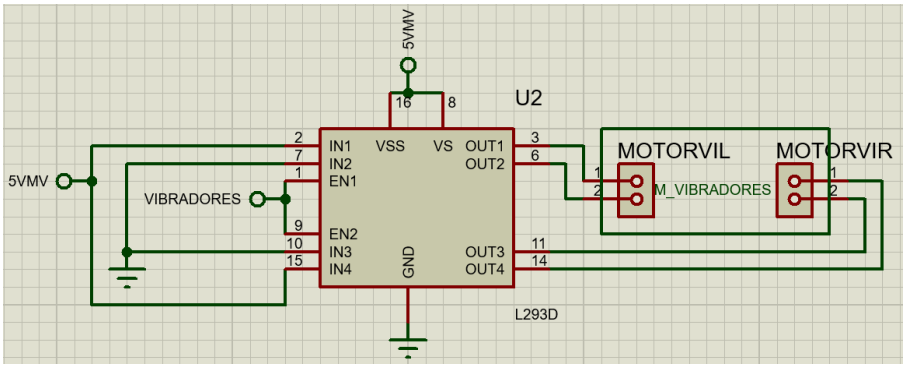


Figura 47: *Circuito para los motores vibradores (7805 + L293D)*

Fuente: Investigadores

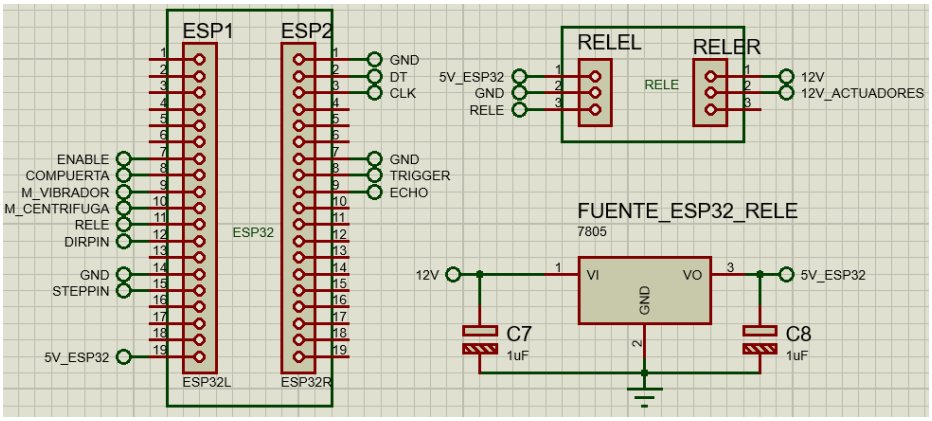


Figura 48: *Circuito de control y potencia (7805 + ESP32 + RELÉ)*

Fuente: Investigadores

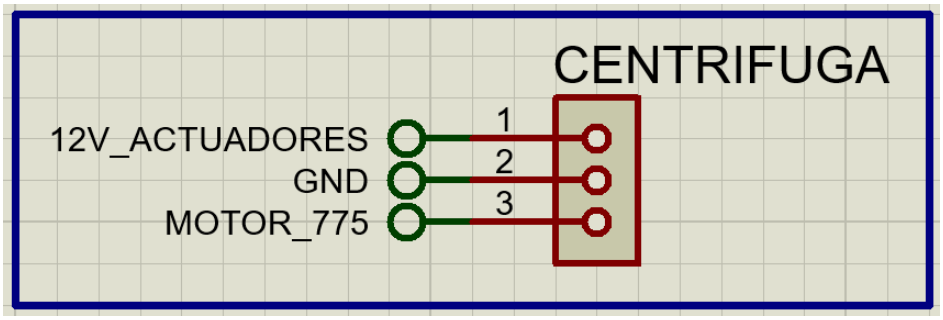


Figura 49: *Circuito para el driver de la centrifuga (L298N*

Fuente: Investigadores

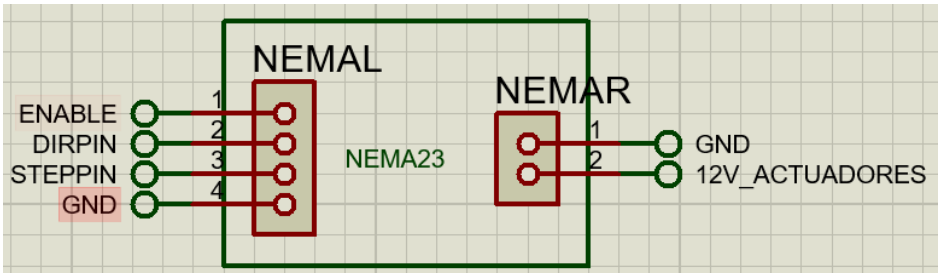


Figura 50: *Circuito para el driver del sinfín (TB6560)*

Fuente: Investigadores

Para el análisis de las corrientes de consumo y potencias del sistema, se tomará en cuenta 3 estados de trabajo. El modo en reposo, en funcionamiento y arranque. Además, se registrará los tiempos de cada actividad.

Tabla 38: *Consumo de corrientes*

Cargas	Corrientes (Amperios)		
	REPOSO	ARRANQUE	FUNCIONANDO
REGULADOR+HX711+HC-SR04	NA	NA	0,02
REGULADOR+L293D	0,03	NA	0,14
REGULADOR+SERVO946R	0,02	NA	0,04
REGULADOR+ESP32+RELE	0,18	NA	0,35
REGULADOR+ESP32 (DEEP SLEEP) +RELE	NA	NA	0,04
Driver Centrifuga L298N	0,04	2	1,20
Driver sinfín TB6560	0,12	NA	0,20

Fuente: Investigadores

Tabla 39: *Consumo de potencias*

Cargas	Potencias (Watt)		
	REPOSO	ARRANQUE	FUNCIONANDO
REGULADOR+HX711+HC-SR04	NA	NA	0,24
REGULADOR+L293D	0,36	NA	1,68
REGULADOR+SERVO946R	0,24	NA	0,42
REGULADOR+ESP32+RELE	2,16	NA	4,20
REGULADOR+ESP32 (DEEP SLEEP) +RELE	NA	NA	0,48
Driver Centrifuga L298N	0,48	24	14,40
Driver sinfín TB6560	1,44	NA	2,40

Fuente: Investigadores

Tabla 40: *Tiempo de actividades*

Cargas	Tiempos (Horas)		
	REPOSO	ARRANQUE	FUNCIONANDO
REGULADOR+HX711+HC-SR04	NA	NA	24,00
REGULADOR+L293D	0,44	NA	0,01
REGULADOR+SERVO946R	0,44	NA	0,01
REGULADOR+ESP32+RELE	11	NA	0,44
REGULADOR+ESP32 (DEEP SLEEP) +RELE	NA	NA	13,00
Driver Centrifuga L298N	0,44	0,01	0,01
Driver sinfín TB6560	0,02	NA	0,03

Fuente: Investigadores

Para determinar los tiempos, se ha tomado en cuanta los intervalos máximos recomendados para la alimentación de los alevines de trucha (10 veces al día). La frecuencia de alimentación disminuye a medida que la trucha crece, por ejemplo, para la última etapa (engorde) solamente hay que alimentarlas 2 veces al día. Por lo tanto, el proyecto al estar enfocado en la alimentación de los alevines se limitará al número máximo recomendado para esta etapa. Con lo indicado anteriormente, se evita el sobre dimensionamiento innecesario del sistema.

Los componentes HX711 y HC-SR04 son los encargados de pesar el alimento y medir el nivel del dispensador respectivamente. Estos sensores funcionan las 24 horas del día, debido a su bajo consumo de corriente (15mA – 20mA) no influye en la capacidad final de la batería, por lo que no se controla la activación/desactivación.

Para el análisis del consumo de la ESP32 y RELÉ se ha tomado en cuenta los horarios de alimentación para los alevines. Se recomienda empezar con la alimentación desde las 7:00 AM hasta las 5:00 PM. Por lo tanto, se necesita que el circuito de control y potencia se encuentren encendidos en ese lapso de tiempo para cumplir con los horarios establecidos o para registrar futuros cambios. Tomando en cuenta lo descrito anteriormente, la ESP32 y RELÉ estarán funcionando en reposo durante 11 horas es decir desde las 7:00 AM – 6:00 PM. Hay que mencionar que para el circuito de potencia y control el estado se reposo es cuando la ESP32 esta encendida y el RELÉ está en reposo, es decir, su bobina no está activada. Y el estado de funcionamiento es cuando la ESP32 y RELÉ están encendidos. Para el estado de funcionamiento se ha

tomado en cuenta el tiempo de activación de RELÉ que es 1 minuto antes del horario de dispensación de alimento. Se ha tomado en cuenta el tiempo que se mantiene activado para permitir el giro del sinfín para dispensar una media libra (alimento máximo que se puede dispensar en el estanque de alevines) que es 100 segundos aproximadamente. La suma de los tiempos dará 160 segundos, este valor se multiplica por el número máximo de repeticiones (10 veces al día), obteniendo 1600 segundos que pasando a horas son 0,44 horas. A partir del horario de 6:00PM hasta las 6:50AM del día siguiente la ESP32 entra en modo DEEP SLEEP para reducir su consumo.

Para los circuitos restantes mencionados anteriormente, se activa un minuto antes de empezar a dispensar el alimento y desactiva cuando terminen con la actividad designada, evitando el desperdicio de la capacidad de la batería en los intervalos de reposo.

Por lo tanto, el driver TB6560 encargado de mover el sinfín al encenderse 60 segundos antes y durante 10 veces al día, consumirá un tiempo de 600 segundos en reposo que en horas es aproximadamente 0,02 horas. El tiempo máximo en funcionamiento del TB6560 para mover el sinfín es 100 segundos y durante 10 veces al día da un aproximado de 0,03 horas.

La centrífuga durante el arranque consume una corriente aproximada de 2A durante 4 segundos y con una frecuencia de 10 veces da un tiempo total de 0,01 horas. Al encenderse 60 segundos antes y permanecer 100 segundos adicionales debido al sinfín durante 10 veces al día se encuentra en reposo 0,44 horas. Para dispersar el alimento se toma 4 segundos que al final del día ocupa un total de 0,01 horas. Este mismo análisis se puede aplicar para los circuitos del servomotor y motores-vibradores del dispensador.

Tabla 41: *WH*

Cargas	WH		
	REPOSO	ARRANQUE	FUNCIONANDO
REGULADOR+HX711+HC-SR04	NA	NA	5,76
REGULADOR+L293D	0,16	NA	0,02
REGULADOR+SERVO946R	0,11	NA	0,00
REGULADOR+ESP32+RELE	23,76	NA	1,86
REGULADOR+ESP32 (DEEP SLEEP) +RELE	NA	NA	6,24
Driver Centrifuga L298N	0,21	0,13	0,16
Driver sinfín TB6560	0,02	NA	0,07

Fuente: Investigadores

Dimensionado para el banco de baterías

Sumando todos los consumos diarios da un total de 38,51 WH/día.

Para mantener un margen de seguridad se aplica la siguiente fórmula [42, pp. 57-59].

$$Consumo_{total} = Consumo_{CC} + \frac{Consumo_{CA}}{1 - k_c}$$

$$Consumo = \frac{Consumo_{total}}{R}$$

$$R = (1 - k_b - k_c - k_r - k_v) * \left(1 - k_a * \frac{N}{Pd}\right)$$

Donde:

R: Es el rendimiento global

k_a: Coeficiente de pérdida debido a la autodescarga del acumulador

k_b: Coeficiente de pérdidas debido al rendimiento de los acumuladores

k_c: Coeficiente de pérdidas del inversor

k_r: Coeficiente por pérdidas del regulador de carga

k_v: Coeficiente de pérdidas debido a otros factores

N: Días de autonomía del sistema fotovoltaico

Pd: Profundidad de descarga de la batería

Tabla 42: *Descripción de los coeficientes del rendimiento global de la instalación*

Coeficientes	Descripción
k_a	Acumulador (Níquel-Cadmio) = 0,02
	Acumulador (Plomo-Ácido) = 0,005
	Acumulador de alta autodescarga = 0,012
k_b	Descargas intensas = 0,1
	Sin descargas intensas = 0,05
k_c	Sin inversor = 0
	Inversor onda pura = 0,05
	inversor onda cuadrada = 0,4
k_r	Regulador eficiente = 0,1
	Regulador ineficiente = 0,15
k_v	Si no se ha tomado en cuenta las perdidas en conductores y equipos = 0,1
	Si se ha realizado un estudio previo = 0,05
Pd	Profundidad de descarga del acumulador
	batería descarga un 80% = 0,8
	batería descarga un 60% = 0,6
	batería descarga un 50% = 0,5

Fuente: [42, p. 58]

Por lo tanto, para el dispensador se tiene un rendimiento global

$$R = (1 - 0{,}1 - 0 - 0{,}1 - 0{,}1) * \left(1 - 0{,}005 * \frac{1}{0{,}7}\right)$$

$$R = 0{,}793$$

$$Consumo_{total} = 38{,}51\ \text{WH/día} + 0$$

$$Consumo = \frac{38{,}51\ \text{WH/día}}{0{,}695} = 55{,}41\ \text{WH/día}$$

Para la capacidad nominal de la batería se utilizará la siguiente formula.[43]

$$Capacidad_{bateria} = \frac{Consumo * N}{Pd * V_{bateria} * \left(1 - \frac{\Delta t}{160}\right)}$$

Donde:

$V_{bateria} = Voltaje\ nominal\ de\ la\ bateria\ que\ se\ va\ implementar$

$\Delta t = Temperatura\ de\ funcionamiento\ de\ la\ bateria$

$$Capacidad_{bateria} = \frac{55{,}41 * 1}{0{,}7 * 12 * \left(1 - \frac{15}{160}\right)} = 7{,}28\ Ah$$

Por lo tanto, una batería comercial que se acerca a dicho valor es una de 7,5 Ah

Dimensionado de los paneles solares

Para determinar el número de paneles solares se necesita como dato el HSP, para ello se hará uso de la siguiente tabla.

Tabla 43: *Tabla de insolación difusa en Ecuador*

Mes	WH/m^2/dia
Enero	2440
Febrero	2336
Marzo	2296
Abril	2204
Mayo	1687
Junio	1367
Julio	1289
Agosto	1621
Septiembre	1828
Octubre	2011
Noviembre	1959
Diciembre	2026
Promedio	1922

Fuente: [44]

Para obtener HSP (Horas Sol Pico) hay que dividir el valor promedio entre 1000. Por lo tanto, el HSP para el sistema es 1.922. Además, hay que tener en cuenta que los paneles solares tendrán un rendimiento del 90% debido a la incidencia de sombras, suciedad, etc. Entonces al valor HSP se multiplica por 0.9 dando como resultado un nuevo valor de $HSP_{90\%}$ = 1.730.

Para el cálculo de la potencia pico para determinar el panel solar se hará uso de la siguiente fórmula.

$$P_p = \frac{Consumo}{HSP_{90\%}} = \frac{55{,}41}{1{,}730} = 32{,}02\ W_p$$

Por lo tanto, el panel solar adecuado para generar la energía necesaria para el dispensador debe ser superior al valor calculado [15].

Dimensionado del regulador de carga

El panel solar que cumple con los requisitos de la potencia calculada anteriormente tiene las siguientes características.

Tabla 44: *Características técnicas del panel solar*

Tipo	**Poli - cristalino**
Potencia máxima	50W
Voltaje circuito abierto	22,5 VDC
Corriente de corto circuito	2,86A
Voltaje en potencia máxima	18,7 VDC
Corriente en potencia máxima	2,68A

Fuente: Investigadores

Para determinar la corriente generada se aplicará la siguiente fórmula [45].

$$Corriente_{generada} = 1{,}25 * NP_p * I_{SC}$$

Donde

NP_P: Número de paneles en paralelo

I_{SC}: Corriente de corto circuito

∴

$Corriente_{generada} = 1{,}25 * 1 * 2{,}68 = 3{,}35A$

Por lo tanto, para cumplir con los requerimientos calculados hay que implementar un regulador de carga que soporte en su entrada al menos 5A,

Cálculo de la inclinación del panel solar

La inclinación del panel solar está en función de la latitud en donde se va ubicar. Si la ubicación es el hemisferio Norte, el panel tendrá una orientación hacia el Sur, mientras si la ubicación es al Sur, el panel se orientará hacia el Norte. En este caso, al estar en el hemisferio Sur la implementación del proyecto se ubicará con orientación al Norte.

El consumo del sistema es constante a lo largo del año. Por lo tanto, se busca obtener la máxima cantidad de insolación en los meses que haya poca o nula energía solar [46, p. 114].

En este caso se hará uso de la siguiente ecuación

$$\beta = |\varphi| + 10^{\circ}$$

Donde

β: Ángulo de inclinación del panel solar

φ: Latitud de la ubicación del proyecto

$\therefore$

$$\beta = |\varphi| + 10^{\circ} = 1^{\circ} + 10^{\circ} = 11^{\circ}$$

Con esto se puede concluir que el panel solar se ubicó con orientación hacia el Norte y con un ángulo de elevación de 11°

Programación del dispensador

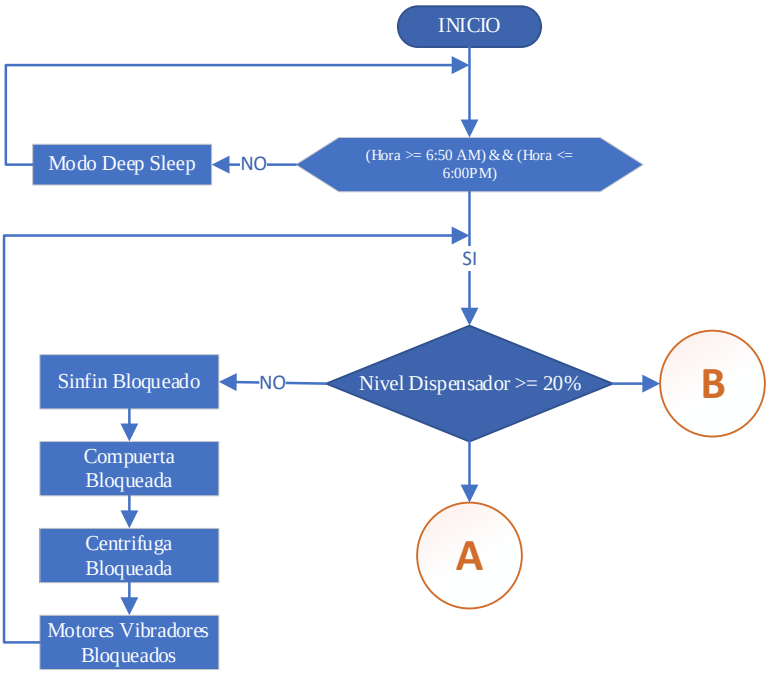


Figura 51: *Diagrama de flujo del dispensador. Parte 1*

Fuente: Investigadores

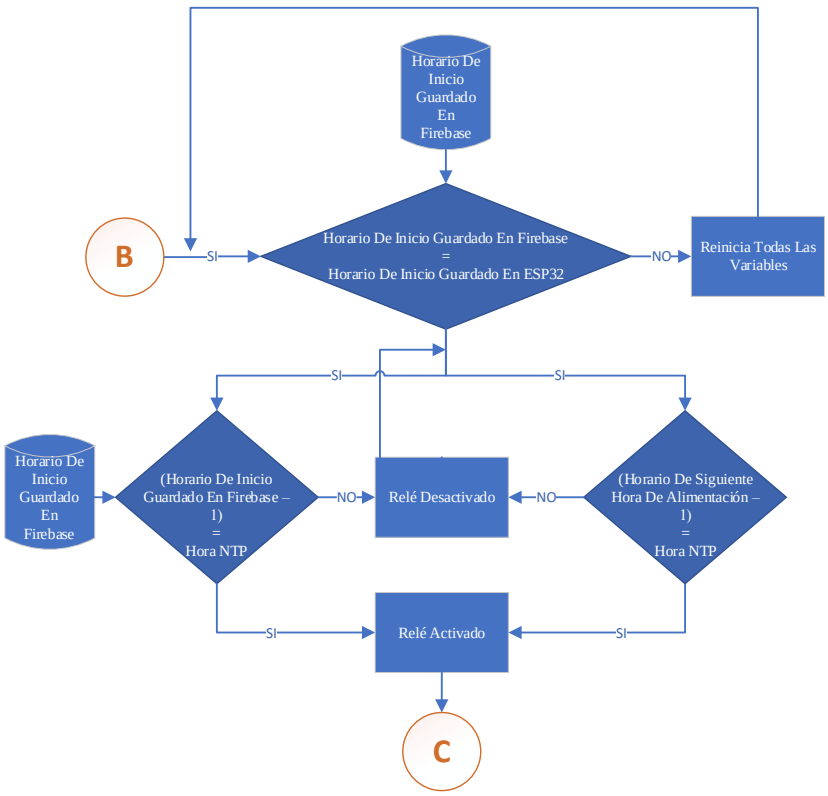


Figura 52: *Diagrama de flujo del dispensador. Parte 2*

Fuente: Investigadores

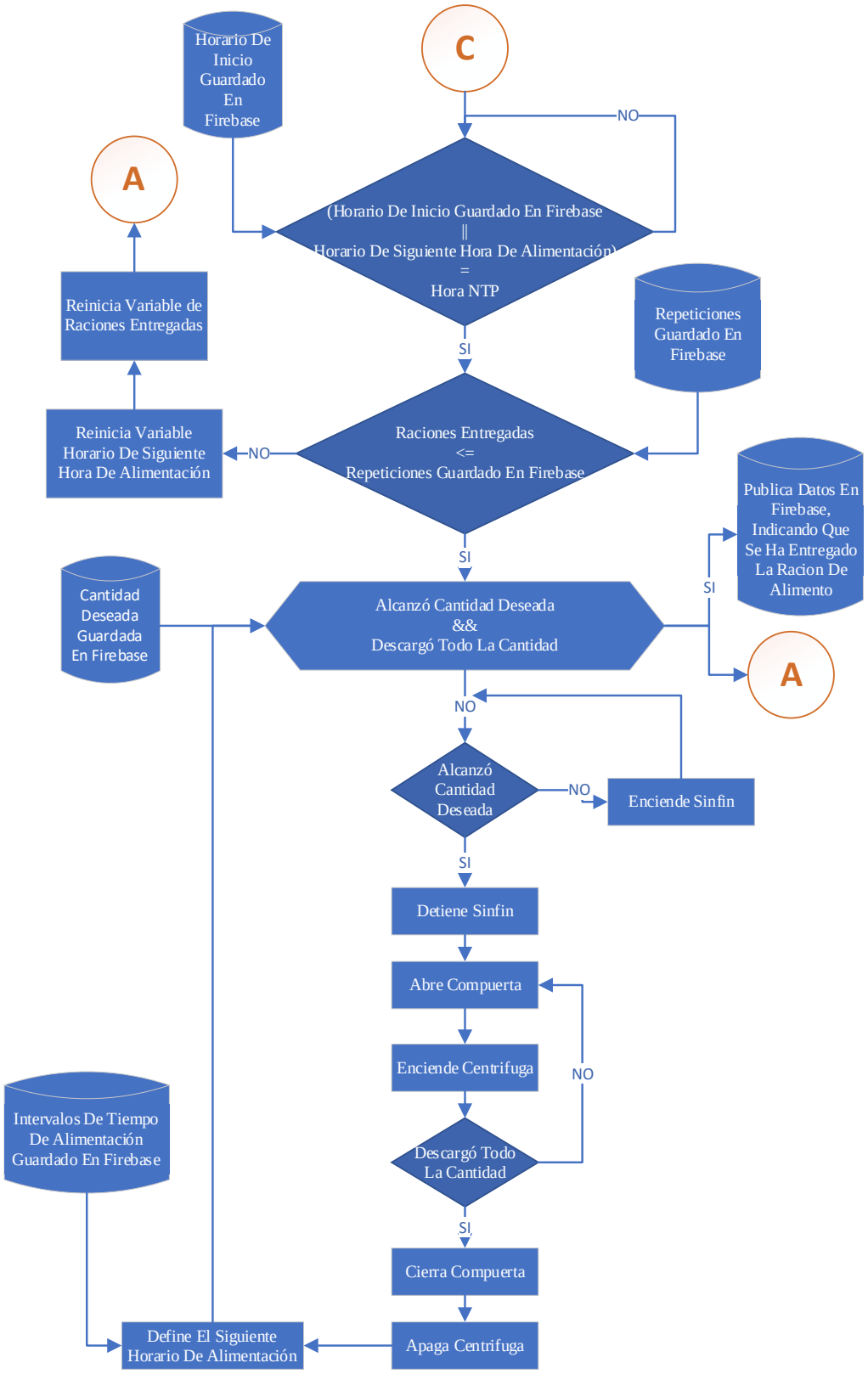


Figura 53: *Diagrama de flujo del dispensador. Parte 3*

Fuente: Investigadores

Tabla 45: *Librerías instaladas durante la programación*

Librería	Descripción
#include <WiFi.h>	Esta librería proporciona las funcionales WIFI para la tarjeta ESP32.
#include<Firebase_ESP_Client.h>	Crea la conexión entre la ESP32 con FIREBASE. Además, permite gestionar la información almacenada en FIREBASE (CRUD).
#include <NTPClient.h>	Permite implementar el protocolo NTP para obtener las horas, minutos y segundos desde los servidores NTP en Internet.
#include <math.h>	Proporciona funcionalidades matemáticas
#include <HX711_ADC.h>	Librería compatible con el módulo HX711 para la celda de carga
#include <AccelStepper.h>	Entrega funcionalidades a la ESP32 para el control de motores a paso.
#include <Servo.h>	Permite controlar un servomotor.
#include "NewPing.h"	Librería compatible con el sensor ultrasónico HC-SR04.
#include <EEPROM.h>	Permite acceder a la memoria no volátil de la placa para almacenar variables. En caso de reinicio o apagado de la ESP32 aun conservará el valor de las variables.

Fuente: Investigadores

A continuación, se describirán los tramos de código más relevantes dentro de la programación del dispensador de alimento.

Funcionamiento en intervalos de tiempo establecidos

Toda la programación se incluye dentro de la función “void setup()” porque al implementar el modo Deep Sleep en la ESP32, todo el código se ejecuta desde el setup y entra en el modo sueño profundo después de ejecutar todos las líneas de programación. Cuando termine el ciclo de sueño, después de determinado tiempo la ESP32 ejecuta de nuevo el código y después entra en el modo sueño otra vez. Este ciclo se repite infinidad de veces.

El siguiente código define la condición necesaria para ejecutar el programa principal (dentro de ella se especifica las rutinas del dispensador de alimento) en determinadas horas y pasado ese tiempo entrar en el modo Deep Sleep para aumentar el rendimiento de la batería.

Dentro del ciclo while la variable “sumas” hace referencia a las sumas de las horas (hora en minutos) y de los minutos obtenidos de los servidores NTP. En este caso el valor 420 equivale a 7 horas y 0 minutos. Mientras el valor 1080 equivale a 18 horas y 0 minutos. Entonces, en este rango de tiempo desde las 7AM hasta las 6PM el dispensador estará activo para ejecutar sus tareas rutinarias. Pasado ese tiempo la

variable “sumas” se reiniciará y entrará en el modo sueño profundo mediante la línea “esp_deep_sleep_start();”. El tiempo máximo de sueño que permitió implementar la ESP32 sin provocar errores fue de 12 minutos, por lo que, durante la noche se llega a despertar 65 veces.

```
 while(sumas>=420 && sumas <= 1080) {
   timeClient.update();
   HorasNew = timeClient.getHours();
   MinutesNew = timeClient.getMinutes();
   sumas = (HorasNew*60) + MinutesNew;
   principal ();
  }

 sumas = 0;
  esp_deep_sleep_start();

}
```

Reinicio de variables al cambiar el horario de alimentación

El siguiente bloque de programación permite determinar si ha habido cambios en el horario inicial de la rutina de alimentación definido con anterioridad. En caso de existir cambios todas las variables son devueltas a sus valores por defecto para empezar con la nueva rutina definida. Para ello la variable “compararSumaTiempo” almacena la suma de la hora (en minutos) con los minutos que son tomados desde FIREBASE. Por lo tanto, al ingresar al condicional if, la variable “compararTiempo” tomará el valor la variable “compararSumaTiempo” recordando así, el horario que se ha definido en FIREBASE, esto valor se actualizará cada 5 segundos. En caso de que se cambie el horario de alimentación la variable “compararSumaTiempo” actualizará su valor, pero la variable “compararTiempo” al actualizarse cada 5 segundos, conservará todavía el valor anterior, por lo que habrá una “desincronización” haciendo que la condición no se cumpla y se reinicien todas las variables para empezar con la nueva rutina.

```
  compararSumaTiempo = intHora + intMinutos;

 if (compararTiempo == 0 || compararTiempo == compararSumaTiempo){
   if (millis() - compararTiming > 5000){
     compararTiming = millis();
     compararTiempo = compararSumaTiempo;

   }
} else {
       if (millis() - compararTiming > 5000){
         compararTiming = millis();
         compararTiempo = 0;

       }
```

```
intVueltas = 0;
intTiempoTotal = 0;
sumaIntervalos = 0;
newHora = 0;
finalHora = 0;
finalauxiliar = 0;
finalMinutos = 0;

}
```

Encendido y apagado de los actuadores

Para reducir al mínimo el consumo de energía hay que encender y apagar los actuadores en función del relé. El relé se enciende y apaga al ocurrir determinadas circunstancias.

Tabla 46: *Activación y Desactivación del relé*

Encendido	Apagado
1. Un minuto antes de la primera hora definido en el horario de alimentación guardado en FIREBASE.	1. Un minuto después de haber dispersado la comida durante cada intervalo de tiempo indicado.
2. Un minuto antes se la siguiente hora de alimentación definido previamente por la ESP32.	2. Un minuto después de haber hecho cambios en los horarios de alimentación definidos con anterioridad.

Fuente: Investigadores

Condición de encendido de relé

1. **Un minuto antes de la primera hora definido en el horario de alimentación guardado en FIREBASE.**

La variable “finalHoraActuadores” es producto de la suma de la hora (en minutos) y los minutos tomados desde FIREBASE menos 1 minuto, esto permite encender el relé un minuto antes de empezar a dispersar el alimento al comparar el valor de la variable con el tiempo tomado desde los servidores NTP.

```
if (habilitarActuadores == true){
  digitalWrite(rele, HIGH);
 }

 if (((finalHoraActuadores) == timeClient.getHours()) && (finalMinutosActuadores ==
timeClient.getMinutes())){
  Serial.println("Los actuadores están encendidos");
  habilitarActuadores = true;
  cicloActuadores = false;
  repeticionActuadores = true;
 }
```

2. **Un minuto antes se la siguiente hora de alimentación definido**

La variable “finalHoraCiclo” hace referencia a la suma de la hora y minutos del siguiente horario de alimentación definida por la ESP32 menos 1 minuto. Esto permite encender el relé un minuto antes de cada intervalo de alimentación al comparar la variable con el tiempo tomado desde NTP.

```
if(actuadoresHorario == true){
  digitalWrite(rele, HIGH);
 }

 if (((finalHoraCiclo) == timeClient.getHours()) && (finalMinutosCiclo ==
timeClient.getMinutes())){
  Serial.println("Los actuadores están encendidos");
  repeticionActuadores = true;
  cicloActuadores = false;
  actuadoresHorario =true;
 }
```

Condición de apagado de relé

1. **Un minuto después de haber dispersado la comida durante cada intervalo de tiempo indicado.**

La variable “cicloActuadores” es de tipo booleana por lo que al finalizar cada rutina de alimentación esta cambia a true. Adicionalmente, la variable “repeticionActuadores” es puesta en false, haciendo que el relé se desactive.

```
if (cicloActuadores == true){
   repeticionActuadores = false;
   habilitarActuadores = false;
   actuadoresHorario =false;
  }

 if (repeticionActuadores == false){
  delay(2000);
  Serial.println("Se ha apagado los actuadores");
  digitalWrite(rele, LOW);
 }
```

2. **Un minuto después de haber hecho cambios en los horarios de alimentación definidos con anterioridad.**

La variable “diferenciaHorario” se obtiene mediante la diferencia entre la suma del tiempo obtenido de los servidores NTP y la suma del tiempo del horario inicial guardado en FIREBASE. Si el valor de esa diferencia es mayor a 2, se apagarán los actuadores. Por ejemplo, como suposición se puede decir que es 10AM y el usuario desea definir un nuevo horario que empiece a las 8AM del día siguiente, por lo que

para empezar a dispersar hay una diferencia de 22 horas, por lo que es tiempo muerto en donde la bobina del relé estará encendida consumiendo recursos de la batería. Por lo tanto, la implementación del siguiente código permite apagar el relé y a la vez los actuadores.

```
if (((diferenciaHorario > 2)||(((-1)*(diferenciaHorario)) > 2)) && intVueltas == 0){
    Serial.println("Los actuadores están APAGADOS     DEBIDO AL CAMBIO
    DE HORARIO");
  habilitarActuadores = false;
  actuadoresHorario =false;
  digitalWrite(rele, LOW);
}
```

Definición del nuevo horario de alimentación

Cuando se cumpla el tiempo del horario de alimentación para empezar a dispensar la comida se ejecutará las siguientes variables en conjunto con las demás líneas de código. La variable "intVueltas" se incrementa cada vez que se cumpla el tiempo para entregar la comida. La variable "sumaIntervalos" suma el tiempo de intervalos que se guardó en FIREBASE previamente, esa suma se multiplica por el número de repeticiones que se haya cumplido. Esa nueva cantidad entre sumas y productos será agregada al horario inicial ("intTiempoTotal") guardado en FIREBASE para definir la siguiente hora que se debe entregar la comida.

```
        intVueltas = intVueltas + 1;
intTiempoTotal = intHora + intMinutos;
      sumaIntervalos = (intIntervalosHora +   intIntervalosMinutos)*intVueltas;
newHora = sumaIntervalos+intTiempoTotal;
finalHora = newHora/60;
finalauxiliar = newHora/60;
finalMinutos = (round((finalauxiliar-finalHora)*60));
```

Publicación de datos en FIREBASE

Después de dispensar cada ración de comida se publicó un enunciado indicando que se ha dispensado la comida. Para ello la variable previamente definida "ramdondato" almacena un valor de 255, decrementando cada vez que se entrega una ración. Este valor es utilizado como ID que sirve para identificar cada nodo dentro de FIREBASE. Además, este ID es tomado desde la página web y aplicación móvil para ir mostrando en pantalla un historial de las raciones de comida que se han entregado.

```
        --ramdondato;
EEPROM.write(0, ramdondato);
EEPROM.commit();
```

```
String datosssss= String(ramdondato);

                    if (Firebase.RTDB.setString(&fbdo, "/Base de Datos-Comida/" + datosssss
                    + "/comida", "Se ha dispensado " + String(floatCantidad+7) + " gramos a
                    las: " + String(horacomidanew)+ ":" + String(minutocomidanew)+ " " +
                    "con fecha " +
                    String(monthDay)+"/"+currentMonthName+"/"+String(currentYear))){
}
 else {
   Serial.println("Error");
   Serial.println("Causa: " + fbdo.errorReason());
 }
```

Programación del contador

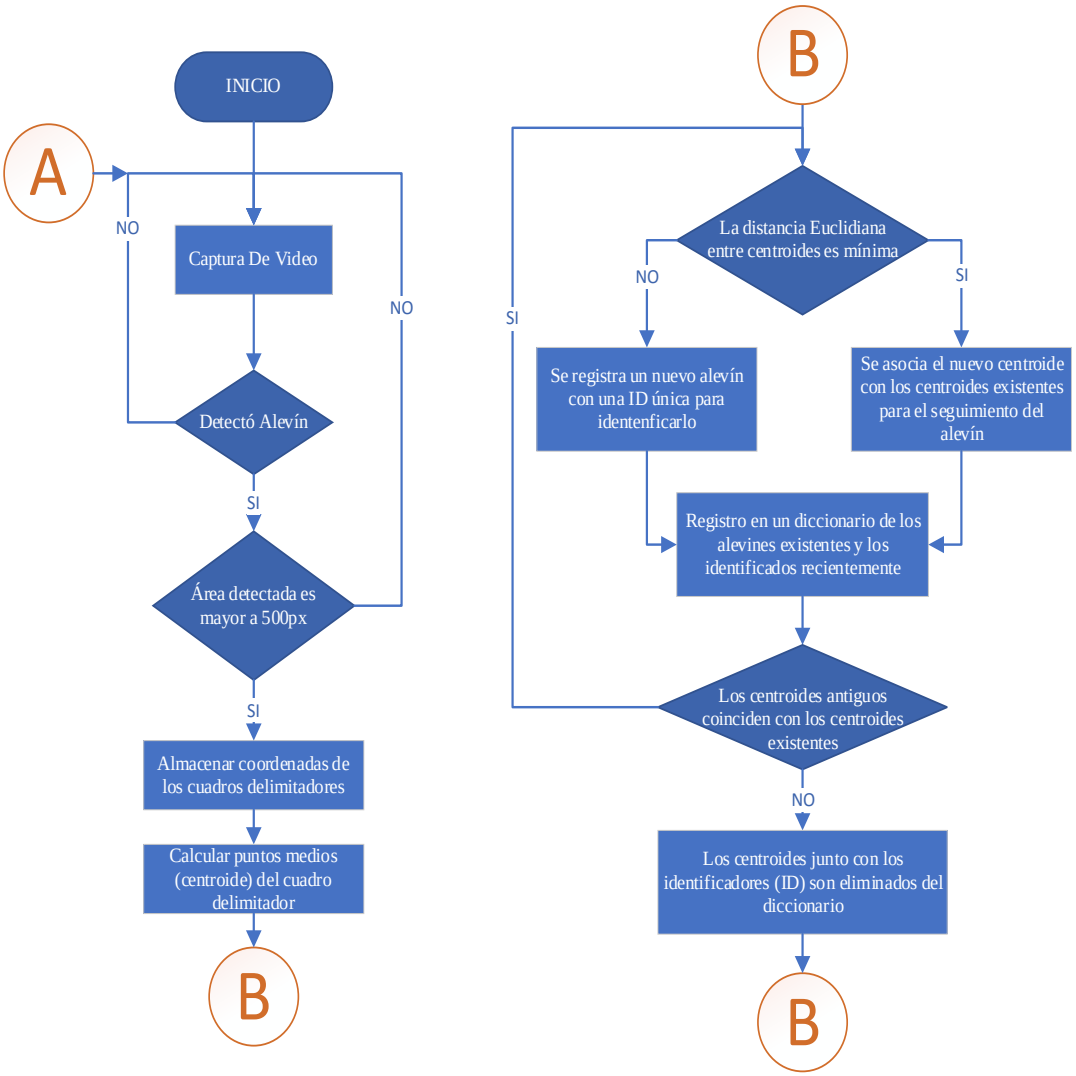


Figura 54: *Diagrama de flujo del sistema de visión artificial. Parte 1*

Fuente: Investigadores

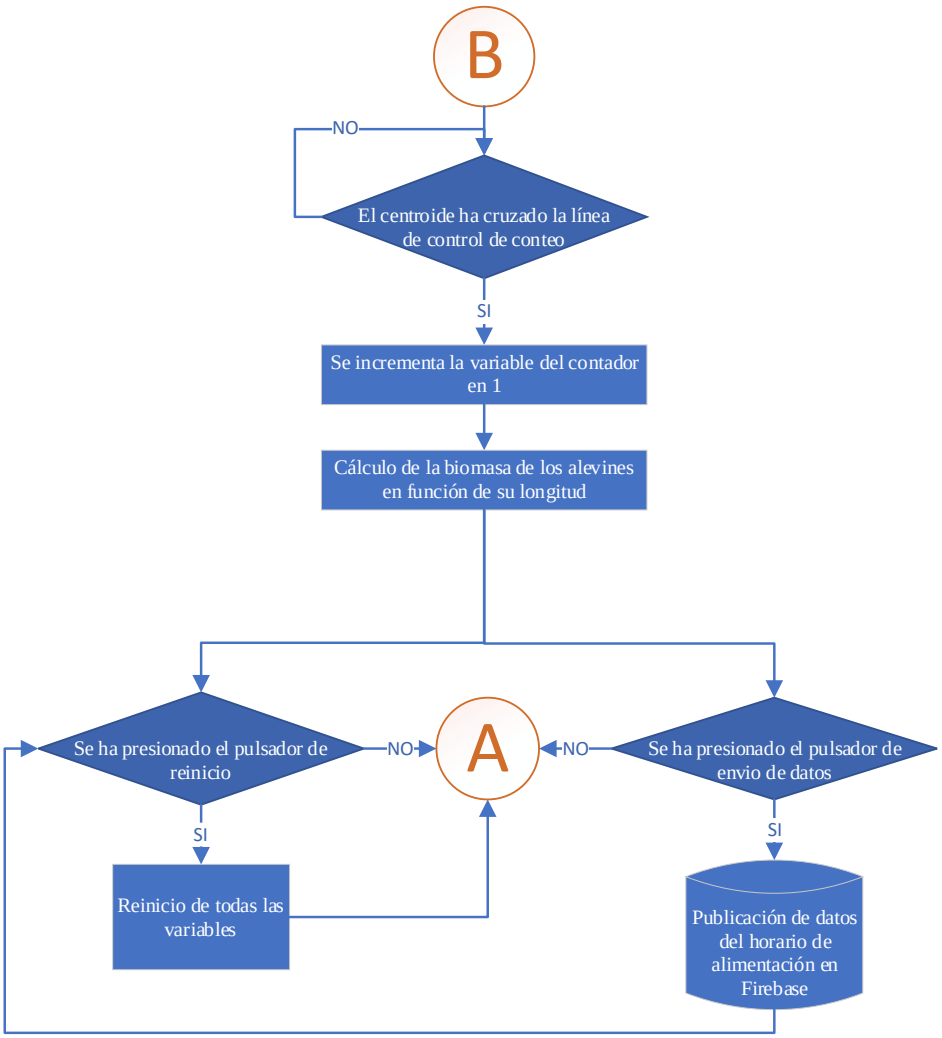


Figura 55: *Diagrama de flujo del sistema de visión artificial. Parte 2*

Fuente: Investigadores

Tabla 47: *Librerías instaladas durante la programación*

Librería	Comando de instalación	Descripción
math	Librería incluida de forma predeterminada en el paquete Python.	Permite usar funcionalidades matemáticas durante la programación.
RPLCD	➢ sudo pip3 install RPLCD ➢ sudo apt-get install python-smbus	Proporciona funcionalidades para la comunicación con pantallas LCD con módulos I2C
imutils	sudo pip3 install imutils	Entrega herramientas para el procesamiento de imágenes como el redimensionamiento o la rotación.
numpy	Librería incluida de forma predeterminada en el paquete Python.	Esta librería facilita el trabajo con arrays.
cv2	sudo pip3 install opencv-python	Librería utilizada para trabajar con temas relaciones con la visión por computadora
firebase_admin	sudo pip3 install firebase-admin	Permite la comunicación entre Python y la base de datos FIREBASE
datetime	Librería incluida de forma predeterminada en el paquete Python.	Permite obtener la fecha y hora.
OrderedDict	Librería incluida de forma predeterminada en el paquete Python.	Recuerda el orden de ingreso de datos en un diccionario.
distance	sudo apt-get install python3-scipy	Librería incluida dentro del paquete scipy.spatial. Permite el cálculo de distancias entre cada par de coordenadas.

Fuente: Investigadores

Estructura de la programación

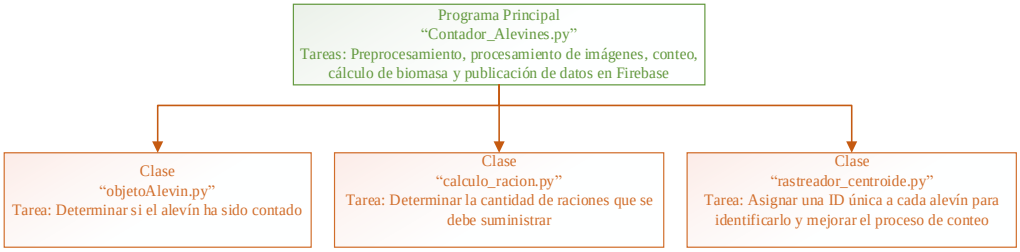


Figura 56: *Estructura de programación del contador*

Fuente: Investigadores

Calibración de la cámara

OpenCV proporciona herramientas para la calibración en su sitio oficial.

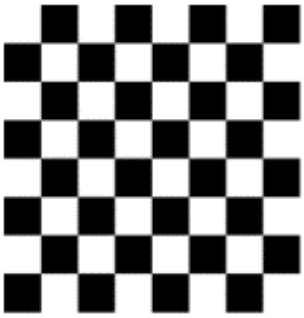


Figura 57: *Representación de una imagen ideal sin distorsión*

Fuente: [47, p. 8]

Las cámaras introducen dos tipos principales de distorsión en las imágenes.

- **Distorsión radial**

Introduce deformación en una imagen, a tal punto que las líneas rectas parecen curvas. Este efecto se incrementa a medida que el objeto se aleja del centro de la imagen.

Figura 58: *Distorsión radial en un tablero de ajedrez*

Fuente: [48]

Por ejemplo, en la figura superior se puede observar que la línea roja no coincide con el borde del tablero debido a que esta presenta mayor abultamiento en sus lados.

La distorsión radial puede ser representada de la siguiente manera.

$$x_{distorsionado} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{distorsionado} = y(1 + k_1 r^2 + k_2 r^4 + k_3 \quad ^6)$$

- **Distorsión tangencial**

Por otra parte, este tipo de distorsión es causada por la mala alineación del lente de la cámara o si la cámara no está alineada paralelamente con el plano de la imagen. Esto provoca que las imágenes capturadas parezcan más cerca de lo normal.

Figura 59: *Representación de la distorsión tangencial*

Fuente: [47, p. 8]

Se puede definir de la siguiente manera.

$$x_{distorsionado} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$

$$y_{distorsionado} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

Por lo tanto, para corregir estas deformaciones en la imagen, hay que encontrar los siguientes parámetros, definidos como coeficientes de distorsión.

$$Coeficientes\ distorsión = (k_1 + k_2 + p_1 + p_2 + k_3)$$

Además, hay que encontrar la matriz de la cámara. Para ello se hace uso de los parámetros intrínsecos o internos de la misma. Este parámetro describe las propiedades geométricas de funcionamiento como la distancia focal y el centro óptico de la cámara.

La matriz tiene un arreglo de 3x3. En su diagonal principal se encuentran los valores de la distancia focal (F_x, F_Y), mientras en la tercera columna los valores del centro óptico(C_x, C_Y), donde el eje óptico cruza con el plano de la imagen [47, p. 7].

$$matriz\ cámara = \begin{bmatrix} F_X & 0 & C_x \\ 0 & F_Y & C_y \\ 0 & 0 & 1 \end{bmatrix}$$

Pasos para la calibración

Se ha utilizado el método de calibración basado en un tablero de ajedrez, debido que los patrones de esta son fáciles de detectar dentro de una imagen.

1. Identificar las esquinas interiores (horizontal y vertical) del tablero de ajedrez.

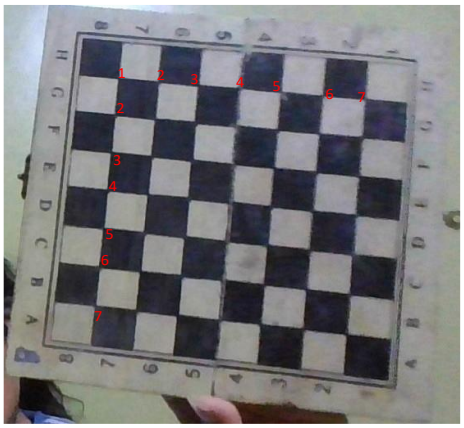


Figura 60: *Esquinas interiores (horizontal y vertical) del tablero de ajedrez*

Fuente: Investigadores

Por lo tanto, en este tablero hay 7 esquinas horizontales y 7 verticales. Este dato define el código de calibración.

```
chessboardSize = (7,7)
```

2. Desde la cámara que se desea calibrar, se toma al menos 27 fotografías en diferentes ángulos para obtener una buena calibración.

Figura 61: *Galería de imágenes capturadas*

Fuente: Investigadores

3. Al terminar con la ejecución del programa, este acoge los resultados con los coeficientes de distorsión y la matriz de la cámara y el nivel de error.

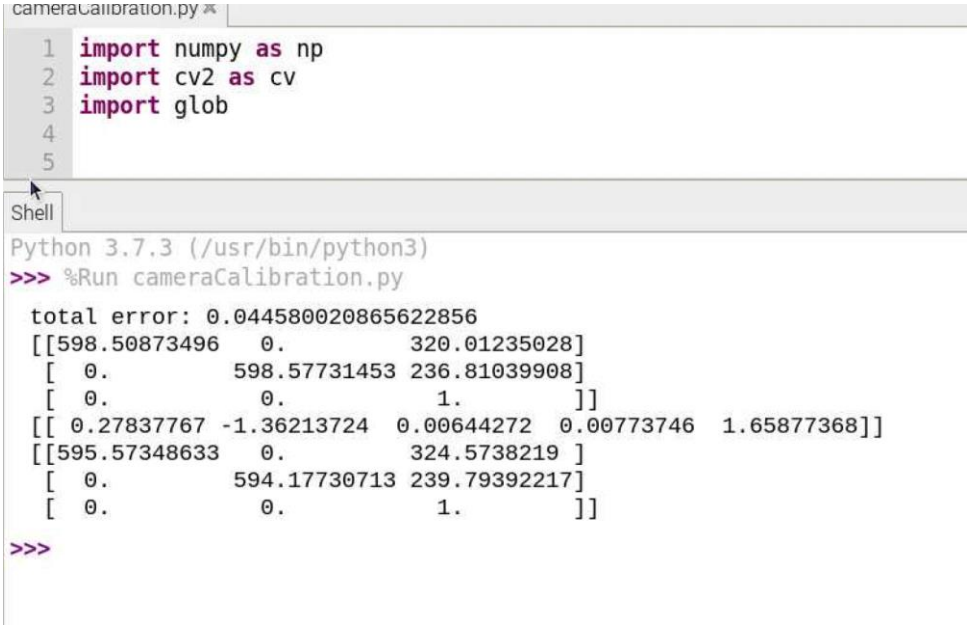


Figura 62: *Resultados de calibración*

Fuente: Investigadores

El parámetro de error, ayuda a determinar si la calibración ha sido eficiente. Entre más cerca este de cero, mejor será la calibración. En este caso el valor de error es 0,045 eso significa que la calibración fue exitosa.

4. Los datos obtenidos durante la calibración se pueden almacenar en matrices para ser implementadas más adelante para el contador de alevines.

```
distCoeffs = np.array([[0.278,  -1.362, 0.006,  0.008, 1.659]])

cameraMatrix = np.array([[595.573,   0,         324.574],
                         [  0,         594.177, 239.794],
                         [  0,           0,           1]])
```

5. Mediante OpenCV y el comando “undistort” se ha implementado los parámetros de coeficientes de distorsión y matriz de la cámara en el programa principal “Contador_Alevines.py”.

```
frame = cv2.undistort(corregido, cameraMatrix, distCoeffs, None)
```

Clases implementadas en Python

Clase “objetoAlevin.py”

Esta clase permite almacenar las IDs de los alevines que ya han sido previamente identificados. Además, lleva el control mediante la variable “self.contado” si el alevín ha sido contado.

```
class ObjetoAlevin:
    def __init__(self, objectID):
        self.objectID = objectID
        self.contado = False
```

Clase “calculo_racion.py”

La clase “calculo_racion.py” permite calcular la cantidad de porciones de balanceado que se debe dispersar en un día, a una cantidad determinada de alevines. Para ello se utiliza una tabla que relaciona la talla promedio de los peces con el porcentaje de alimento que deben recibir en función de su biomasa.

La biomasa es calculada en el programa principal y pasada como parámetro hacia la clase “calculo_racion.py”, así como la talla promedio y el número de alevines.

Con todos los parámetros implementados se multiplica la biomasa, por el valor que corresponda a cada talla promedio (clave del diccionario “tabla”). Finalmente, se divide entre 8 (número de raciones diarias).

```
class Racion:
    def __init__ (self, biomasa, talla_media, alevines):
        tabla = {3:0.1169,
                 4:0.0869,
                 5:0.0691,
                 6:0.0574,
                 7:0.0504}
        self.biomasa = biomasa
        self.talla_media = talla_media
        self.alevines = alevines
        self.raciones = (self.biomasa*tabla[self.talla_media])/8
        self.horas = 8
        self.minutoss = 0
        self.intervaloHora = 1
        self.intervaloMinutos = 0
        self.repeticiones = 8
```

Clase “rastreador_centroide.py”

La implementación de esta clase permite hacer el seguimiento de los alevines identificados y asignarles una ID única para su posterior conteo. Además, en esta clase se agrega en un diccionario el centroide de los cuadros delimitadores con su respectiva longitud (estos serán los valores del diccionario) y también los IDs (claves del diccionario).

Esta clase se compone de tres funciones fundamentales.

1. **Función registro**

En esta función se ingresa como parámetro el centroide de los cuadros delimitadores. Como su nombre lo indica, en esta sección se registrarán los nuevos alevines que han sido identificados. Para ello, se hace uso del diccionario “self.objetos” que almacena como valor el centroide y como clave la ID. Además, se hace uso de la variable “self.nextObjetoID” que se incrementa (generando una clave nueva) cada vez que se llama a la función.

```
def registro(self, centroide):
    self.objetos[self.nextObjetoID] = centroide
    self.desaparecido[self.nextObjetoID] = 0
    self.nextObjetoID += 1
```

2. **Función suprimir**

Se llama a esta función cada vez que un objeto (alevín) ha desaparecido de los fotogramas posteriores. Por lo tanto, la palabra clave “del” permitirá eliminar ese objeto del diccionario en función del ID

```
def suprimir(self, objetoID):
    del self.objetos[objetoID]
    del self.desaparecido[objetoID]
```

3. **Función actualizar**

Esta función recibe únicamente como parámetro las coordenadas de los cuadros delimitadores, pero es en esta que se permite implementar el seguimiento de objetos del centroide.

```
def actualizar(self, rectangulos):
```

3.1. Seguimiento de objetos mediante el centroide

El algoritmo del rastreador del centroide necesita únicamente las coordenadas de los cuadros delimitadores, que son obtenidos previamente a través de un detector de objetos [49].

Mediante el ciclo for se extrae el índice y los valores del arreglo "rectángulos" para posteriormente calcular sus respectivos centroides y almacenarlos en otro arreglo "entradaCentroides" con su índice. Además, en otro arreglo se agrega la longitud de cada cuadro delimitador para calcular el peso de cada alevín mas adelante.

```
for (i, (inicioX, inicioY, finX, finY)) in enumerate(rectangulos):
     cX = int((inicioX + finX) / 2.0)
     cY = int((inicioY + finY) / 2.0)
     ancho = int(finY - inicioY)
     entradaCentroides[i] = (cX, cY)
     entradaAncho[i] = (ancho)
```

Antes de empezar con la implementación de código. A continuación, se muestra una ilustración para entender cómo funciona el algoritmo del rastreador.

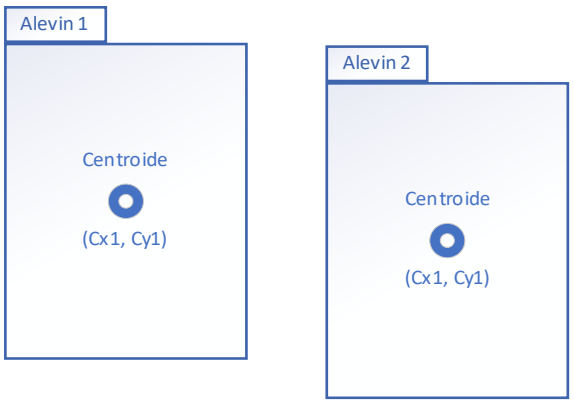


Figura 63: *Cuadros delimitadores con sus respectivos centroides*

Fuente: Investigadores

Como se puede observar en la imagen superior, se está representado los recuadros que envuelven al alevín después de pasar por la etapa de detección. Además, se ha dibujado los respectivos centroides de cada cuadro.

Hasta el momento se ha encontrado cada objeto (alevín). Pero, el detector solo se encarga de hallar nuevos objetos dentro de los fotogramas posteriores sin tomar en cuenta, si los recuadros delimitadores pertenecen al mismo objeto o a un objeto diferente, por lo que durante el proceso de conteo se llegará a obtener un valor erróneo.

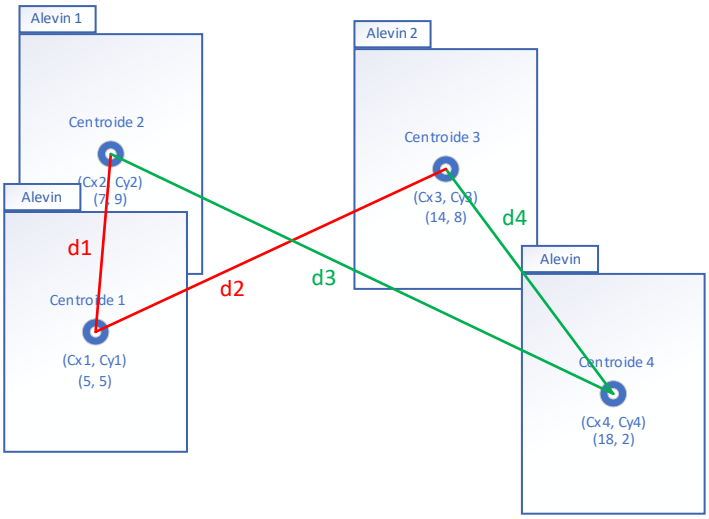


Figura 64: *Distancia euclidiana entre coordenadas de los centroides*

Fuente: Investigadores

Para obtener mejores resultados durante el proceso de conteo se ha implementado el algoritmo de rastreo de objetos mediante el centroide. Para ello se asigna a cada objeto una ID única para identificarla y asociar los centroides solamente con los respectivos objetos.

Como ejercicio práctico en la figura superior se ha planteado cuatro centroides con sus respectivos puntos y mediante las distancias euclidianas (valores mínimos) se buscará asociar cual centroide pertenece a cada objeto previamente detectado.

La distancia euclidiana se define mediante la siguiente formula.

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Entonces se tienen los siguientes puntos.

$$Centroide_1 = (5,5);\ Centroide_2 = (7,9);\ Centroide_3 = (14,8);\ Centroide_4 = (18,2)$$

Al hallar la distancia entre cada par de puntos se obtuvo los siguientes valores.

$$d_1 = 4{,}47;\ d_2 = 9{,}48;\ d_3 = 13{,}04;\ d_4 = 7{,}21$$

Por lo tanto, el centroide 1 forma parte del mismo objeto (alevín 1) debido que su distancia euclidiana $d_1 = 4{,}47$ está más cerca del centroide 2. La asociación con el centroide 3 queda descartada porque su distancia d_2 es mayor a d_1. Para los centroides 3 y 4 se aplica el mismo análisis para su relación de pertenencia.

Con lo planteado previamente, se puede deducir que el algoritmo de rastreo asocia cada nuevo centroide mediante el cálculo de las distancias mínimas por lo que entre menor sea la distancia entre dos puntos hay más probabilidad que pertenezca al mismo objeto.

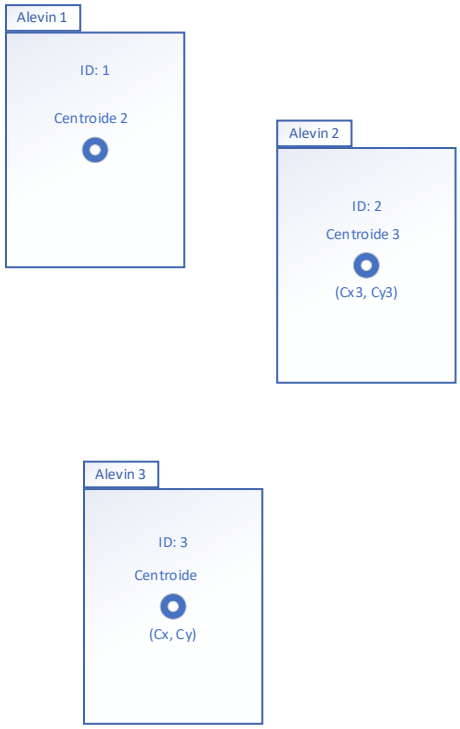


Figura 65: *Detección de un nuevo objeto*

Fuente: Investigadores

Cuando el algoritmo no pueda asociar un centroide con los existentes se deduce que es un nuevo objeto, por lo tanto, se le asigna una nueva ID única. El proceso de rastreo sigue hasta que el objeto desaparece del área de interés, entonces este es eliminado del diccionario de objetos.

En Python, la distancia euclidiana se calcula mediante la librería “distance”. Esta librería permite manejar pares de puntos.

El cálculo de las distancias es almacenado mediante un arreglo bidimensional en "Distancia"

```
filas = Distancia.min(axis=1).argsort()
```

$$filas_{indices} = [0 \quad 1] \qquad filas_{valores\ pequeños} = [0.48 \quad 0.89]$$

Para las columnas se hace lo mismo, se toma los índices de las columnas del arreglo bidimensional y se comparan las filas y se las ordena según el índice de las filas encontradas previamente.

```
columnas = Distancia.argmin(axis=1)[filas]
```

$$columnas_{indices} = [0 \quad 2] \qquad columnas_{valores\ pequeños} = [0.48 \quad 0.89]$$

Este proceso permite asociar en pares mediante la función "zip" cada fila con su respectiva columna para identificar cual es la distancia mínima en el arreglo bidimensional.

$$(fila, columna) = [(0,0) \quad (1,2)]$$

A modo de ejemplo se ha creado el arreglo "$(fila, columna)$" donde se agrupa fila y columna. Con esto se puede ubicar en el arreglo bidimensional "Distancia" donde se encuentra cada valor mínimo. La expresión (0,0) hace referencia que el valor mínimo se encuentra ubicado en el índice (0) de la fila y en el índice (0) de la columna, el cual corresponde al valor 0.48, lo mismo aplica para la expresión (1,2) donde hace referencia al valor 0.89.

Al mismo tiempo, los valores de la "fila" tienen como función identificar los valores de las claves del diccionario "objetos" para asociar los centroides de ingreso con las IDs existentes en función de las "columnas".

Este es el proceso que se sigue para enlazar los centroides de entrada con los existentes en función de su distancia mínima.

```
objetoIDs = list(self.objetos.keys())
objetoID = objetoIDs[fila]
self.objetos[objetoID] = lista[columna]
```

Programa principal "Contador_Alevines"

Dentro del programa se realiza todo el proceso encargado de la visión artificial (se explicará más adelante los pasos que se han aplicado) para la detección de los alevines.

Al instanciar la Clase "RastreadorCentroide" este devuelve el diccionario "objetos" con la siguiente información.

$$objetos = \{ID: ((Cx, Cy), (longitud))\}$$

Donde:

$ID: es\ el\ identificador$ único $de\ cada\ alevin$

$(Cx, Cy): es\ el\ centroide\ de\ cada\ cuadro\ delimitador$

$(longitud): es\ la\ altura\ del\ cuadro\ delimitador\ que\ envuelve\ al\ alevin$

1. **Proceso de conteo**

Para este proceso se trazó una línea de apoyo para determinar si el centroide ha pasado por esta. Caso contrario el alevín no será contado.

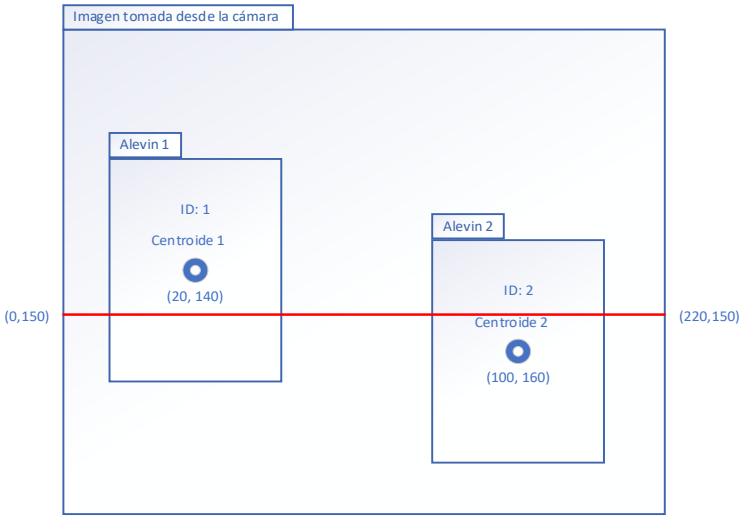


Figura 66: *Representación del proceso de conteo*

Fuente: Investigadores

Con fines ilustrativos la línea roja, está en la ubicación $P_1 = (0{,}150)$ y $P_2 = (320{,}150)$. Del centroide se ha obtenido los datos (Cx, Cy). El sentido del movimiento solamente ocurre en eje de las ordenadas por lo tanto para el conteo solo se tomará el valor Cy.

Como se puede observar en la imagen superior, el "centroide 1" ($Cy = 140$) todavía no ha cruzado la línea roja (ubicada en eje de la ordenada 150), por lo tanto, el contador de alevines no se incrementa. Pero, el "centroide 2" ($Cy = 160$) ya ha cruzado la línea por lo tanto el contador se incrementará una unidad. Este proceso se sigue durante todo el proceso de conteo. A continuación, se mostrará el proceso mediante código.

```
if 150< centroid[0][1] < 220:
    cv2.line(frame, (0,150), (320,150), (0,255,0), 4)
    ContadorAlevines += 1
```

2. Encontrar relación de proporción de pixeles a cm

Para medir el tamaño de los objetos mediante la cámara hay que determinar la distancia que hay entre el objeto de interés y la cámara.

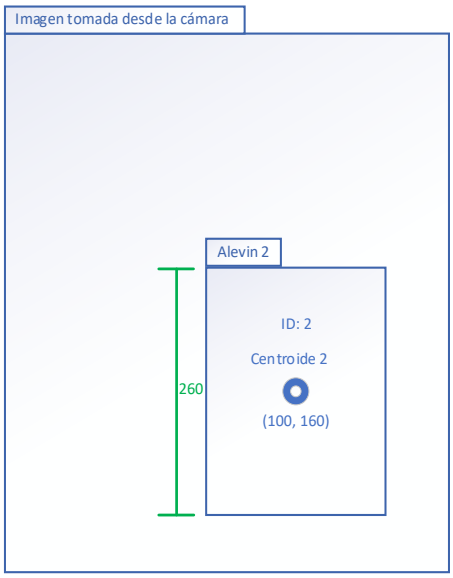


Figura 67: *Representación de la altura del cuadro delimitador en pixeles*

Fuente: Investigadores

Por ejemplo, la longitud promedio de un alevín es 4 centímetros, entonces a una determinada distancia de la cámara se obtendrá un valor en pixeles de 260 unidades.

Entonces, para obtener una relación entre pixeles y cm, hay que dividir los pixeles entre el valor conocido.

$$pixeles\_to\_cm = \frac{260}{4} = 65$$

Esta relación puede ser utilizada para calcular el tamaño de todos los alevines que vayan apareciendo. Por ejemplo, si un alevín mide en pixeles 300 unidades, para hallar su valor en centímetros se divide por la relación (65) y su tamaño sería de 4,62 centímetros.

Para hallar una relación de pixeles a cm constante, se ha aplicado marcadores aruco. Cada aruco maker mide un determinado tamaño. En este caso se está usando un aruco de 5cm a cada lado, es decir, un cuadrado. OpenCV proporciona herramientas para implementar marcadores aruco.

```
parametros = cv2.aruco.DetectorParameters_create()
aruco_dicc = cv2.aruco.Dictionary_get(cv2.aruco.DICT_4X4_100)
```

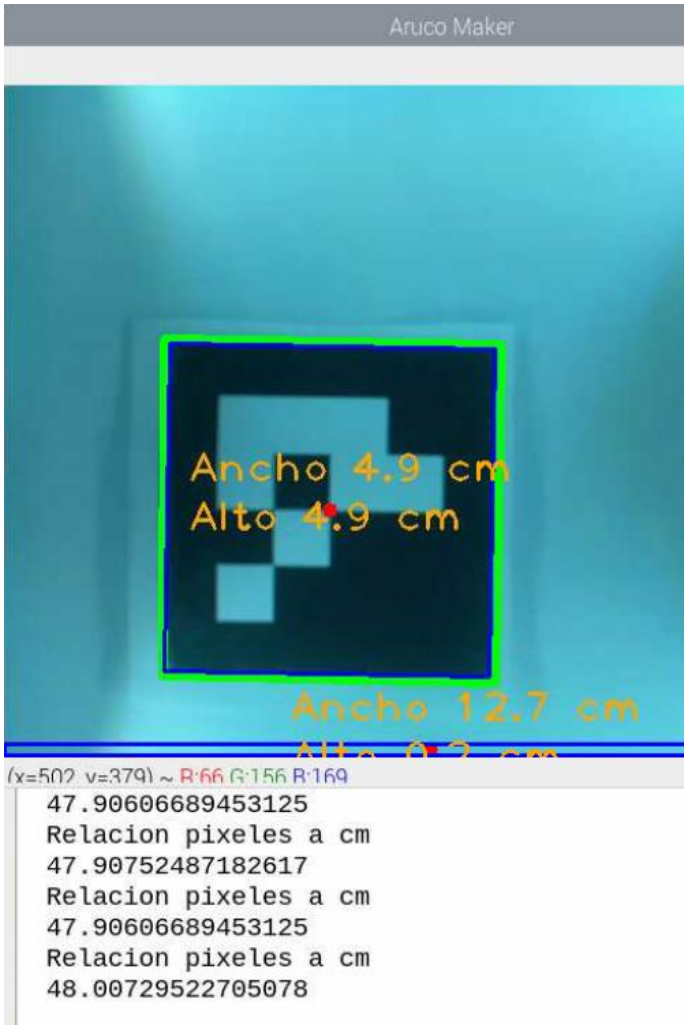


Figura 68: *Relación de pixeles a cm mediante aruco*

Fuente: Investigadores

Por lo tanto, en la figura superior se puede observar que la cámara está midiendo objetos, en este caso el marcador aruco cuyo tamaño real es 5x5 cm, el margen de error es de solo 0,1 cm. Además, se ha obtenido la relación de pixeles a cm. Este valor puede ser usado como constante para calcular el tamaño real de todos los alevines durante proceso de conteo.

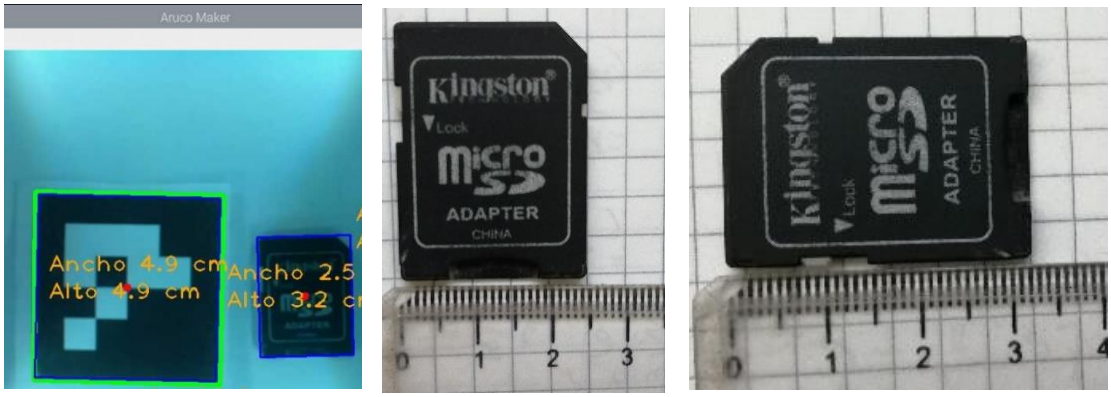


Figura 69: *De izquierda a derecha: Se captura el tamaño de un adaptador SD. Se mide el tamaño mediante una regla del ancho de la tarjeta. Se mide el alto de la tarjeta*

Fuente: Investigadores

Como se puede apreciar en la figura superior el tamaño medido con la cámara en comparación con las medidas de una regla son prácticamente las mismas.

3. Relación entre longitud y peso del alevín.

Para encontrar esta relación, la Organización de las Naciones Unidas para la Alimentación y la Agricultura pone a disposición una tabla.

Tabla 48: *Relación entre la longitud y peso de la trucha arcoíris*

Longitud (cm)	Peso (gramos)
2,0	0,11
2,5	0,18
3,0	0,40
3,5	0,61
4,0	0,86
4,5	1,15
5,0	1,49
5,5	2,18
6,0	2,87
6,5	3,72
7,0	4,60
7,5	5,60
8,0	6,70
8,5	7,90
9,0	9,20
9,5	10,53
10,0	12,00
10,5	14,00

Fuente: [6, p. 38]

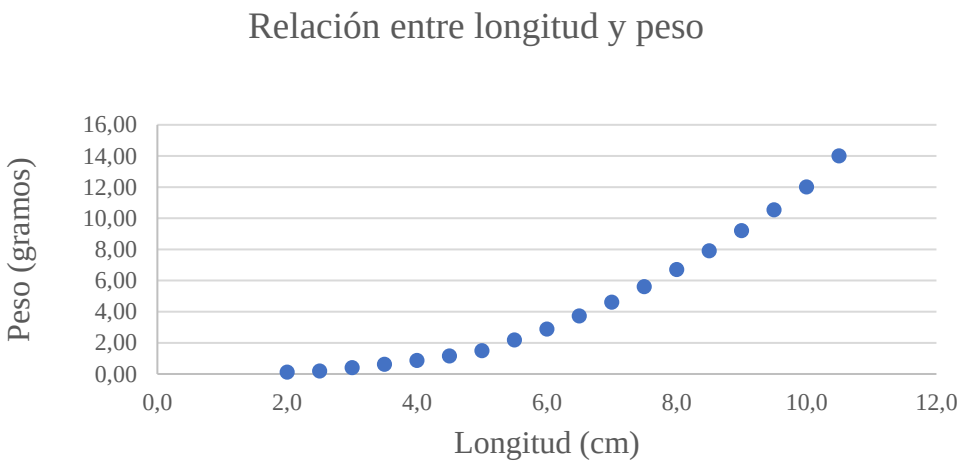


Figura 70: *Diagrama de dispersión de la relación entre peso y longitud*

Fuente: Investigadores

Por lo tanto, para estimar el peso del alevín en función de su longitud hay que modelar una ecuación que relacione ambas variables cuantitativas (peso y longitud). Para ello, se puede aplicar métodos estadísticos como la regresión lineal, cuadrática o cúbica.

Mediante las siguientes gráficas se puede obtener el valor de los diferentes métodos estadísticos.

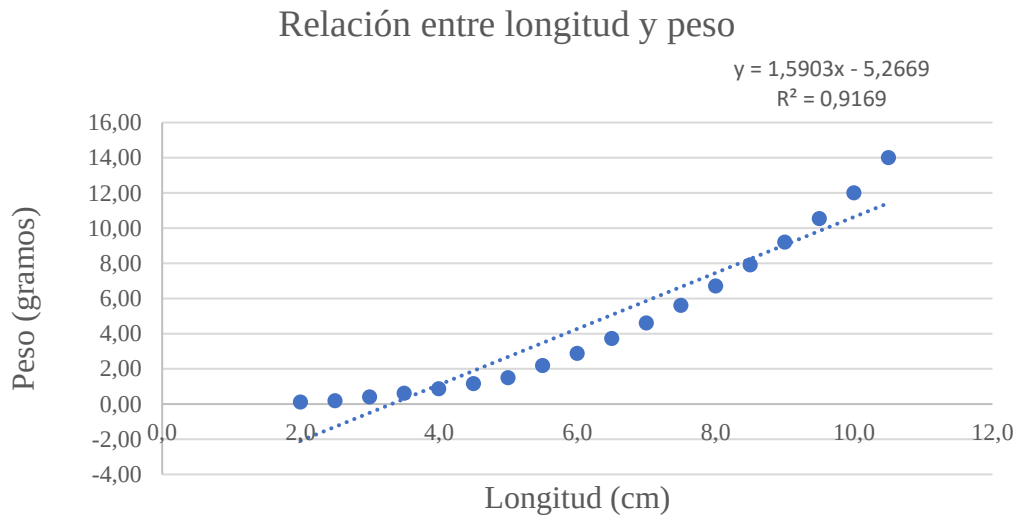


Figura 71: *Regresión lineal*

Fuente: Investigadores

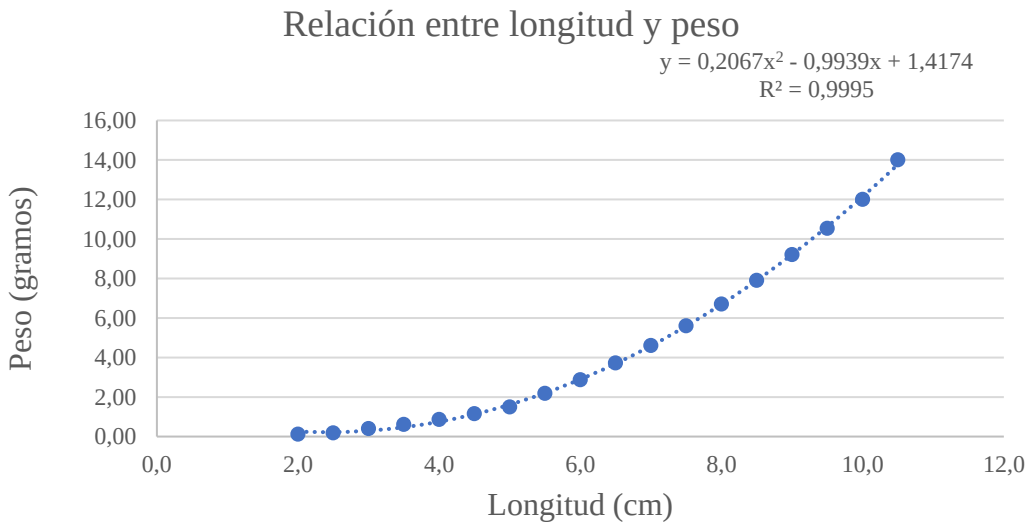


Figura 72: *Regresión cuadrática*

Fuente: Investigadores

Como se puede observar la curva que mejor se adapta a los datos es la regresión cuadrática. Además, hay que recordar que el coeficiente de determinación (R^2) entre más cerca este de la unidad mejor será el ajuste del modelo.

A continuación, se define mediante código como se obtiene los datos la talla de los alevines. Para ello hay que descomponer el diccionario "objetos". La posición donde se almacenan los centroides está en $centroid[0]$ y donde se guardan las longitudes de los alevines está en $centroid[1]$. El valor de las longitudes se divide entre la relación de pixeles-cm encontrada previamente y se obtendrá el tamaño de cada alevín que ha sido contado. La variable "t" será el valor independiente dentro del modelo estadístico para obtener el peso de cada alevín.

Finalmente, la variable "masa" irá sumando el peso de cada alevín para obtener la biomasa total.

```
t = np.double(centroid[1]/47.90)
regresion = 0.2067*math.pow(t,2) - 0.9939*t + 1.4174
masa += regresion
```

Publicación de datos en FIREBASE

Para la publicación de datos en FIREBASE se define la dirección del nodo padre y se actualiza la información de los nodos hijos mediante "ref.update".

```
ref = db.reference('Datos RealTime-Manual')
ref.update({
            "cantidad":"{}".format(round(calculo.raciones,2)),
            "hora":"{}".format(calculo.horas),
            "intervaloHora":"{}".format(calculo.intervaloHora),
            "intervaloMinutos":"{}".format(calculo.intervaloMinutos),
            "minutos":"{}".format(calculo.minutoss),
            "repeticiones":"{}".format(calculo.repeticiones)
                    })
```

Etapas implementadas durante la visión por computadora

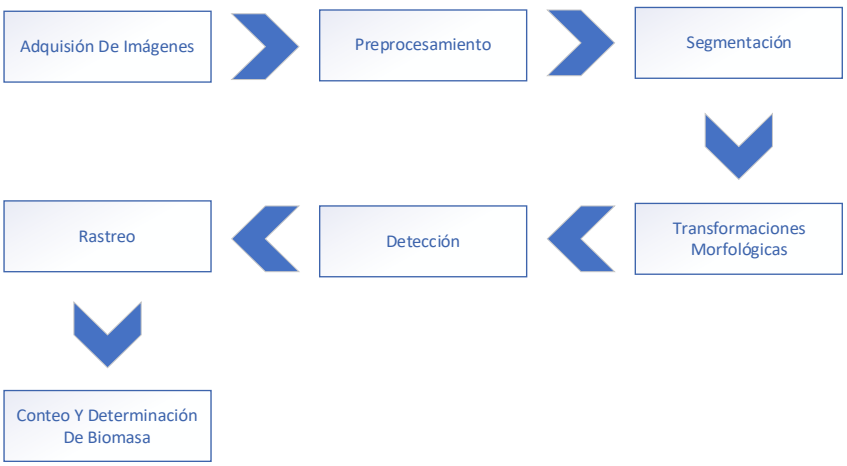


Figura 73: *Etapas implementadas en el desarrollo del contador*

Fuente: Investigadores

Adquisición de la imagen

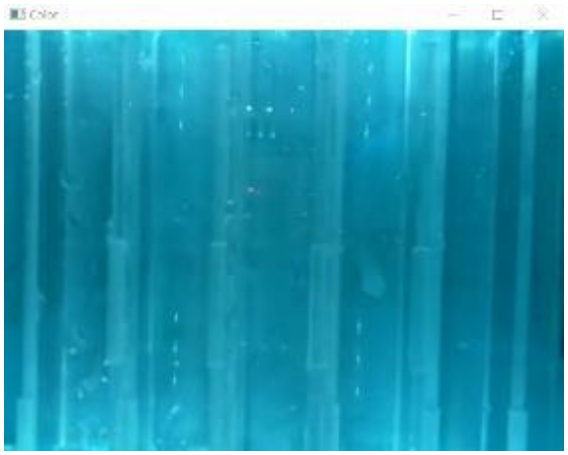


Figura 74: *Imagen tomada desde cámara*

Fuente: Investigadores

Para empezar con el proceso de conteo se ha empezado capturando video en tiempo real desde la cámara.

```
cap = VideoStream(usePiCamera=1 > 0).start()
```

Preprocesamiento

Esta etapa consiste en eliminar defectos o ruidos introducidos por la cámara mediante filtros. Además, resaltar ciertos detalles para mejorar el proceso de conteo.

El ambiente donde se ha implementado el contador, tiene la cámara estática en todo momento. La iluminación es constante. Por lo tanto, se trata de un escenario controlado donde no hay cambios bruscos de luz y movimientos aleatorios de la cámara. Por ello, el uso de filtros no fue necesario.

En esta etapa el único cambio que hizo es, transformar el video a escala de grises.

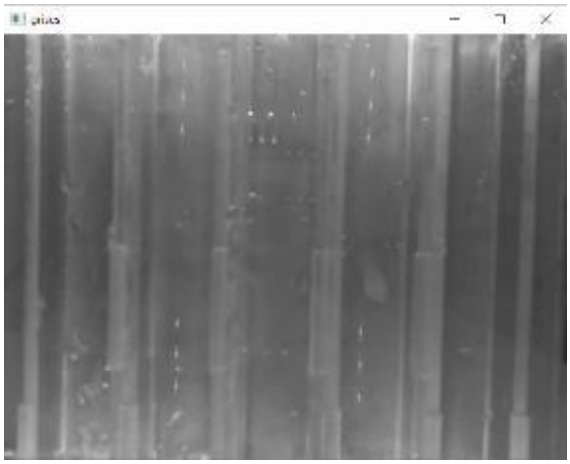


Figura 75: *Imagen en escala de grises*

Fuente: Investigadores

```
grises = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Segmentación

Para la etapa de segmentación se tienen diferentes técnicas. Pero, para esta aplicación se ha implementado el método de sustracción de fondo, que viene integrado de forma predeterminada en su algoritmo la segmentación por umbralización.

Figura 76: *Proceso de sustracción de fondo*

Fuente: Investigadores

Para encontrar objetos en primer plano, el método de sustracción compara el fondo (cuadros donde no hay objetos) con los fotogramas posteriores donde hay nuevos objetos. Esta diferencia ayuda a aislar el objeto de interés.

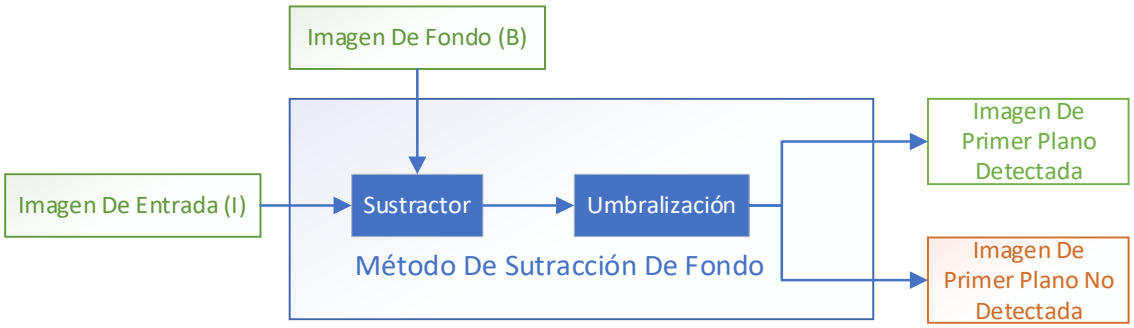


Figura 77: *Algoritmo de sustracción de fondo*

Fuente: [50, pp. 245-262]

Como se puede observar en el diagrama de bloques superior, se tiene una imagen de entrada (I) y también el fondo (B) que permanecerá constante durante todo el tiempo. La imagen de entrada introducirá el objeto en movimiento (en este caso el alevín, que tendrá un color diferente con respecto al fondo, para poder aislarlo). La diferencia entre (I) y (B) pasará por el valor de umbralización (el valor del umbral es definido mediante los primeros fotogramas del video). Si la diferencia de fotogramas es mayor al valor de umbralización, se detectará el objeto en primer plano caso contrario será tomado como parte del fondo [50, pp. 245-262].

Transformaciones morfológicas

Las transformaciones permiten cambiar la forma de los objetos, con el fin de mejorar la geometría o separar unos objetos de otros. Se la aplica por lo general en las imágenes binarizadas (blanco y negro).

En el proceso anterior (Segmentación) se ha obtenido una imagen binarizada después del método de sustracción de fondo. Pero, al separar el objeto de interés, también ha sustraído las gotas de agua que, dentro del proceso de conteo, dicha información no es relevante.

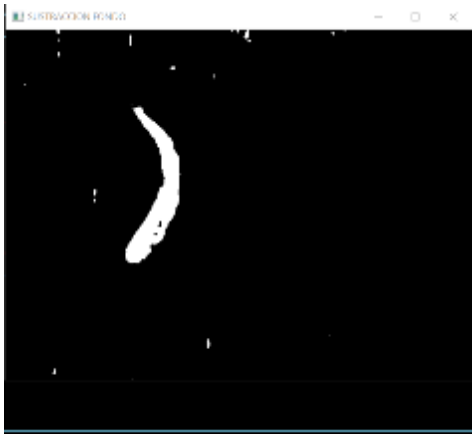


Figura 78: *Gotas de agua junto al objeto de interés*

Fuente: Investigadores

Para eliminar las gotas se ha implementado la transformación de erosión que suprime las partículas que se encuentran distantes del objeto de interés. Además, se ha utilizado como kernel una matriz cuadrada de 3x3.

Figura 79: *Transformación morfológica erosión*

Fuente: Investigadores

Con la transformación de erosión se han eliminado las gotas de agua de la imagen binarizada, pero el objeto de interés no tiene una estructura uniforme.

Por lo tanto, para acondicionar la estructura del objeto, se ha implementado la transformación de dilatación que, en lugar de eliminar manchas aisladas, expande o dilata las regiones que se encuentran casi juntas, formando una figura sólida.

Figura 80: *Transformación morfológica dilatación*

Fuente: Investigadores

Como se puede apreciar en la figura superior se ha obtenido un objeto con una estructura más sólida.

Al proceso de aplicar erosión y después dilatación también se lo denomina apertura.

Detección

Para este proceso primero se debe encontrar los contornos de los objetos. Para ello OpenCV entrega herramientas para definirlas.

```
cnts = cv2.findContours(dilatacion, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]
```

Con los contornos encontrados y mediante un ciclo for se ha hace un recorrido por este arreglo. Al mismo tiempo, se calcula el área de estas figuras encontradas. Si el área del objeto es mayor a 1000 pixeles, entonces se almacenan las coordenadas y se grafica el cuadro delimitador alrededor del objeto.

```
for i in cnts:
        if cv2.contourArea(i) > 1000:
            x,y,w,h = cv2.boundingRect(i)
            endX = x+w
            endY = y+h
            cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0),2)
```

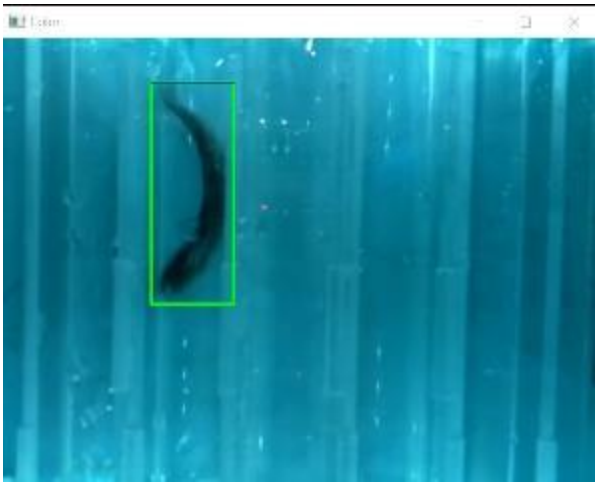


Figura 81: *Alevín detectado*

Fuente: Investigadores

Rastreo, Conteo y Determinación de la biomasa

Estas etapas han sido detalladas en la sección del desarrollo de la programación.

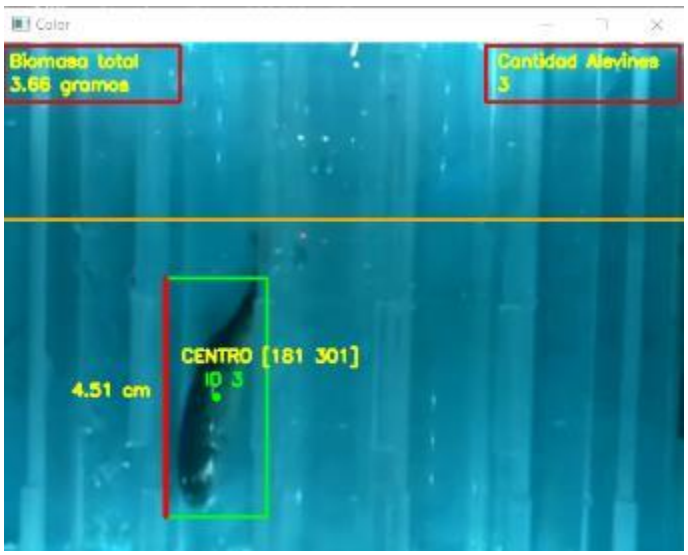


Figura 82: *Conteo y biomasa*

Fuente: Investigadores

En la figura superior se puede apreciar el conteo y determinación de la biomasa. Cabe aclarar que en la etapa final de implementación del sistema estas variables solo se visualizarán mediante un LCD. Lo que se ha mostrado en la figura superior es para evidenciar que el sistema realiza las actividades planteadas.

Figura 83: *Implementación final para visualización en LCD*

Fuente: Investigadores

Desarrollo del dispensador y contador mediante software

Dispensador de alimento para alevines de truchas

SolidWorks es un software enfocado al diseño asistido por computador, CAD en 3D y en 2D, ofreciendo soluciones de desarrollo a elementos de sistemas mecánicos, así como también realizar diferentes estudios y casos que pueden ocurrir productos de la manufactura [51].

De esta manera se aprovechó las ventajas que brinda SolidWorks para realizar el diseño y emplear un análisis de tipo estructural por medio de un estudio estático.

Para garantizar la eficiencia correcta de la estructura, tanto del material como en el dimensionamiento del mismo, vamos a tener en cuenta el factor de seguridad.

El factor de seguridad es el rango de seguridad necesario para una estructura, las cuales dependen de acuerdo a códigos y leyes de diseños.

El software SolidWorks es una herramienta muy empleada para diseños, análisis y estudios mecánicos. Es por ello que se ha optado utilizar dicha herramienta para el desarrollo del proyecto con respecto a la estructura física, mediante el modelado 3D, permitiendo realizar un análisis de estudio estático, con el fin de hallar uno de los parámetros importantes como lo es el factor de seguridad.

Se emplea un análisis de estudio estático al proyecto propuesto, para ello se tomó la estructura base que es el soporte principal en donde estará sometido a fuerzas, cargas y vibraciones por los elementos acoplados y de igual manera por su respectiva función.

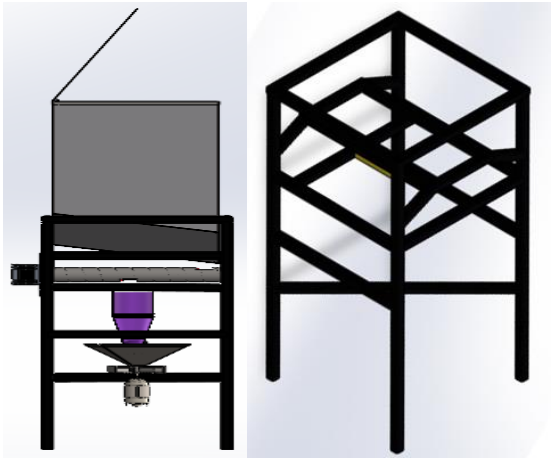


Figura 84: *Diseño del dispensador en el software SolidWorks*

Fuente: Investigadores

En donde, se aplica el siguiente procedimiento para dicho análisis:

a) Asignar el material a la estructura (Acero AISI 1020)

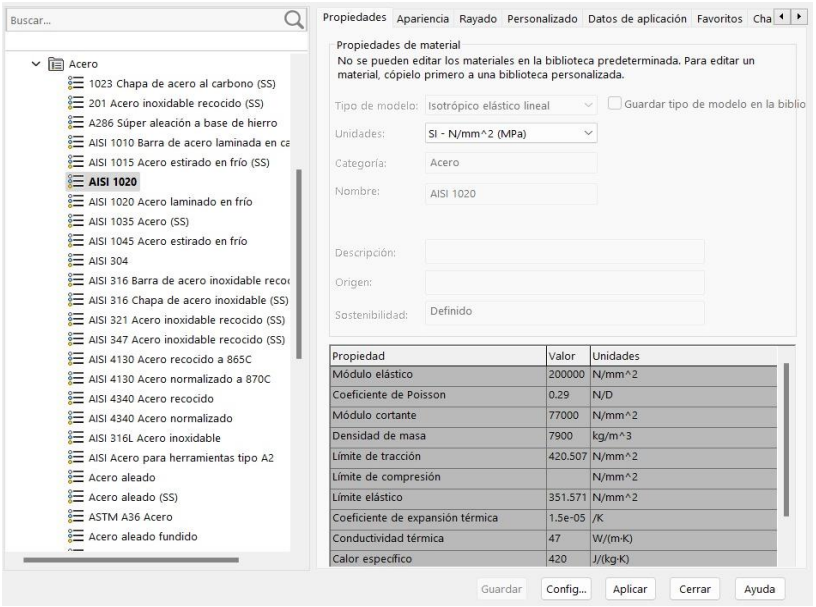


Figura 85: *Tabla de materiales*

Fuente: Investigadores

b) Aplicar sujeciones, fuerzas y cargas externas de manera distribuida alrededor de la estructura teniendo en cuenta la masa de alimentos de 25.79

kg más la gravedad (9.81 m/s^2) con el fin de tener unidades de fuerza, misma que será aplicada en la estructura.

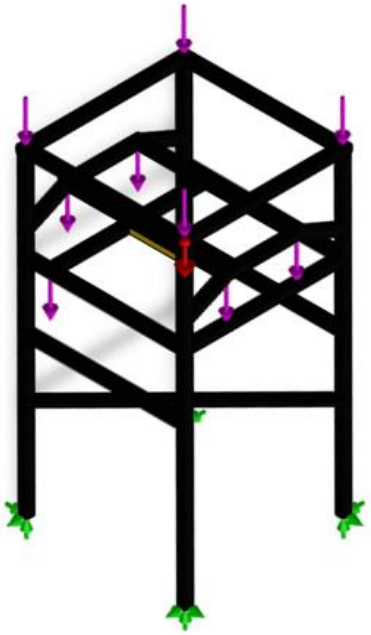


Figura 86: *Estructura aplicada sujeciones, fuerzas y cargas*

Fuente: Investigadores

c) Ejecutar el estudio para poder visualizar el comportamiento de las cargas aplicadas a la estructura, en donde se puede ver claramente distorsiones del material debido a las cargas aplicadas

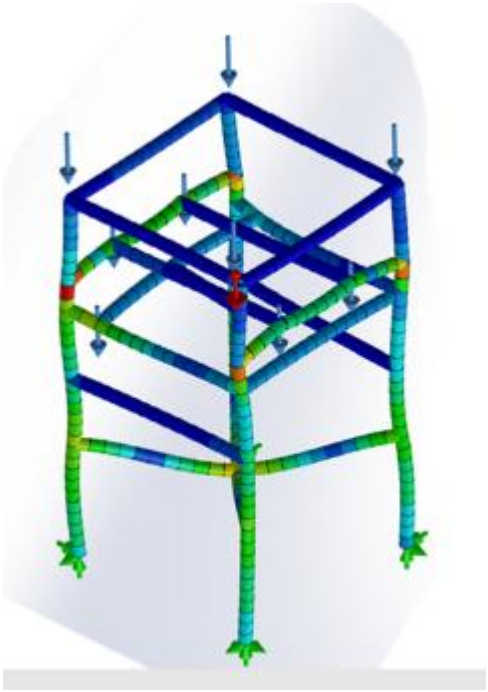


Figura 87: *Análisis de la estructura*

Fuente: Investigadores

d) SolidWorks permite ver y analizar el comportamiento de la estructura metálica por medio de un estudio de factor de seguridad, el cual mediante normativas de diseño mecánico estructural se menciona y recomienda estar dentro del rango de 1.5 a 2.5

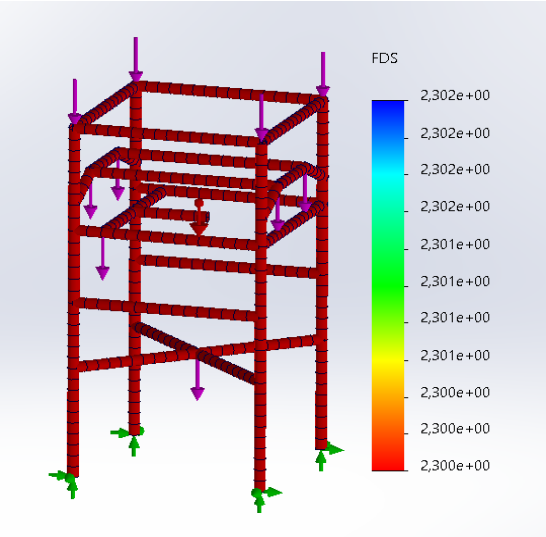


Figura 88: *Estudio de factor de seguridad*

Fuente: Investigadores

Como resultado del análisis de estudio estático se obtiene un valor de factor de seguridad de 2.3, valor que se encuentra dentro del rango recomendado para estructuras metálicas.

Concluyendo que el dimensionamiento y la selección del material son los adecuados para cumplir su función de sujetar los elementos móviles y fijos del alimentador de peces.

Figura 89: *Estructura ensamblada*

Fuente: Investigadores

Contador de alevines de truchas

Del mismo modo, se ha asegurado que el diseño del contador de alevines sea el adecuado para trabajar bajo las condiciones a la cual será expuesta.

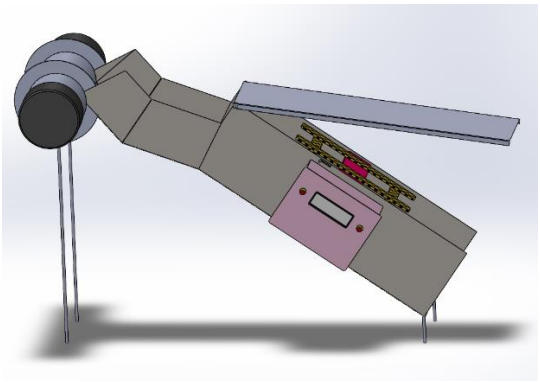


Figura 90: *Diseño del contador en el software SolidWorks*

Fuente: Investigadores

Con la ayuda del software SolidWorks se dimensiono la estructura metálica para el presente proyecto, de tal manera que las dimensiones sean las adecuadas y de igual manera la selección del material. Material usado: acero inoxidable AISI 304.

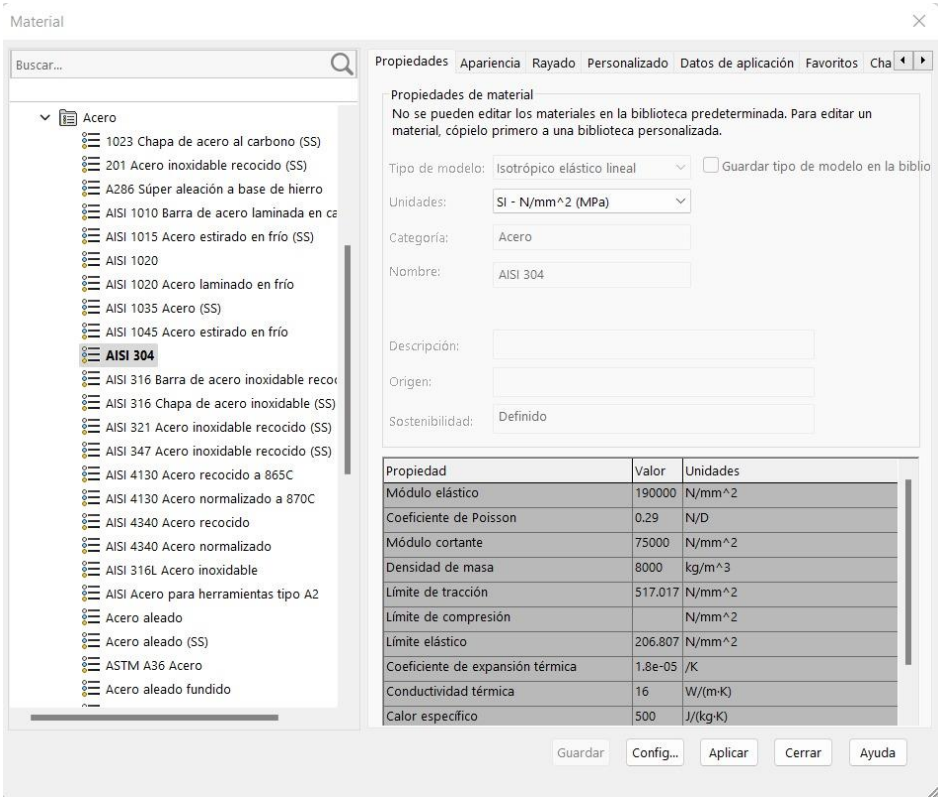


Figura 91: *Tabla de materiales*

Fuente: Investigadores

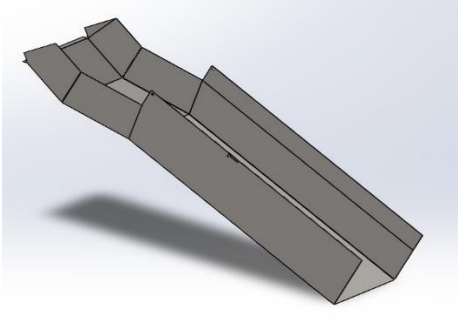


Figura 92: *Estructura metálica*

Fuente: Investigadores

a) Se procede a colocar cargas de sujeción y cargas externas teniendo en cuenta la masa de los alevines que en este caso es de 1 gramo promedio por unidad, a más de eso hay que considerar que son 5 canales con capacidad de circular 2 litros de agua. Parámetros que hay que considerar para ingresar en el software:

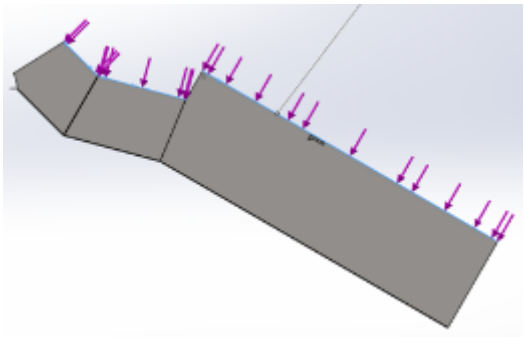


Figura 93: *Estructura aplicada sujeciones, fuerzas y cargas*

Fuente: Investigadores

b) Ejecutar el estudio para poder visualizar el comportamiento de las cargas aplicadas a la estructura, en donde se puede ver claramente distorsiones del material debido a las cargas aplicadas

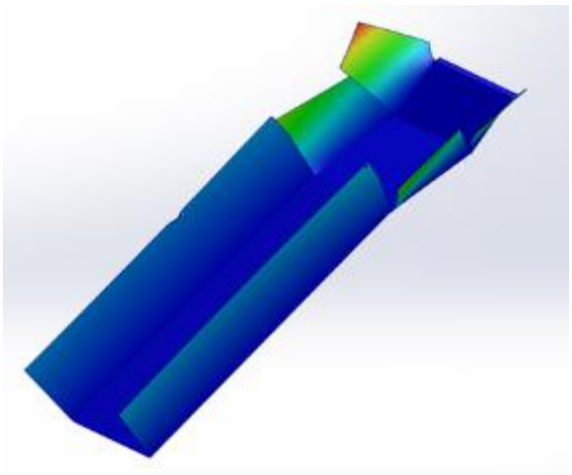


Figura 94: *Análisis de la estructura*

Fuente: Investigadores

c) SolidWorks permite ver y analizar el comportamiento de la estructura metálica por medio de un estudio de factor de seguridad, el cual mediante normativas de diseño mecánico estructural menciona y recomienda estar dentro del rango de 1.5 a 2.5

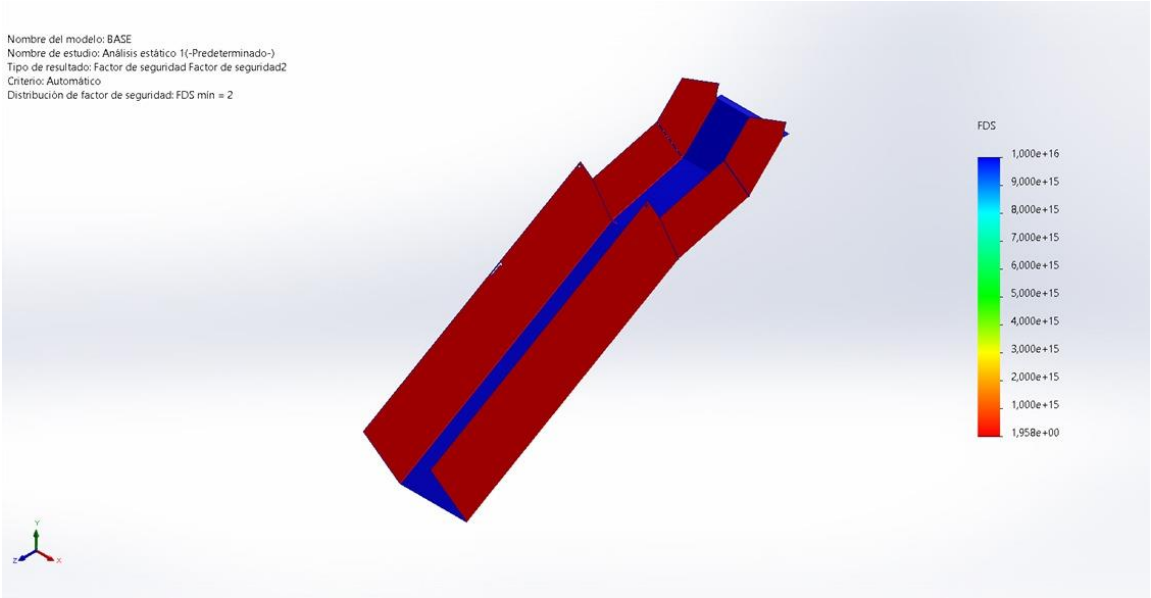


Figura 95: *Estudio de factor de seguridad*

Fuente: Investigadores

Como conclusión del análisis del estudio estático para hallar el factor de seguridad de la estructura del contador de alevines, se puede decir que se encuentra dentro del rango recomendado ya que presenta un factor de seguridad de 2, valor que garantiza un adecuado funcionamiento para la actividad que sea designada.

Figura 96: *Estructura ensamblada*

Fuente: Investigadores

Desarrollo de la interfaz Web

Selección del framework para el desarrollo de la página web

En la tabla 49 se puede visualizar el cuadro comparativo entre tres framewoks que son utilizadas para la creación de Páginas Web.

Tabla 49: *Cuadro comparativo entre React.js, Blazor y Angular*

	Frameworks		
Características	Javascript (React.js).	C#, Razor Pages (Blazor).	TypeScript (AngularJS).
Velocidad de desarrollo	Al utilizar componentes al momento de crear una página web, hace que sea super rápido.	Al utilizar un solo lenguaje de programación, ayuda a que una sola persona pueda realizar un proyecto.	Utiliza componentes por lo que la velocidad de desarrollo en esta plataforma es rápida, pero no tanto como react.js.
Compatibilidad de navegadores	Es compatible con todos navegadores incluyendo a Internet Explorer.	Safari, Chrome, Firefox, Edge.	Chrome, Firefox, Edge, Safari.
Manejar este lenguaje, en un futuro ¿Ayudara a entender otro tipo de lenguaje?	Al trabajar con la plataforma de React.js, facilitara al desarrollador a crear y entender la programación de React-native.	En la actualidad el uso del lenguaje de JavaScript es muy popular, por lo que aprender este lenguaje ayudara en un futuro a entender otros tipos de lenguajes. Blazor al utilizar el lenguaje C++ se limita a muchas cosas que puede ofrecer el lenguaje JavaScript.	Angular al eliminar la versión de angularJS limita el aprendizaje a otro tipo de lenguaje.
Comunidad de desarrollo Grande	El número de contribuidores a react.js es de 1.8k.	Según campusMVP.es Blazor trabaja bajo la licencia Apache 2.0, por lo que es factible que otros desarrolladores puedan colaborar con sus ideas por medio de pull requests.	El número de contribuidores Angular es de 798.
Soporte	Es soportado por Facebook.	Es soportado por Microsoft.	Es soportado por Google.
Flexibilidad	Es super flexible.	Requiere un conocimiento más profundo por lo que hace que no sea flexible, sin olvidar que ayuda a crear SPAs.	Es un poco flexible debido a que es un framework robusto.

Fuente: Investigadores

Justificación

Con ayuda de la tabla comparativa se puede concluir que la plataforma React.js es el indicado para la creación de la página web. Debido a que presenta un número elevado de participantes dentro de esta comunidad, ayudando a solventar cualquier duda y subir cualquier librería que aporte a los desarrolladores al momento de crear un proyecto.

- **Diagrama de flujo de la comunicación de React.js y Firebase**

Antes de empezar con la programación es muy importante tener claro la comunicación entre la interfaz Web y Firebase. En la figura 92, se puede visualizar la comunicación entre ellos.

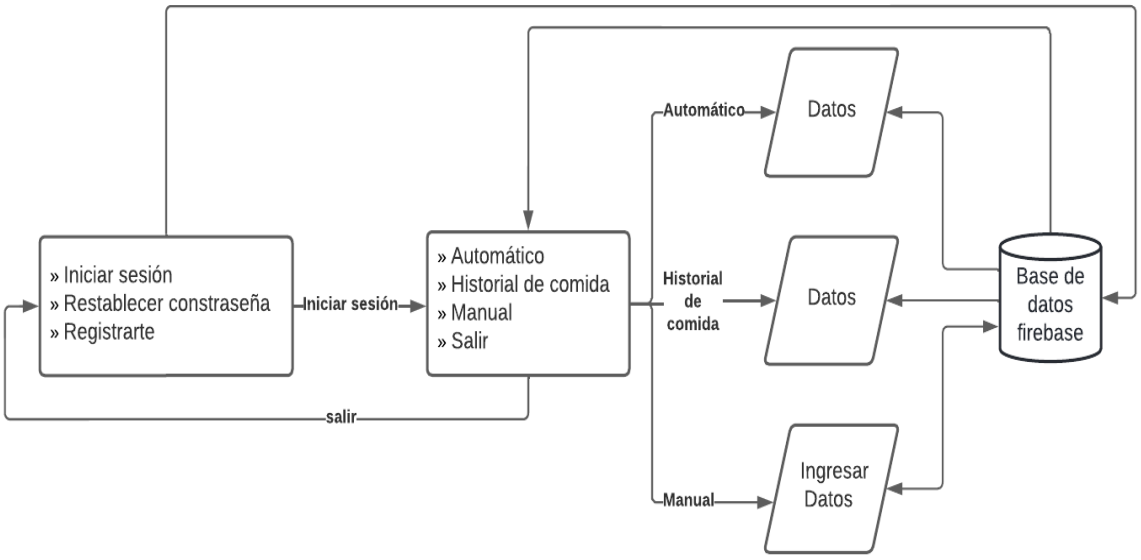


Figura 97: *Diagrama de flujo de la comunicación de React.js y Firebase*

Fuente: Investigadores

- **Diagrama de flujo del sistema de autenticación**

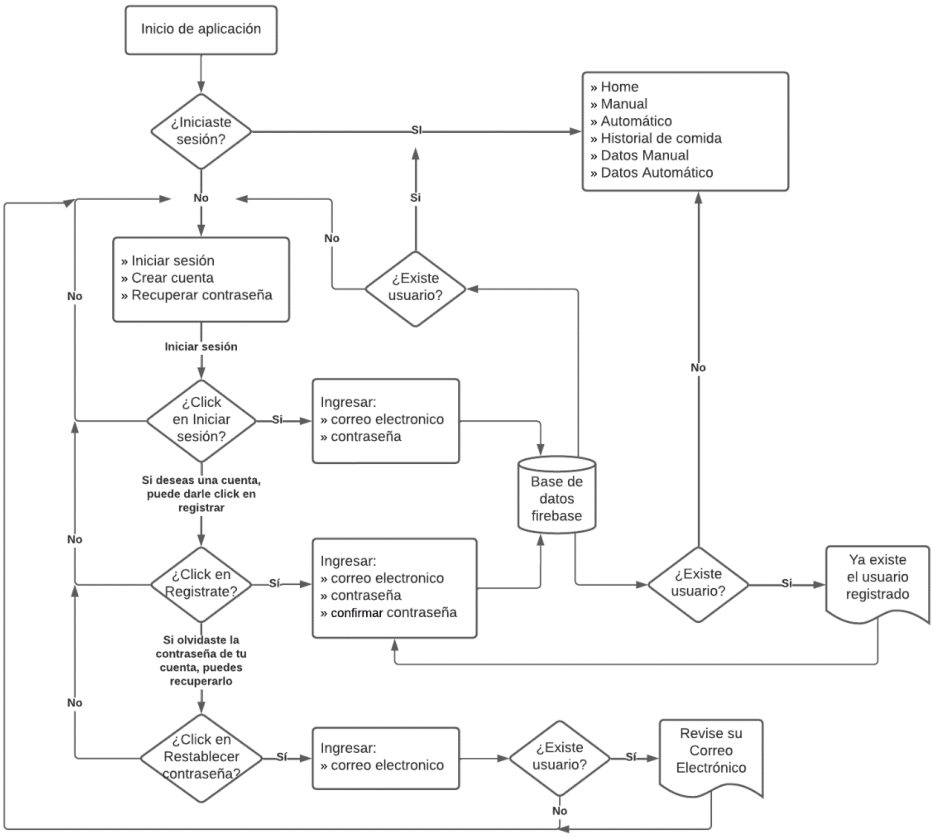


Figura 98: *Diagrama de flujo del funcionamiento de la interfaz Web*

Fuente: Investigadores

La figura 93 representa el funcionamiento de la interfaz web mediante un diagrama de flujo, ayudando al lector a entender de manera gráfica el funcionamiento del mismo.

Programación de la página web

Es muy importante tener en cuenta que las librerías de react.js y react-native son diferentes. Por lo que a continuación, en la tabla 50 se describirá de manera explícita y ordenada cada una de las librerías utilizadas en react.js

Tabla 50: *Librerías instaladas en el desarrollo de la interfaz Web*

Nombre de librería	Línea de código	Descripción
StrictMode	N/A	La librería "StrictMode" está encargada en verificar si alguna librería instalada en el proyecto se encuentra obsoleta, se tiene que tener en cuenta que el uso de este componente no será visible como mensaje de error, sino que es un punto clave en modo de desarrollo.
react-router-dom	npm i react-router-dom	Esta librería se encarga de la navegación entre pantallas, react-router-dom está compuesto por switch, BrowserRouter, Route, Redirect que permiten realizar redirecciones.

firebase	npm install --save firebase	La siguiente librería se encarga de instalar todos los paquetes necesarios para la comunicación entre React.js y la plataforma de Firebase
bootstrap 5.0.0-beta1	Npm i bootstrap@5.0.0-beta1	Bootstrap es una librería de estilos que permite al desarrollador tener una presentación más adecuada, también permite realizar ventanas emergentes entre otros aspectos importantes que se utilizó en la construcción del proyecto.

Fuente: Investigadores

- **Creación de las ventanas Iniciar sesión, Crear cuenta, Recuperar contraseña**

En la figura 99, se observa la ventana Iniciar sesión, el propósito principal de esta ventana es ingresar a la pantalla home por medio de la autenticación. Además, permite navegar a la ventana "Crear cuenta nueva" y a la ventana "¿Olvidaste tu contraseña?".

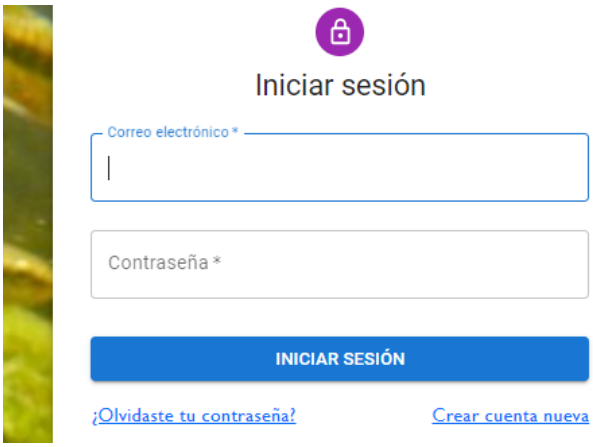


Figura 99: *Ventana iniciar sesión*

Fuente: Investigadores

El bloque de programación para el botón 'Iniciar sesión' de la figura 99, sirve para el envío de datos a la plataforma de firebase desde React.js. Se puede observar en la figura 100 que handleSubmit se encarga de verificar si existe el usuario ingresado, en caso de existir permitirá el paso a home y en caso de no existir simplemente no permitirá el paso a home hasta que ingrese un usuario existente. Las líneas de código que permiten el logueo se puede observar en la figura 101.

```
const handleSubmit = async (e) => {
  e.preventDefault();
  setLoading(true);
  try {
    await login(email, password);
    setLoading(false);
    history.push('/');
  } catch (error) {
    setLoading(false);
    setError('Introduce tu correo electrónico y tu contraseña');
    setTimeout(() => setError(''), 1500);
  }
}
```

Figura 100: *Funcionamiento del Botón 'Iniciar sesión'*

Fuente: Investigadores

```
const login = (email, password) => {
  return auth.signInWithEmailAndPassword(email, password);
}
```

Figura 101: *Bloque de programación de login*

Fuente: Investigadores

En la figura 102, se puede observar la ventana de registro encargada de crear un nuevo usuario, en caso de no tenerlo.

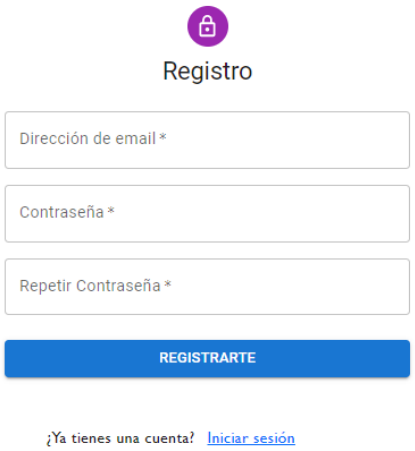


Figura 102: *Ventana de registro*

Fuente: Investigadores

En la figura 103, se puede visualizar las líneas de código que permiten la creación de un nuevo usuario. Para que la aplicación móvil se haga más interactiva se ha programado con mensajes a consola, estos mensajes servirán para que el desarrollador pueda poner condiciones, por ejemplo: si el usuario existe se enviará por consola que "ingrese otros datos". En la figura 104, se puede ver el funcionamiento del botón 'Registrarte'.

```
const signup = (email, password) => {
  return auth.createUserWithEmailAndPassword(email, password);
}
```

Figura 103: *Código para la comunicación entre react-native y Authentication*

Fuente: Investigadores

```
const handleSubmit = async (e) => {
 e.preventDefault();
 setLoading(true);
 if(password !== confirmPassword){
   setError('Las contraseñas no coinciden');
   setTimeout(() => setError(''), 1500);
 }else{
   try{
     await signup(email, password);
     history.push('/');
   }catch(error){
     setError('Introduce una dirección de correo electrónico');
     setTimeout(() => setError(''), 1500);
   }
      }
     setLoading(false);
```

Figura 104: *figura y Funcionamiento del Botón 'Registrarte'*

Fuente: Investigadores

La ventana encargada en la restauración de la contraseña en caso de que el usuario lo haya olvidado, se puede observar en la figura 105. Una vez que se envía el dato ingresado este verifica si el dato ingresado es el correcto, si es correcto dirige al usuario a la ventana de login y muestra un mensaje de advertencia, y en caso de ser incorrecto simplemente dirige a la ventana login.

Figura 105: *Ventana de Restablecer tu contraseña*

Fuente: Investigadores

En la figura 106, se puede visualizar las líneas de código que permiten el restablecimiento de la contraseña en caso de que el usuario se haya olvidado. Para que la aplicación móvil se haga más interactiva se ha programado con mensajes a consola, estos mensajes servirán para que el desarrollador pueda poner condiciones, por ejemplo: si el usuario no existe, se enviará por consola que "Falló al restaurar tu

contraseña”. En la figura 107, se puede ver el funcionamiento del botón ‘Restablecer contraseña’.

```
function resetPassword1(email) {
  return auth.sendPasswordResetEmail(email)
}
```

Figura 106: *Código para la comunicación entre react-native y Authentication*

Fuente: Investigadores

```
async function handleSubmit(e) {
  e.preventDefault();
  try {
    setMessage('Verifica tu bandeja de entrada y sigue las instrucciones')
    setMessage('')
    setError('')
    setLoading(true)
    history.push('/');
    await resetPassword1(email2Ref.current.value)
    setMessage('Verifica tu bandeja de entrada y sigue las instrucciones')
  }catch{
    setError('Falló al restaurar tu contraseña')
      }
    setLoading(false)
```

Figura 107: *Funcionamiento del Botón ‘Restablecer contraseña’*

Fuente: Investigadores

- **Creación de la ventana Home**

La ventana home presenta un menú de opciones, cada una de ellas le direcciona al usuario a una nueva ventana, tal como se muestra en la figura 108. Uno de los aspectos importantes en esta ventana, es la lectura y visualización de la tolva

Figura 108: *Ventana Home*

Fuente: Investigadores

El bloque de programación de la figura 109, está encargada en la comunicación entre react.js y Firebase Real-time. Gracias al uso de useEffect se puede leer los datos en tiempo real sin la necesidad de actualizar la página.

```
const [contactObjects, setContactObjects] = useState('')
useEffect(() => {
    firebaseDb.child('Tolva/Nivel/Value')
    .on('value', snapshot => {
    setContactObjects(snapshot.val())
                                    }
      )
                                  }
     ,[])
```

Figura 109: *Código para la lectura del dato Tolva*

Fuente: Investigadores

GaugeChart es una librería que se encarga en presentar el nivel de la tolva de manera animada, tal como se muestra en la figura 108. El bloque de programación de la figura 110 indica el nivel de la tolva por medio de un GaugeChart, el mismo se encarga de indicar hasta el 100% que es el nivel de tolva llena.

```
<Grid item xs={12}>
  <GaugeChart id="gauge-chart2"
    nrOfLevels={20}
    percent={contactObjects/100}
    textColor="#000000"

  />
</Grid>
```

Figura 110: *Código para indicar el nivel de la tolva*

Fuente: Investigadores

- **Creación de la ventana Automático**

La ventana de la figura 111, se encarga en mostrar los datos que se encuentran en la plataforma de Firebase Real-Time, estos datos leídos pertenecen al “contador de alevines”. Además, en la figura 112, se puede observar el uso de “useEffect”, este bloque de programación ayuda a actualizar los datos sin la necesidad de refrescar la página.

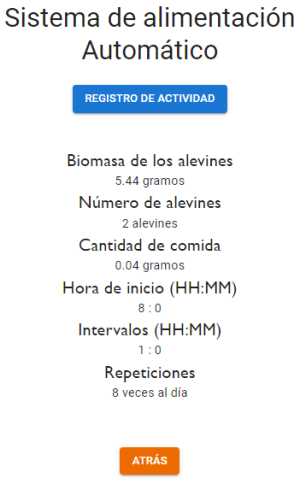


Figura 111: *Ventana Automático*

Fuente: Investigadores

```
const [contactObjects, setContactObjects] = useState({})
useEffect(() => {
    firebaseDb.child('Datos RealTime-Automatico')
    .on('value', snapshot => {
        if (snapshot.val() != null)
        setContactObjects({...snapshot.val()})
        else
        setContactObjects({})
        }
        )
            }
        ,[])
```

Figura 112: *Código para la lectura de datos*

Fuente: Investigadores

Para la visualización de los datos en la ventana automático, se hizo uso del siguiente bloque de programación como se muestra en la figura 113.

```
<Typography align='center' component="h4" variant="inherit">
  Biomasa de los alevines
</Typography>
<Typography align='center' component="h4" variant="subtitle1">
    {contactObjects[id].biomasa} gramos
</Typography>

<Typography align='center' component="h4" variant="inherit">
  Número de alevines
</Typography>
<Typography align='center' component="h4" variant="subtitle1">
    {contactObjects[id].peces} alevines
</Typography>

<Typography align='center' component="h4" variant="inherit">
    Cantidad de comida
</Typography>
<Typography align='center' component="h4" variant="subtitle1">
    {contactObjects[id].comida} gramos
</Typography>
```

Figura 113: *Código para la visualización de los datos en la ventana Automático*

Fuente: Investigadores

Creación de la ventana Manual

La ventana Manual de la figura 114, se encarga de la subida de los datos a Firebase Real-Time. El bloque de programación de la figura 115, se encarga en subir los datos que haya ingresado el usuario.

Figura 114: *Ventana Manual*

Fuente: Investigadores

```
const addOrEdit = obj => {
if (currentId == 'ss'){
}else{
    firebaseDb.child(`Datos RealTime-Manual/${currentId}`)
    .set(
      obj,
      err => {
       if (err)
         console.log(err)
       else
         setCurrentId('')
                }
       )
     }
                                }
```

Figura 115: *Comunicación entre react.js y Firebase Real-Time*

Fuente: Investigadores

Uno de los aspectos interesantes que tiene la ventana Manual, es la comunicación de React.js y Firebase utilizando datos booleanos. La figura 116, es la encargada en habilitar y deshabilitar un botón. En este caso cuando se envíe un dato booleano el botón Automático cambiará a ‘Habilitar’ o ‘Deshabilitar’, cuando el valor booleano sea “false” el botón va estar Habilitado y cuando sea “true” el botón se Deshabilitará, en la figura 117, se puede observar la lectura del dato booleano con ayuda de “useEffect”.

```
<Grid item xs={12} sm={6}>
  <Button fullWidth variant="contained"
          onClick={hAutomatico}
          disabled={boton} >Automático</Button>
</Grid>
```

Figura 116: *Botón utilizando la función disabled={}*

Fuente: Investigadores

```
const [boton, setBoton] = useState(false);
useEffect(() => {
  firebaseDb.child('boton/estado/booleano')
  .on('value', snapshot => {
  setBoton(snapshot.val())
                          }
    )
              }
    ,[])
```

Figura 117: *Comunicación entre react.js y Firebase Real-Time*

Fuente: Investigadores

- **Creación de las ventanas Historial de comida, Base de datos manual, Base de datos automático**

Las ventanas de las figuras 118, 119, 120 están encargadas en mostrar los datos que se encuentran almacenados en la plataforma de Firebase Real-Time, estas tres ventanas necesitan el uso de useEffect, la estructura general para leer los datos desde react.js a firebase se puede observar en la figura 122. Para mostrar estos datos en una tabla de manera ordenada se puede visualizar en la figura, a excepción de la ventana historial debido a que hace uso de la función map en la figura 121, se puede observar los datos en una tabla con la función map.

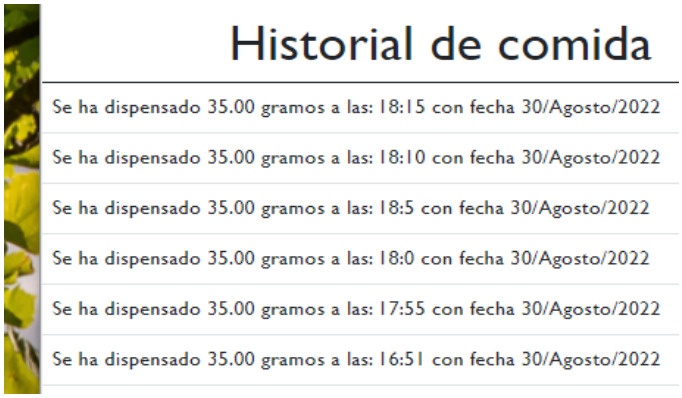
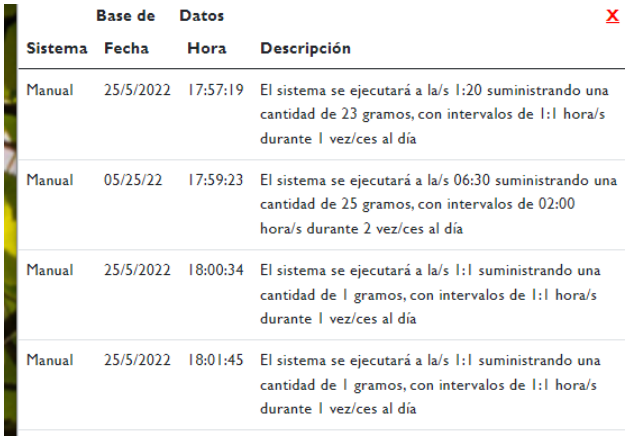


Figura 118: *Ventana de Base de Historial de comida*

Fuente: Investigadores



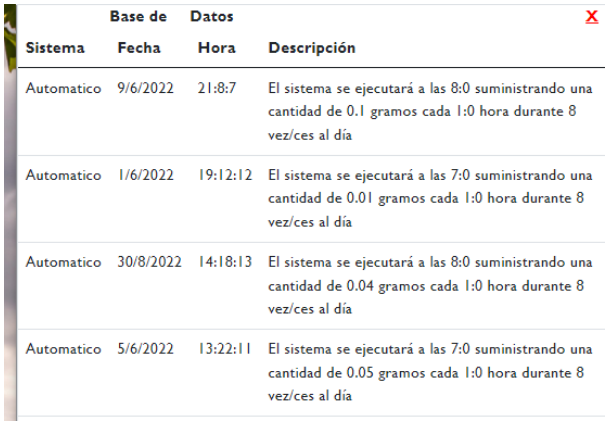
Base de Datos

X

Sistema	Fecha	Hora	Descripción
Manual	25/5/2022	17:57:19	El sistema se ejecutará a la/s 1:20 suministrando una cantidad de 23 gramos, con intervalos de 1:1 hora/s durante 1 vez/ces al día
Manual	05/25/22	17:59:23	El sistema se ejecutará a la/s 06:30 suministrando una cantidad de 25 gramos, con intervalos de 02:00 hora/s durante 2 vez/ces al día
Manual	25/5/2022	18:00:34	El sistema se ejecutará a la/s 1:1 suministrando una cantidad de 1 gramos, con intervalos de 1:1 hora/s durante 1 vez/ces al día
Manual	25/5/2022	18:01:45	El sistema se ejecutará a la/s 1:1 suministrando una cantidad de 1 gramos, con intervalos de 1:1 hora/s durante 1 vez/ces al día

Figura 119: *Ventana de Base de Datos Manual*

Fuente: Investigadores



Base de Datos

X

Sistema	Fecha	Hora	Descripción
Automatico	9/6/2022	21:8:7	El sistema se ejecutará a las 8:0 suministrando una cantidad de 0.1 gramos cada 1:0 hora durante 8 vez/ces al día
Automatico	1/6/2022	19:12:12	El sistema se ejecutará a las 7:0 suministrando una cantidad de 0.01 gramos cada 1:0 hora durante 8 vez/ces al día
Automatico	30/8/2022	14:18:13	El sistema se ejecutará a las 8:0 suministrando una cantidad de 0.04 gramos cada 1:0 hora durante 8 vez/ces al día
Automatico	5/6/2022	13:22:11	El sistema se ejecutará a las 7:0 suministrando una cantidad de 0.05 gramos cada 1:0 hora durante 8 vez/ces al día

Figura 120: *Ventana de Base de Datos Automático*

Fuente: Investigadores

```
<tbody>{
  Object.keys(contactObjects).map(id => {
  return <tr key={id}>
  <td>{contactObjects[id].usuario}</td>
  </tr>
                                        }
                                      )
}</tbody>
```

Figura 121: *Código encargado en poner los datos en una tabla haciendo uso de map*

Fuente: Investigadores

```
useEffect(() => {
    firebaseDb.child('Base de Datos-Manual').on('value', snapshot => {
    if (snapshot.val() != null)
    setContactObjects({...snapshot.val()})
                                                    }
                                                  )
                }
        ,[])
```

Figura 122: *Comunicación entre react.js y Firebase Real-Time*

Fuente: Investigadores

```
<tbody>{
  Object.keys(contactObjects).map(id => {
  return <tr key={id}>
  <td>{contactObjects[id].usuario}</td>
  <td>{contactObjects[id].fechaactual}</td>
  <td>{contactObjects[id].horaactual}</td>
  <td>{contactObjects[id].descripcion}</td>
  </tr>
                                    }
                                  )
}</tbody>
```

Figura 123: *Código encargado en poner los datos en una tabla*

Fuente: Investigadores

Desarrollo de la interfaz móvil

Selección de Lenguaje de programación

A continuación, se compararon algunos lenguajes de programación más utilizadas en la creación de aplicaciones móviles. En la tabla 51, se puede observar dicha comparación.

Tabla 51: *Tabla comparativa entre React-native, App inventor y Kivy*

Lenguajes de Programación			
Características	JavaScript, Java, Python, C++, Objective-C (React-native)	Java, kawa, Scheme (App Inventor)	Python, Cython (kivy)
Velocidad de desarrollo	Al tener conocimiento de React.js, resulta más rápido y rentable utilizar React-native	Es rápido pero las aplicaciones son de nivel medio	Para la creación de aplicaciones avanzadas toma mucho tiempo a los desarrolladores
Reusabilidad de Código	El lenguaje de programación es compatible para Android y iOS, además desarrolla aplicaciones para Android TV, tvOS, Windows, macOS, UWP, y Web.	El mismo programa es compatible con Android y iOS	Kivy es una librería encargada en la creación de aplicaciones móviles que pueden ser ejecutadas en Android y iOS
Bidireccional	Tiene una característica llamada 'Bridge' el cual permite comunicación bidireccional entre lenguaje JavaScript y módulos nativos	El lenguaje de programación está basado solo en Java, Kawa y Scheme está relacionado con Java.	Trabaja solamente con el lenguaje de Python.
Comunidad de desarrollo Grande	Se actualiza constantemente debido a que están bajo una licencia Open source (cualquier persona pueda ver o modificar el código fuente)	El lenguaje de programación de App inventor es de tipo estático, por tanto, se debe esperar las actualizaciones del creador. (limita las actualizaciones ya que esta desarrollada)	Kivy es una librería de Python Open source
Costo	La realización de aplicaciones en la	La realización de aplicaciones en la	La realización de aplicaciones en la

	plataforma de React-native es totalmente gratuita	plataforma de App inventor es totalmente gratuita, pero al mismo tiempo bastante limitada.	plataforma de kivy es totalmente gratuita

Fuente: Investigadores

Justificación

Por tanto, una vez analizado estos tres lenguajes de programación, se llega a concluir que React-Native es la plataforma indicada para la creación de la aplicación, debido a que presenta una extensa comunidad de desarrollo que permite que el usuario puede hacer uso de APIs ya desarrolladas en sus aplicaciones y en caso de que tenga dudas en la creación del mismo.

- **Diagrama de flujo de la aplicación móvil conectado a la Base de datos Firebase**

En la figura 124, se visualiza la comunicación que tiene cada una de las ventanas con firebase.

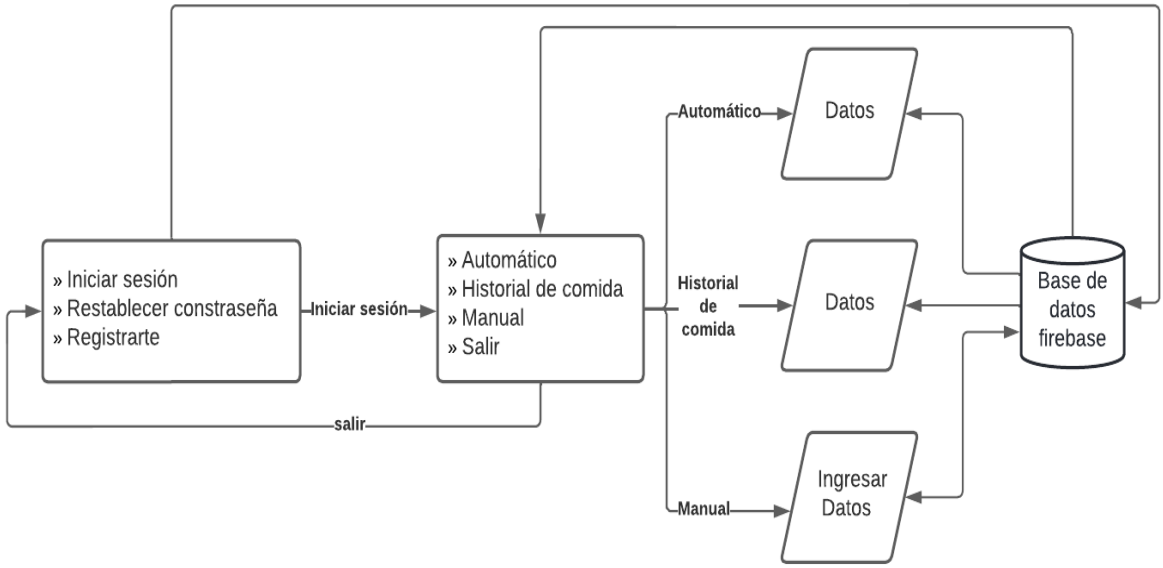


Figura 124: *Diagrama de flujo de la comunicación de React-native y Firebase*

Fuente: Investigadores

- **Diagrama de flujo del sistema de autenticación:**

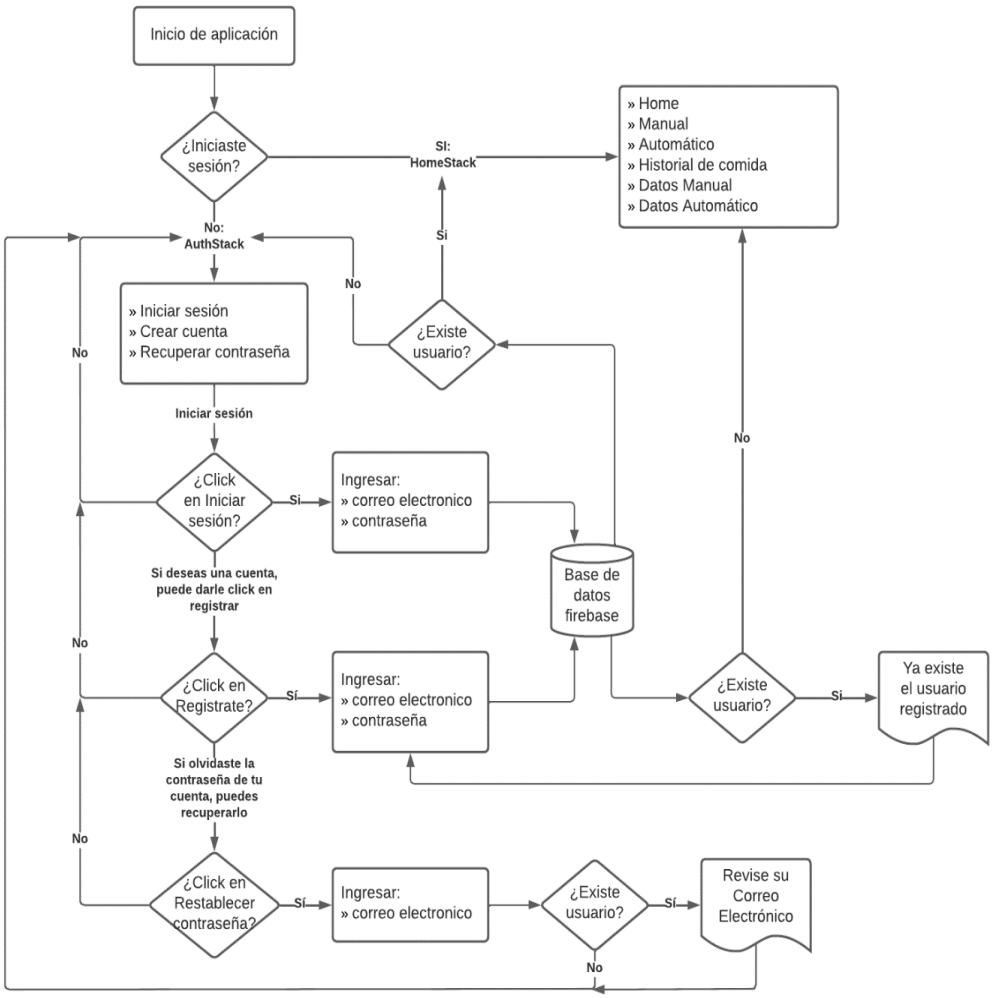


Figura 125: *Diagrama de flujo del funcionamiento de la aplicación móvil*

Fuente: Investigadores

Programación de la aplicación móvil

Para el funcionamiento correcto de la aplicación se tiene que instalar algunas librerías, en la tabla 52, se puede observar las librerías utilizadas en el proyecto.

Tabla 52: *Librerías instaladas para el desarrollo de la aplicación móvil*

Nombre de librería	Línea de código	Descripción
Stack Navigator	npm i @react-navigation/stack	La siguiente librería instalada permitirá la navegación o transición entre pantallas.
React Native Firebase	npm i @react-native-firebase/app	La siguiente librería instalada permitirá configurar la comunicación entre Firebase y react-native.
Firebase-Realtime Database	npm i @react-native-firebase/database	La siguiente librería instalada permitirá la comunicación con la base de datos en tiempo real de Firebase. Por lo que se pueden obtener los datos de manera actualizada.

Librería de Firebase-Authentication	npm i @react-native-firebase/auth	La siguiente librería instalada permitirá la creación de sistemas seguros, ya que proporciona servicios de BaaSS (BACKEDN).
paper	npm i react-native-paper	La siguiente librería instalada permitirá la creación de app más interactivas, la librería especifica que sea ha utilizado es "DataTable".
select dropdown	npm install react-native-select-dropdown	La siguiente librería instalada permitirá la creación de menús desplegables personalizados.
Gesture Handler	npm i react-native-gesture-handler	La siguiente librería instalada permitirá brindar a la aplicación una mejor experiencia en la parte de interacciones táctiles, etc.

Fuente: Investigadores

A continuación, se describió el funcionamiento de cada ventana creada y las líneas de código más importantes dentro de la programación de la aplicación móvil.

- **Creación de las ventanas Iniciar sesión, Crear cuenta, Recuperar contraseña**

En la figura 126, se puede observar la ventana de Iniciar sesión, la cual consta de 3 opciones: Iniciar sesión, Crear cuenta nueva y ¿Olvidaste tu contraseña? Cada una de estas opciones están conectadas a la plataforma de Firebase.

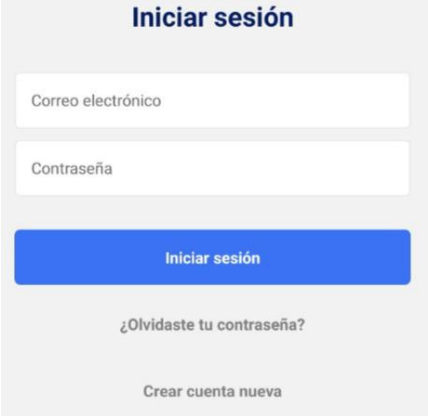


Figura 126: *Ventana de iniciar sesión*

Fuente: Investigadores

Una vez ingresado los datos, el usuario debe dar click en 'Iniciar sesión' tal como se observa en la figura 127, al dar click inmediatamente llama a la función 'login' el cual está encargado de verificar si los datos ingresados existen, en caso de no existir mandará un mensaje de error a la consola. El siguiente bloque de programación representa a la función 'login'.

```
login: async (email, password) => {
  try {
    await auth().signInWithEmailAndPassword(email, password);
  } catch (e) {
    console.log(e);
  }
```

Figura 127: *Función login*

Fuente: Investigadores

La ventana de Registro que se puede observar en la figura 128, está diseñada para la creación de un nuevo usuario, una vez ingresado el correo y la contraseña el botón 'Registrarte' enviará los datos a la Base de Datos de Firebase, dando un control de acceso satisfactorio para el usuario.

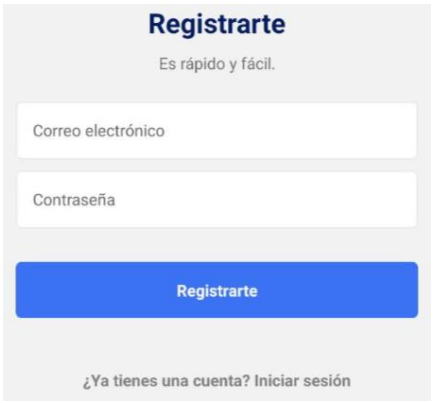


Figura 128: *Ventana de Registro*

Fuente: Investigadores

La línea de código encargado de la creación de usuarios se puede observar a continuación, cuando el usuario haya ingresado los datos de manera adecuada la función 'register' creará una cuenta nueva, en caso de no existir ningún error automáticamente será direccionado a la ventana principal denominada 'Home' y en caso de existir algún error mandara un mensaje de error a la consola.

```
register: async (email, password) => {
  try {
    await auth().createUserWithEmailAndPassword(email, password);
  } catch (e) {
    console.log(e);
  }
```

Figura 129: *Función register*

Fuente: Investigadores

La ventana de restablecimiento de contraseña, está diseñada para recuperar la contraseña del usuario en caso de olvidarlo. La figura 130, se muestra la ventana de restablecimiento de contraseña.

Figura 130: *Ventana de Restablecimiento de contraseña*

Fuente: Investigadores

Cuando el usuario ya haya ingresado el correo electrónico, deberá presionar click en 'Restablecer contraseña' tal como se muestra en la figura 131, el siguiente bloque de programación envía al correo ingresado un mensaje de restablecimiento de contraseña, al mismo tiempo en la pantalla del móvil se desplegará una pantalla de alerta pidiendo que el usuario revise su correo electrónico, es muy importante revisar estos correos en la bandeja de entrada o como spam. En caso de existir algún error se envía un mensaje de error a la consola.

```
forgotpassword: async (email) => {
  try {
    await auth().sendPasswordResetEmail(email);
    alert("Revise su correo electronico");
  } catch (e) {
    console.log(e);

}},
```

Figura 131: *Función forgotpassword*

Fuente: Investigadores

- **Creación de la ventana Home**

La ventana Home o también conocida como ventana principal, consta de un menú de opciones a diferentes ventanas y está encargada de mostrar el nivel de Tolva del sistema y usuario ingresado.

Figura 132: *Ventana Home*

Fuente: Investigadores

El bloque de programación utilizado para obtener el dato de la Tolva de Firebase-RealTime, se muestra en la figura 133. 'useEffect' junto con 'database().ref().on' están encargados en leer el dato de manera actualizada sin la necesidad de actualizar la ventana de Home, useEffect está compuesta por una función y un arreglo vacío de dependencias.

```
useEffect(() => {
  database().ref('/Tolva/Nivel/Value')
  .on('value', snapshot => {
    console.log('User data: ', snapshot.val());
    setValue(snapshot.val())
  });
}, []);
```

Figura 133: *Bloque de programación para la lectura de la tolva*

Fuente: Investigadores

- **Creación de la ventana Automático**

La ventana Automático expone los datos en tiempo real mediante la plataforma de Firebase-Realtime, además consta de un botón que permite navegar a una nueva ventana donde se visualiza los datos almacenados.

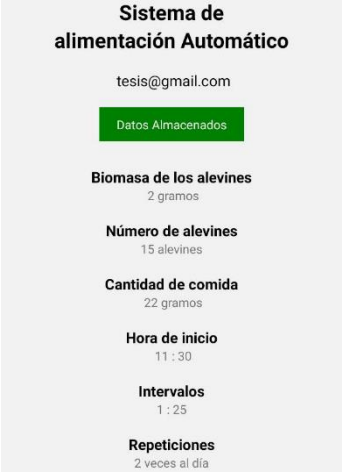


Figura 134: *Ventana de Modo Automático*

Fuente: Investigadores

Para la visualización de los datos almacenados en la plataforma de Firebase Real-Time se utilizan las siguientes líneas de código, tal como se muestra en la tabla 53. Es muy importante poner dentro de ref(), la dirección exacta de donde se quiere leer el dato, caso contrario no leerá nada.

Tabla 53: *Variables utilizadas para obtener datos de la ventana Automático*

Datos mostrados	useEffect está compuesta por una función y un arreglo vacío de dependencias.

biomasa	`const [biomasa, setBiomasa] = useState('');`
	`useEffect(() => {`
	`database().ref('/Datos RealTime-Automatico/sistema automatico/biomasa')`
	`.on('value', snapshot => {`
	`console.log('biomasa: ', snapshot.val());`
	`setBiomasa(snapshot.val())`
	`});`
	`}, []);`

comida	`const [comida, setComida] = useState('');`
	`useEffect(() => {`
	`database().ref('/Datos RealTime-Automatico/sistema automatico/comida')`
	`.on('value', snapshot => {`
	`console.log('comida: ', snapshot.val());`
	`setComida(snapshot.val())`
	`});`
	`},[])`

IntervaloHora	`const [intervaloHora, setIntervaloHora] = useState('');`
	`useEffect(() => {`
	`database().ref('/Datos RealTime-Automatico/sistema automatico/intervaloHora')`
	`.on('value', snapshot => {`
	`console.log('Hora intervalo: ', snapshot.val());`
	`setIntervaloHora(snapshot.val())`
	`});`
	`},[])`

IntervaloMinutos	`const [intervaloMinutos, setIntervaloMinutos] = useState('');`
	`useEffect(() => {`
	`database().ref('/Datos RealTime-Automatico/sistema automatico/intervaloMinutos')`
	`.on('value', snapshot => {`
	`console.log('Hora intervalo: ', snapshot.val());`
	`setIntervaloMinutos(snapshot.val())`
	`});`
	`},[])`

Hora	`const [hora, setHora] = useState('');`
	`useEffect(() => {`
	`  database().ref('/Datos RealTime-Automatico/sistema automatico/hora')`
	`  .on('value', snapshot => {`
	`  console.log('Hora intervalo: ', snapshot.val());`
	`  setHora(snapshot.val())`
	`  });`
	`},[])`

Minutos	`const [minutos, setMinutos] = useState('');`
	`useEffect(() => {`
	`  database().ref('/Datos RealTime-Automatico/sistema automatico/minutos')`
	`  .on('value', snapshot => {`
	`  console.log('Hora intervalo: ', snapshot.val());`
	`  setMinutos(snapshot.val())`
	`  });`
	`},[])`

Repeticiones	`const [repeticiones, setRepeticiones] = useState('');`
	`useEffect(() => {`
	`  database().ref('/Datos RealTime-Automatico/sistema automatico/repeticiones')`
	`  .on('value', snapshot => {`
	`  console.log('Hora intervalo: ', snapshot.val());`
	`  setRepeticiones(snapshot.val())`
	`  });`
	`},[])`

Peces	`const [peces, setPeces] = useState('');`
	`useEffect(() => {`
	`  database().ref('/Datos RealTime-Automatico/sistema automatico/peces')`
	`  .on('value', snapshot => {`
	`  console.log('Hora intervalo: ', snapshot.val());`
	`  setPeces(snapshot.val())`
	`  });`
	`},[])`

Fuente: Investigadores

- **Creación de la ventana Manual**

La ventana Manual es la encargada de enviar los datos a la base de datos Firebase Real-time, además con ayuda de un botón permitirá al usuario navegar a una nueva ventana en donde se visualizará los datos almacenados.

Figura 135: *Ventana de Modo Manual*

Fuente: Investigadores

Para el envió de datos desde React-native hasta la plataforma de Firebase, se utilizará el siguiente bloque de programación. Al presionar click en 'Guardar' inmediatamente llama a la función create, encargada en subir los datos en tiempo real teniendo en cuenta que no se hará uso de useEffect, sino de una función junto con 'database().ref().set()'. En caso de existir algún error se notificá por consola.

```
function create  ()  {

  database()
  .ref('/Datos RealTime-Manual/')
  .set({
    cantidad: cantidad,
    hora: hora,
    minutos: minutos,
    intervaloHora: intervaloHora,
    intervaloMinutos: intervaloMinutos,
    repeticiones: repeticiones,
    usuario: user.email,
    descripcion:"El sistema se ejecutará a la/s " +hora+ ":" +minutos+
    horaactual: new Date().toLocaleTimeString(),
    fechaactual: new Date().toLocaleDateString(),

  })
  .then(() => console.log('Data set.'))
  .catch(error => {
    console.log(error)
  })
```

Figura 136: *Código para el envió de datos a Datos RealTime-Manual*

Fuente: Investigadores

- **Creación de las ventanas Historial de comida, Base de datos manual, Base de datos automático**

Las ventanas representadas en las figuras 137, 138, 139, se utilizarán para la visualización de todos los datos que se van guardando cuando se programa, ya sea de manera Automático o Manual.

La ventana Historial de comida de la figura 137, indica de manera ordenada la cantidad de comida que se ha dispensado, hora y fecha, estos datos se exponen después de que el sistema haya dispensado.

El bloque de programación que se utilizó para la visualización de los datos en la ventana, se puede observar en la figura 142. Y para la lectura de los datos de Firebase Real-time, se muestra en la figura 140.

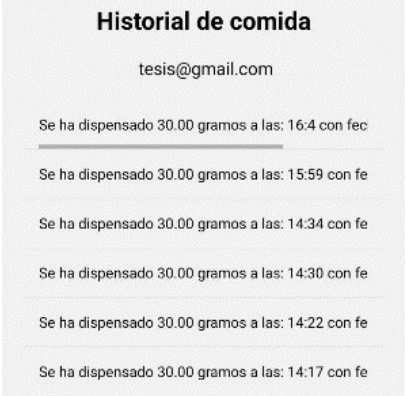


Figura 137: *Ventana de historial de comida*

Fuente: Investigadores

La ventana Base de Datos del sistema Manual, como se muestra en la figura 138, indica los datos que se almacenan al momento de dar click en 'Guardar', este botón se puede visualizar en la figura 135. El bloque de programación que permite el almacenamiento en la base de datos, se encuentra en la figura 136 y el bloque de programación para la visualización de los datos, se encuentra en la figura 141.

Figura 138: *Ventana de Base de Datos Manual*

Fuente: Investigadores

La ventana Base de Datos del sistema Automático, como se expone en la figura 139. muestra los datos que se encuentra almacenados en la plataforma de Firebase Real-time, estos datos son almacenados por el "contador de alevines". El bloque de programación que permite la visualización de los datos, se encuentra en la figura 141.

Figura 139: *Ventana de Base de Datos Automático*

Fuente: Investigadores

A continuación, se presenta los bloques de programación de uno de los aspectos más importantes de las ventanas: Historial de comida, Base de datos manual, Base de datos automático.

```
const [comida, setComida] = useState({});
useEffect(() => {
    database().ref('Base de Datos')
    .on('value', snapshot => {
      console.log(snapshot.val())
      if (snapshot.val() != null)
      setComida({
          ...snapshot.val()
      })
    });
  },[])
```

Figura 140: *Código para la lectura de datos en tiempo real*

Fuente: Investigadores

```
Object.keys(comida).map(id => {

    return <DataTable.Row key={id}>

        <DataTable.Cell >{comida[id].usuario}</DataTable.Cell>
        <DataTable.Cell>{comida[id].fechaactual}</DataTable.Cell>
        <DataTable.Cell>{comida[id].horaactual}</DataTable.Cell>
        <DataTable.Cell>
        <Mostrar text="El Sistema..."  onPress={() => alert(comida[id].descripcion)}
        </DataTable.Cell>

    </DataTable.Row>
})
```

Figura 141: *Código para la visualización de los datos Automático/Manual*

Fuente: Investigadores

```
<DataTable>
{
        Object.keys(comida).map(id => {
        return <DataTable.Row key={id}>
        <ScrollView horizontal={true}>
        <DataTable.Cell >{comida[id].comida}</DataTable.Cell >
        </ScrollView>
</DataTable.Row>
                                        })
}
</DataTable>
```

Figura 142: *Código para la visualización del historial de comida*

Fuente: Investigadores

3.1.2 Funcionamiento

El sistema cuenta con dos modos de funcionamiento. El modo automático, el cual necesita del contador de alevines (sistema de visión artificial) para poder funcionar, ya que este es el encargado de calcular la rutina de alimentación en función de la biomasa encontrada. Por otro lado, el modo manual no necesita del contador de alevines, los parámetros de la rutina de alimentación son definidos por el piscicultor. Tanto para el modo manual como el automático se debe verificar que el dispensador de alimento esté encendido para cumplir con las rutinas de alimentación definidas.

Antes de ingresar en el sistema de alimentación, previamente hay que iniciar sesión. Mediante un correo electrónico y contraseña. En caso, de no estar registrado o haber olvidado la contraseña, se debe crear una cuenta o entrar en la opción para recuperar la contraseña respectivamente.

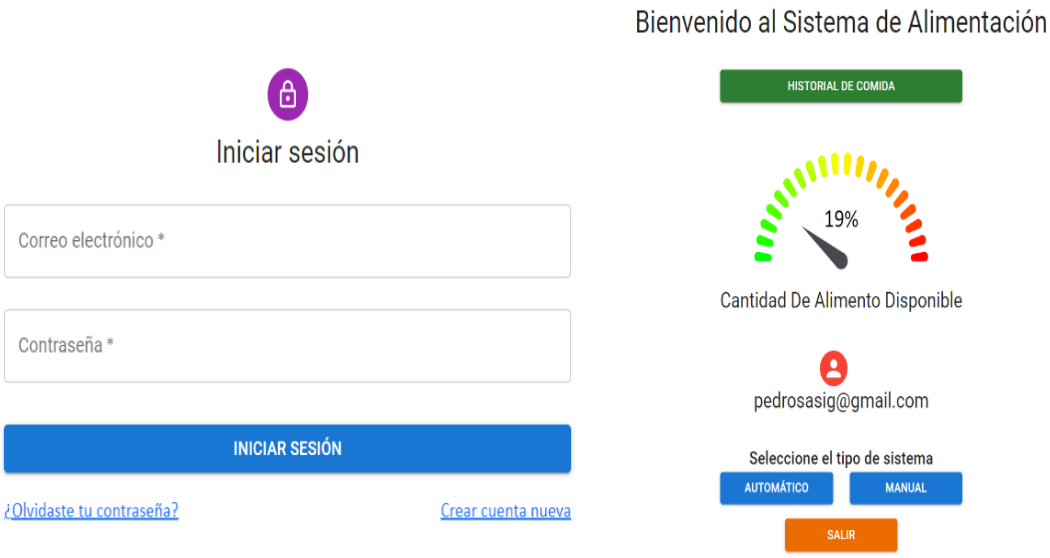


Figura 143: *De izquierda a derecha. Inicio de sesión y página principal*

Fuente: Investigadores

Automático

Para empezar con el modo automático. Primeramente, hay que empezar con el conteo de los alevines, para ello hay que tomar con un recipiente a los ejemplares y soltarlos en el depósito a las entradas de las canaletas.

Figura 144: *De izquierda a derecha. Vista del depósito para el ingreso de los alevines y colocación de los alevines para el proceso de conteo*

Fuente: Investigadores

Cuando se haya terminado con el proceso de conteo, hay que presionar el botón enviar para subir la rutina de alimentación hacia Firebase. O si el piscicultor decide empezar de nuevo el proceso de conteo puede presionar el botón reiniciar para borrar la información anterior.

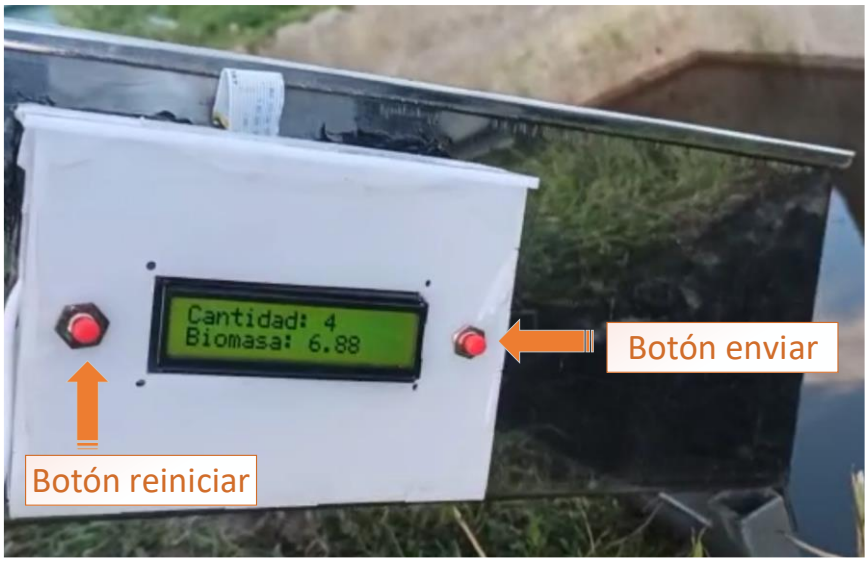


Figura 145: *Botones del contador de alevines*

Fuente: Investigadores

Cuando el contador de alevines no ha enviado datos todavía hacia Firebase, el botón "AUTOMÁTICO" se encuentra deshabilitado por defecto y en ese caso solo se puede

usar el modo “MANUAL”. Por el contrario, cuando se envían los datos desde el contador hacia Firebase el botón “AUTOMÁTICO” se habilita

Figura 146: *De izquierda a derecha. Botón “AUTOMÁTICO” deshabilitado en la página web y botón habilitado al enviar la rutina de alimentación desde el contador*

Fuente: Investigadores

Al dar clic en el botón “AUTOMÁTICO” se expondrá la información que ha enviado el contador de alevines hacia la base de datos (Firebase). En ella se puede visualizar la biomasa, número de alevines, la cantidad de comida que debe suministrar el dispensador, la hora a la que se empezará a dispensar el alimento, los intervalos y el número de repeticiones.

Cada vez que el contador de alevines envíe información hacia la base de datos, se guardará un registro de actividad (información que se puede visualizar al presionar el botón con el mismo nombre) con la fecha y hora. Así, como la rutina de alimentación y otra información relevante.

Figura 147: *De izquierda a derecha. Visualización desde la página web de la información enviada desde el contador de alevines y registro de actividades del contador*

Fuente: Investigadores

El modo automático funcionará hasta que el piscicultor decida cambiar la rutina de alimentación, por una nueva que vaya acorde al proceso de crecimiento de las truchas

Manual

En el modo manual solo se necesita ingresar a la página web o aplicación móvil para empezar a definir los parámetros de la rutina de alimentación en función de las necesidades del piscicultor. El modo automático funciona hasta que el piscicultor decida cambiar la rutina de alimentación, por una nueva que vaya acorde al proceso de crecimiento de las truchas

En caso de que el contador haya mandado datos previamente y se desee ingresar en el modo “MANUAL” se indica un aviso para que el usuario tenga precaución de no alterar los datos del modo “AUTOMÁTICO” en caso de que haya ingresa en el modo “MANUAL” accidentalmente. Caso contrario, dar clic en continuar.

La ventana modal solo se mostrará cuando haya datos del contador, caso contrario se ingresará normalmente.

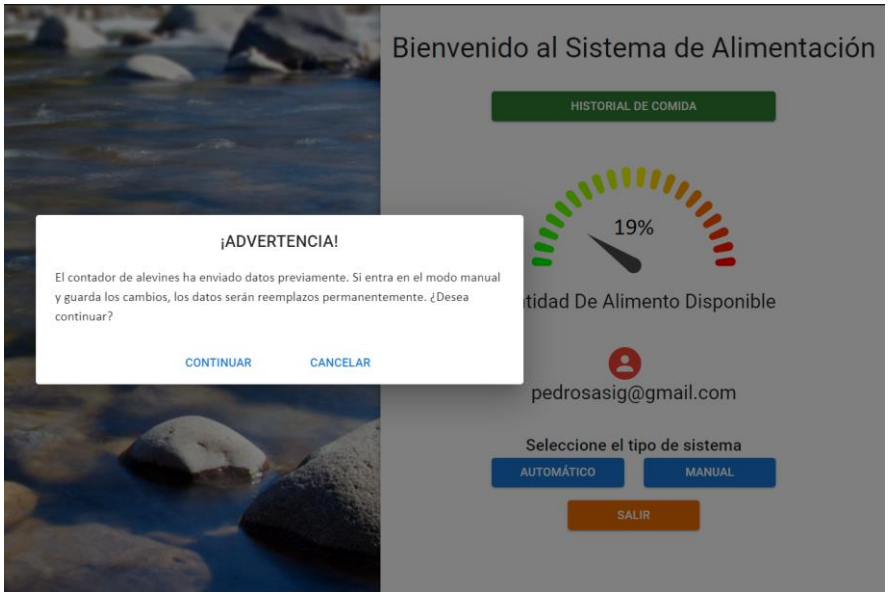


Figura 148: *Ventana modal en caso de querer ingresar en el modo "MANUAL"*

Fuente: Investigadores

En el modo "MANUAL" se expone una ventana para ingreso de los valores para la rutina de alimentación. En esta se puede ingresar la cantidad de comida. Así como el horario que empezará a surtir el alimento, sus intervalos y las repeticiones que hará durante el día. Esta información será tomada por la ESP32 para empezar a dispersar el alimento.

Del mismo que en el modo "AUTOMÁTICO" se tiene un registro de actividades que guardará la fecha y hora en la que se configuró el modo "MANUAL".

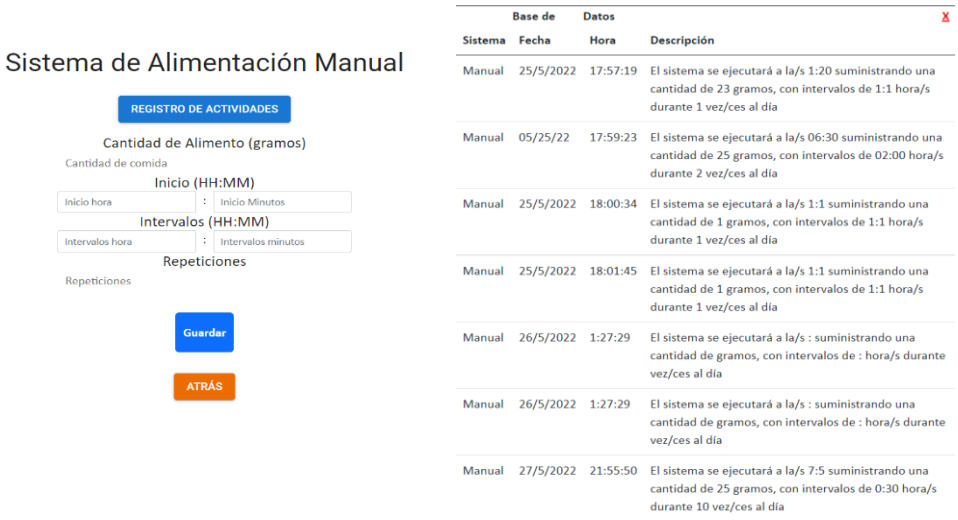


Figura 149: *De izquierda a derecha. Ventana para el ingreso de la rutina de alimentación y visualización del registro de actividades del modo "MANUAL"*

Fuente: Investigadores

General

De modo general, es decir, para el modo manual y automático se tiene el botón para el historial de comida que indica la cantidad de gramos que se ha dispersado indicando su fecha y hora.

Además, se muestra mediante un indicador el porcentaje de balanceado disponible en el dispensador.

Figura 150: *De izquierda a derecha. Ventana principal de la página web y ventana del historial de comida*

Fuente: Investigadores

Figura 151: *De izquierda a derecha. Dispensación del balanceado y contador de alevines junto al dispensador de alimento*

Fuente: Investigadores

3.1.3 Desarrollo de pruebas

El sistema implementado se divide en 2 partes, el dispensador y el contador. Para que el dispensador funcione de manera automática depende del funcionamiento del contador, por lo que a continuación se hizo el análisis de cada uno de esos sistemas.

Dispensador

Pruebas del rpm, tiempo y cantidad

Para el periodo de pruebas de ha utilizado una balanza electrónica CAMRY EK3213 con un margen de error de $+/-$ 2 gramos.

Durante el desarrollo de las pruebas se ha pesado diferentes cantidades de alimento con distintas RPM. Para ello se ha tomado como variable independiente a las RPM del sinfín con el fin de determinar hasta qué punto el tiempo de dispensación de alimento mejora a medida que la variable se incrementa.

Tabla 54: *Tiempos de dispensación*

RPM	Tiempo (segundos) de dispensación para los distintos pesos medidos				
	Cantidad 28 gramos	Cantidad 56 gramos	Cantidad 80 gramos	Cantidad 110 gramos	Cantidad 132 gramos
25	61	95	140	192	252
50	29	52	77	98	119
75	29	52	75	97	118
100	12	24	36	50	62
125	10	25	37	48	59
150	11	25	38	49	60
175	12	24	37	50	60
200	10	24	38	48	58
225	11	25	38	48	59
250	12	25	36	50	60

Fuente: Investigadores

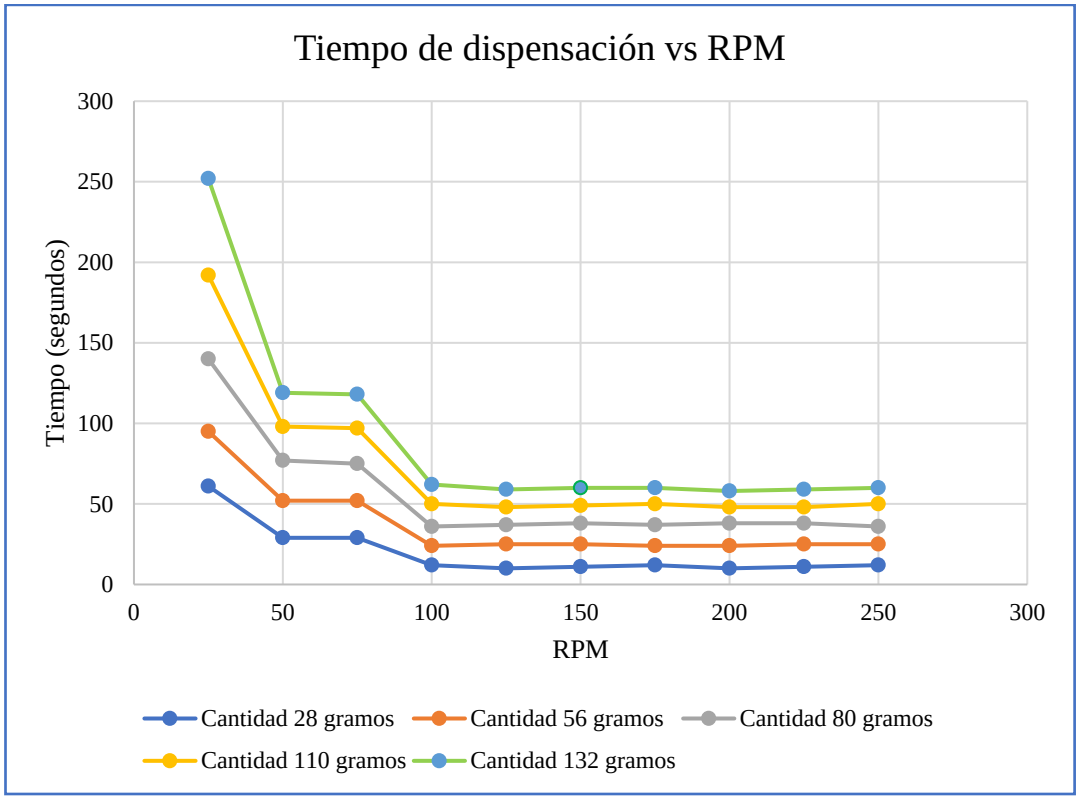


Figura 152: Tiempo de dispensación de alimento vs RPM

Fuente: Investigadores

Gráficamente, se puede observar que a 25 RPM los tiempos de dispensación son exagerados. Por ejemplo, para una cantidad de 132 gramos, se tarda al menos 4 minutos y 12 segundos. Haciéndolo el sistema ineficiente. Además, el tiempo que el sinfín tarda en llevar el alimento hasta el recipiente de la balanza hace que no entregue la comida a la hora indicada. Por ejemplo, para la misma cantidad a 25 RPM y programada para entregar el alimento a las 8 am, lo estaría dispensado a las 8:04 am.

Siguiendo con el análisis, se puede observar gráficamente, que para todas las cantidades pesadas el tiempo de dispensación mejora a los 50 RPM y se mantiene constante hasta las 75 RPM. Pero, al llegar a las 100 RPM hay de nuevo una mejora significativa. A partir de las 100 RPM se puede observar que el tiempo de dispensación se mantiene constante para todas las cantidades pesadas, por lo que se puede definir este valor como ideal ya que desde allí ya no habrá cambios relevantes en el tiempo.

Pruebas de error de peso con el deseado

Para las pruebas de precisión se han tomado 10 muestras para cada cantidad deseada. El peso ha sido medido mediante la misma balanza electrónica. Por lo tanto, se ha obtenido los siguientes resultados.

Tabla 55: *Cantidades pesadas para cada muestra deseada*

Cantidad deseada (gramos)	Número de prueba									
	1	2	3	4	5	6	7	8	9	10
	Cantidad pesada (gramos)									
25	28	28	29	29	29	29	28	28	30	28
50	56	55	55	56	55	56	55	55	57	56
75	80	80	81	80	79	78	88	81	80	85
100	105	109	107	109	106	106	109	105	108	109
125	132	133	134	132	133	132	132	133	132	133
150	157	158	156	157	159	159	159	160	156	157

Fuente: Investigadores

Como se puede observar en la tabla 55, la cantidad pesada siempre tiene algunos gramos adicionales con respecto a la cantidad deseada. Durante la serie de diez pruebas, para cada cantidad requerida, los valores pesados no difieren tanto uno de los otros.

Tabla 56: *Diferencia obtenida para cada muestra deseada*

Cantidad deseada (gramos)	Número de prueba										Promedio de diferencias (gramos)
	1	2	3	4	5	6	7	8	9	10	
	Diferencia (gramos)										
25	3	3	4	4	4	4	3	3	5	3	3,6
50	6	5	5	6	5	6	5	5	7	6	5,6
75	5	5	6	5	4	3	13	6	5	10	6,2
100	5	9	7	9	6	6	9	5	8	9	7,3
125	7	8	9	7	8	7	7	8	7	8	7,6
150	7	8	6	7	9	9	9	10	6	7	7,8

Fuente: Investigadores

A medida que la cantidad requerida se incrementa, la cantidad de gramos adicionales que el dispensador botó también lo hace. Por ejemplo, para cantidades pequeñas el

promedio de diferencias (promedio entre las cantidades deseadas menos las pesadas) es de 3,6 gramos. Mientras, para cantidades grandes el promedio es 7,8 gramos.

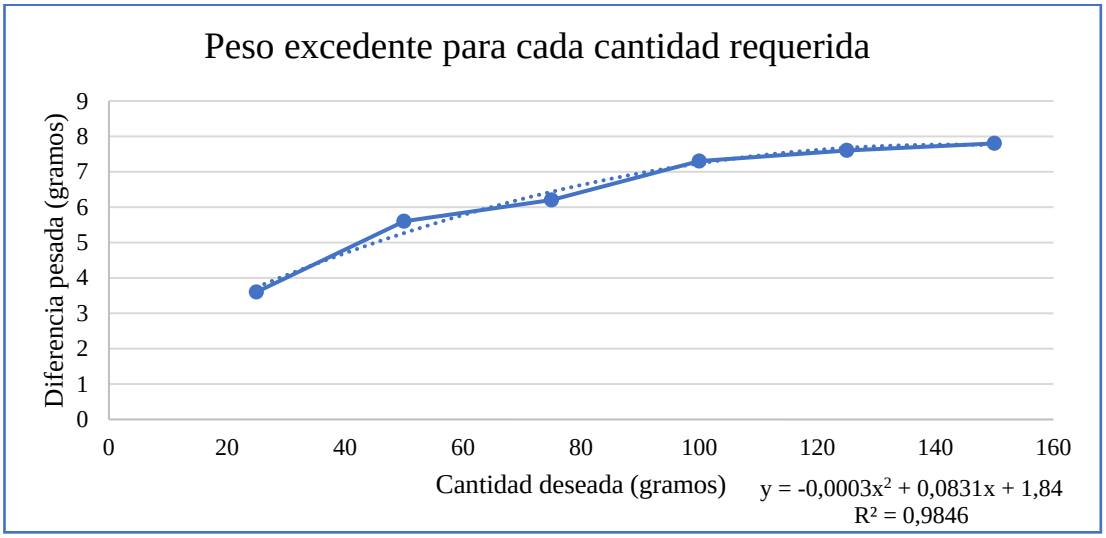


Figura 153: *Relación entre el peso excedente y la cantidad deseada*

Fuente: Investigadores

Por lo tanto, para determinar si hay una relación entre estos gramos adicionales y la cantidad deseada, se ha procedido a graficar estas variables.

En la figura 153, se puede observar que a medida que la cantidad requerida aumenta esta diferencia también lo hace, y esto se ha representado gráficamente.

Esta relación será utilizada para definir una regresión cuadrática. La ecuación de la curva encontrada ayudará a determinar un valor de corrección en función de la cantidad requerida. Por ejemplo, si la cantidad requerida es de 100 gramos, el dispensador podría entregar 108 gramos y mediante la corrección $y = -0{,}0003(100)^2 + 0{,}0831(100) + 1{,}84 = 7{,}15$ se obtendría un peso final de 100,85 gramos, este valor está más cerca del peso real requerido.

Tabla 57: *Porcentaje de precisión para una muestra de 100 gramos antes de ser corregido*

# Prueba	Cantidad definida (gramos)	Cantidad pesada (gramos)	Error (%)
1	100	105	5
2	100	109	9
3	100	107	7
4	100	109	9
5	100	106	6
6	100	106	6
7	100	109	9
8	100	105	5
9	100	108	8
10	100	109	9
Promedio errores (%)			7,3

Fuente: Investigadores

Antes de la implementación del valor de corrección la precisión del dispensador es del 92,7 % para una cantidad de 100 gramos.

Tabla 58: *Porcentaje de precisión para una muestra de 100 gramos después de ser corregido*

# Prueba	Cantidad definida (gramos)	Cantidad pesada (gramos)	Error (%)
1	100	99	1
2	100	98	2
3	100	101	1
4	100	100	0
5	100	102	2
6	100	100	0
7	100	99	1
8	100	98	2
9	100	98	2
10	100	101	1
Promedio errores (%)			1,2

Fuente: Investigadores

Después de implementar la corrección, se ha mejorado el sistema de dispensación con una precisión del 98,7%. Una mejora del 6% con respecto al estado anterior.

Pruebas de dispersión de alimento

Para el desarrollo de las pruebas se ha utilizado recipientes desechables de 14 centímetros de ancho y 24 centímetros de largo. Estos han sido colocados tanto de manera vertical como horizontal en el área de dispersión de alimento. Cuando el alimento es lanzado se ha procedido a pesar el conjunto de recipientes que están en posición vertical (señalados en color rojo) para determinar la cantidad que ha caído a lo largo del eje horizontal.

Figura 154: *Distribución de los recipientes*

Fuente: Investigadores

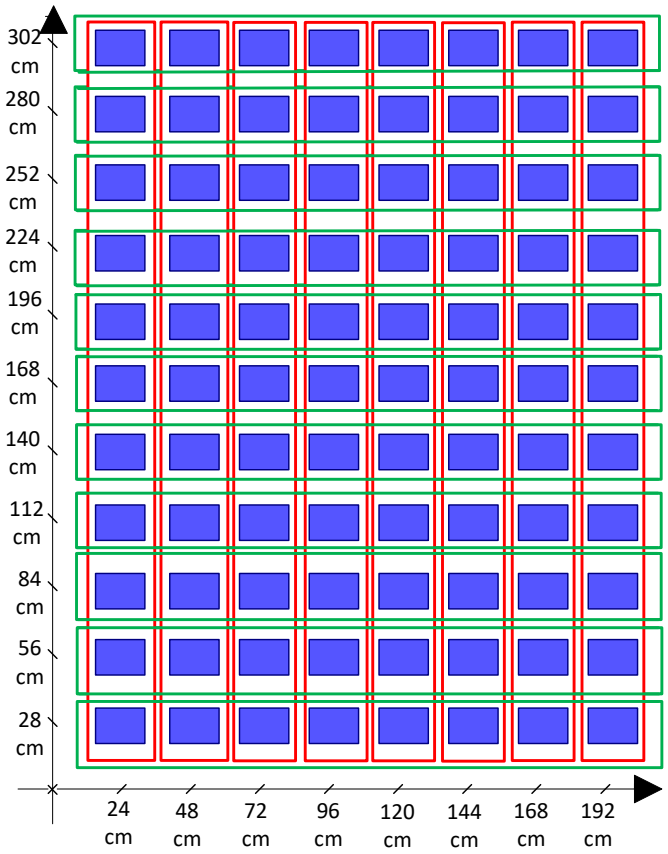


Figura 155: *Disposición de los recipientes en el área de dispersión*

Fuente: Investigadores

Tabla 59: *Distribución de alimento en cada intervalo de distancia horizontal*

Distancia horizontal (cm)	Distancia vertical (302 cm)					Promedio (gramos)
	# Prueba					
	1	2	3	4	5	
	Cantidad dispersada (gramos)					
24	3	2	2	1	2	2
48	4	5	7	5	5	5,2
72	29	28	27	29	26	27,8
96	54	55	55	54	59	55,4
120	58	57	55	57	58	57
144	43	44	45	43	44	43,8
168	7	5	6	8	4	6
192	1	2	1	1	1	1,2
Total	199	198	198	198	199	198,4

Fuente: Investigadores

En la tabla 59, se puede observar que para cada intervalo de distancia horizontal se ha realizado 5 pruebas respectivamente y se ha calculado el promedio para cada intervalo.

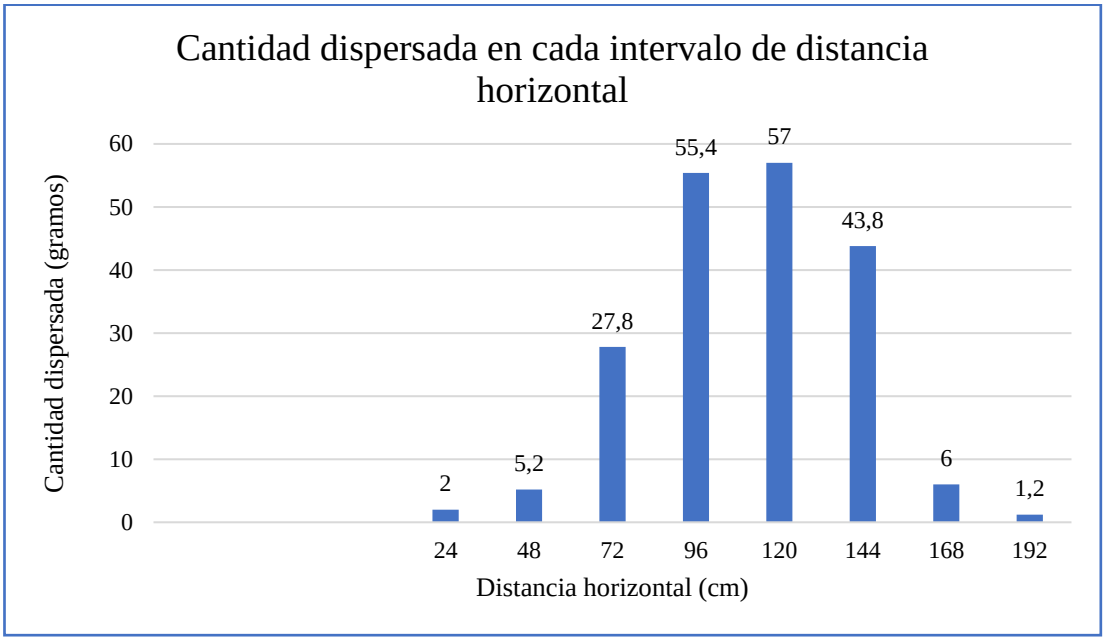


Figura 156: *Cantidad dispersada en cada intervalo de distancia horizontal*

Fuente: Investigadores

Gráficamente, se puede apreciar que hacia los extremos la cantidad de alimento recolectada es mínima. Pero, a medida que se acerca al centro de la gráfica, entre los intervalos de 96 – 120 cm hay mayor cantidad, por lo tanto, mayor dispersión. Esto también se ve influenciado por la ubicación del dispensador que se encuentra en el intervalo 96 – 120 cm. Haciendo que la dispersión tienda a estar centralizada, con poco alcance hacia los espacios laterales.

Tabla 60: *Distribución de alimento en cada intervalo de distancia vertical*

Distancia vertical (cm)	Distancia horizontal (192 cm)					Promedio (gramos)
	# Prueba					
	1	2	3	4	5	
	Cantidad dispersada (gramos)					
28	17	17	16	16	15	16,2
56	15	14	18	15	15	15,4
84	28	29	26	29	25	27,4
112	43	42	43	53	50	46,2
140	43	40	41	38	42	40,8
168	25	28	26	27	30	27,2
196	16	15	17	10	12	14
224	10	10	8	9	9	9,2
252	2	3	2	1	1	1,8
280	0	0	1	0	0	0,2
308	0	0	0	0	0	0
Total	199	198	198	198	199	198,4

Fuente: Investigadores

Para determinar la cantidad de alimento que ha caído en los intervalos de la posición vertical, se ha procedido a pesar el conjunto de recipientes (señalados en color verde) en posición horizontal. Del mismo modo que el apartado anterior, se ha realizado 5 pruebas y se ha tomado el promedio de las cantidades pesadas.

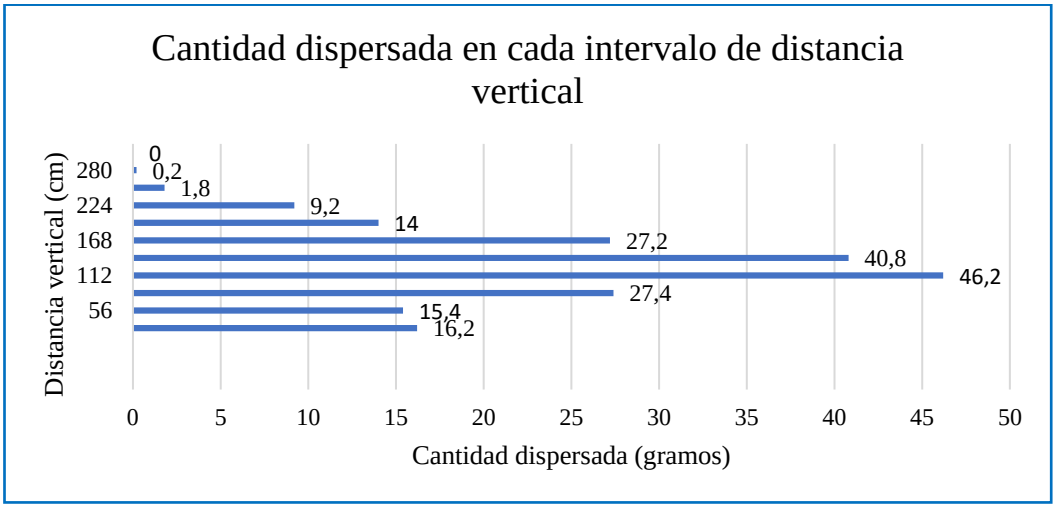


Figura 157: *Cantidad dispersada en cada intervalo de distancia vertical*

Fuente: Investigadores

Gráficamente, se puede observar que para las primeras distancias de lanzamiento 28 – 56 cm la cantidad recolecta es poca en comparación a mayores distancias. A una distancia de 112 centímetros del dispensador se obtiene el punto máximo de gramos recolectados (46,2 gramos). Pero a medida, que se incrementa la longitud de lanzamiento los gramos recolectados disminuyen. Así, que la distancia máxima hasta donde se puede dispersar el alimento es 252 centímetros. A partir de allí, solo llegarán algunos granos de alimento, que debido a su peso despreciable no pudieron ser pesados correctamente.

Pruebas de horario

Con el fin de probar el sistema, se ha definido un horario recomendado para la alimentación de alevines, el cual corresponde empezar desde las 8 am, en intervalos de una hora durante 10 veces al día máximo.

Tabla 61: *Horario de dispersión de alimento*

# Prueba	Cantidad definida 20 gramos	
	Horario definido	Horario dispersado
1	8:00	8:00
2	9:00	9:00
3	10:00	10:00
4	11:00	11:00
5	12:00	12:00
6	13:00	13:00
7	14:00	14:00
8	15:00	15:00
9	16:00	16:00
10	17:00	17:00

Fuente: Investigadores

Con respecto al horario de dispersión se ha obtenido resultados totalmente satisfactorios. En donde, el horario de dispersión de alimento corresponde al horario que se ha definido previamente. Cabe aclarar que, a partir de cantidades requeridas mayores a 150 gramos, el tiempo que tarda el sinfín en llevar el alimento hasta la balanza puede ser mayor a un minuto, haciendo que el tiempo de dispensación de

alimento sea la hora definida más un minuto. Este minuto adicional no influye en el proceso de alimentación.

Contador

Prueba del ángulo para el conteo y biomasa

A continuación, se mostrará los datos obtenidos para diferentes ángulos de la pendiente del contador. Con el fin de determinar, si el ángulo de la pendiente influye en el proceso de conteo y cálculo de la biomasa.

Tabla 62: *Pendiente del contador con un ángulo de 30 grados*

Pendiente 30 grados							
Distancia (m)	Tiempo (s)	Velocidad (m/s)	¿Cuenta?	Tamaño Visión Artificial (cm)	Tamaño real alevín (cm)	Canal	Porcentaje error
0,50	1,04	0,481	No	Ninguno	4,10	1	N. A
0,50	1,03	0,485	No	Ninguno	4,10	2	N. A
0,50	1,01	0,495	No	Ninguno	4,10	3	N. A
0,50	1,05	0,476	No	Ninguno	4,10	4	N. A
0,50	1,07	0,467	No	Ninguno	4,10	5	N. A

Fuente: Investigadores

El contador con una pendiente de 30 grados, no tiene la capacidad para detectar los alevines. Por lo tanto, no puede determinar la talla de los peces y por ende tampoco puede contar en ninguno de los cinco canales que se han diseñado. Se puede decir que, si el alevín pasa a través de la cámara con una velocidad superior a los 0,4 $^{m}/_{s}$ no será detectado.

Tabla 63: *Pendiente del contador con un ángulo de 25 grados*

Pendiente 25 grados							
Distancia (m)	Tiempo (s)	Velocidad (m/s)	¿Cuenta?	Tamaño Visión Artificial (cm)	Tamaño real alevín (cm)	Canal	Porcentaje error
0,50	2,12	0,236	SI	3,21	4,10	1	21,71
0,50	2,62	0,191	SI	3,83	4,10	2	6,59
0,50	2,37	0,211	SI	3,65	4,10	3	10,98
0,50	1,96	0,255	SI	2,28	4,10	4	44,39
0,50	2,19	0,228	SI	3,01	4,10	5	26,59

Fuente: Investigadores

Con una pendiente de 25 grados, el sistema de visión artificial si puede detectar a los alevines, en todos los canales. Por lo tanto, el proceso de conteo fue exitoso. Por otra parte, la determinación de la talla de los peces fue incorrecta con errores de hasta el 44,39 %, hace que el sistema no sea preciso. Por lo que, se puede decir que para velocidades mayores a 0,191 $^{m}/_{s}$ y menores 0,255 $^{m}/_{s}$ el sistema contará los alevines, pero no determinará correctamente la talla de los peces.

Tabla 64: *Pendiente del contador con un ángulo de 20 grados*

Pendiente 20 grados							
Distancia (m)	Tiempo (s)	Velocidad (m/s)	¿Cuenta?	Tamaño Visión Artificial (cm)	Tamaño real alevín (cm)	Canal	Porcentaje error
0,50	3,21	0,156	SI	4,27	4,10	1	4,15
0,50	3,01	0,166	SI	4,24	4,10	2	3,41
0,50	3,15	0,159	SI	4,03	4,10	3	1,71
0,50	3,81	0,131	SI	4,29	4,10	4	4,63
0,50	3,67	0,136	SI	4,30	4,10	5	4,88

Fuente: Investigadores

Con una pendiente de 20 grados, se ha conseguido que el sistema detecte el alevín correctamente. Por lo tanto, se ha contabilizado correctamente los alevines que pasaron por los canales. Además, la talla del alevín tiene un error máximo de 4,88 % que es aceptable en comparación con las pruebas realizadas con pendientes mayores. Con esto se puede decir que, para que el sistema funcione correctamente el alevín debe deslizarse con una velocidad inferior a 0,166 $^{m}/_{s}$.

Para obtener mayor precisión se podría disminuir el ángulo de la pendiente, pero hay que recordar el sistema de conteo es por gravedad, por lo que hay riesgo de que el alevín se quede atorado en los canales.

Pruebas de precisión de número, biomasa y tiempo

Durante el desarrollo de las pruebas se ha probado el contador con diferentes cantidades de alevines que van en intervalos de 5 unidades hasta llegar a los 50 alevines contados.

Tabla 65: *Datos del sistema del contador de alevines*

# De alevines	Alevines Contados V. A	Peso balanza (gramos)	Peso en V.A (gramos)	Error conteo %	Error Biomasa %	Tiempo (minutos)
5	5	6	4,66	0,00	22,33	1
10	11	8	6,52	10,00	21,45	2
15	14	15	12,25	6,67	17,23	3
20	19	19	16,47	5,00	13,81	4
25	26	24	20,01	4,00	15,75	5
30	27	26	22,52	10,00	12,24	8
35	32	30	26,11	8,57	13,37	9
40	38	36	33,19	5,00	7,57	12
45	48	39	35,52	6,67	8,92	14
50	54	44	41,17	8,00	6,43	19
Promedio				6,39	13,91	

Fuente: Investigadores

Según los datos obtenidos, el sistema tiene una precisión global del 93,61 % para el proceso de conteo. Mientras para el proceso de cálculo de biomasa tiene una precisión global del 86,09 %. El trabajar con especies vivas hace difícil alcanzar una mejor precisión, en especial con especies como los alevines de truchas, que tienen un movimiento errático e impredecible, que provoca en ocasiones que el alevín no sea captado completamente en posición vertical, por lo tanto, se estima una talla diferente con respecto a la original.

3.1.4 Costo del proyecto

El costo de los materiales que se han utilizado durante el desarrollo del proyecto se puede ver en la siguiente tabla.

Tabla 66: *Costo de los materiales para el desarrollo del proyecto*

Descripción	Cantidad	Valor unitario (USD)	Valor total (USD)
Acero inoxidable, 1220 x 2440 milímetros	2	189	378
Baquelita, 30 x 30 centímetros	1	7,5	7,5
Gabinete plástico, 20 x 20 x 15 centímetros	1	19,8	19,8
Batería 12v, 7.5 Ah	1	27	27
Regulador de carga 10 Amperios	1	18	18
Panel solar policristalino 50 W	1	97,45	97,45
Motor paso a paso nema 23	1	35,8	35,8
Acople nema 23 8 a 8 milímetros	1	7	7
Driver stepper TB6560	1	19,87	19,87
Motor 775 DC 12v	1	19,5	19,5
Driver motor L298N	1	5,5	5,5
Servomotor MG946R	1	10	10
Celda de carga flexible 5Kg	1	4,5	4,5
Modulo HX711	1	5,5	5,5
Sensor HC-SR04	1	4,5	4,5
Reguladores de voltaje 7805	4	1,75	7
Disipadores de calor	4	0,75	3
Módulo relé	1	3,5	3,5
ESP32	1	15,8	15,8
Borneras	15	0,25	3,75
Raspberry Pi 3	1	72,5	72,5
Pi camera v1.3	1	18	18
LCD 16x2	1	5,5	5,5
Bomba sumergible	1	52,5	52,5
sensor DS18B20	1	4,25	4,25
Alimento de alevines de trucha 25 kg	1	30	30
Balanza electrónica gramera	1	15	15
Luces led 12v	6	1,5	9
Refrigeración raspberry	1	15	15
Fuente 12v 15 A	1	15	15
Fuente raspberry 5v 3A	1	12,5	12,5
Router TP-Link_B300	1	32,75	32,75
Cable UTP Cat 5	40	0,75	30
Total			1004,97

Fuente: Investigadores

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se determinó que la temperatura del agua es directamente proporcional al porcentaje de la tasa de alimentación, es decir, entre más alta sea la temperatura, el alimento requerido será mayor y viceversa. Por ejemplo, durante el mes de mayo se obtuvo una temperatura promedio mensual de 13.42 grados Celsius. Para utilizar este valor en la tabla de alimentación se ha redondeado a su inmediato superior (14), dando una tasa de 6.91%. Durante el periodo de pruebas se utilizó 50 alevines con una talla promedio de 5 centímetros y una biomasa total de 57.5 gramos, obteniendo una ración diaria de 0.5 gramos que fue suministrada durante 8 veces al día, durante un mes. Además, Las truchas en etapa de alevinaje deben consumir balanceados con altos porcentajes de proteínas (44 – 50%), donde cada partícula de alimento mide 1.2 milímetros de diámetro con el fin de garantizar que todos los peces se alimenten.
- Se definió que la tarjeta que mejor se adapta al sistema del dispensador de alimento fue la ESP32. Esta se encarga de conectarse con la base de datos en Firebase para enviar el estado del nivel de la tolva e información relevante cada vez que se dispensa el balanceado de peces. Del mismo modo, para recibir la rutina de alimentación, se utilizó la API de Firebase, esta comunicación permite realizar cambios tan pronto como se los realice en la aplicación móvil o web. Por otro lado, para el sistema de conteo por visión artificial se implementó la raspberry pi 3 como sistema embebido para el procesamiento de video proveniente de la cámara, todo esto ha sido programado mediante Python. El contador al igual que el dispensador está conectado con Firebase, para enviar información sobre la rutina de alimentación calculada en función del número y biomasa de los alevines.
- La estructura del dispensador está en contacto directo con diversas condiciones climáticas debido a su ubicación en el borde del estanque de los alevines. Del mismo, el contador de peces tiene un generador de caída de agua para empujarlos desde la entrada a la salida. Por lo que, se concluyó que el material que mejor se acopla para ambos sistemas es el acero inoxidable tipo 304, debido a su tolerancia a la corrosión y resistencia.
- El dispensador tiene como fuente de alimentación principal un sistema fotovoltaico. Para optimizar el consumo de energía entra en el modo Deep Sleep desde las 6:00 pm hasta las 6:45 am del día siguiente, es decir, el sistema permanece en reposo durante 12 horas y 15 minutos. Por lo tanto, el piscicultor podrá definir sus horarios de alimentación desde las 7am. Además, la activación de los actuadores ocurre un minuto antes del proceso de alimentación y se apagan un minuto después de haber concluido con la rutina asignada. Para definir el porcentaje de precisión del sistema se han realizado 10 pruebas, en cada una se han pesado una cantidad de 100 gramos, dando

como resultado una precisión 98.7%. Del mismo para determinar el alcance del lanzamiento se han tomado 5 pruebas, en cada una de ellas se dispersó 200 gramos, dando un alcance óptimo de 252 centímetros. Por otra parte, el sistema de visión artificial implementa el algoritmo del centroide para asignar una ID única a cada alevín, con esto se consigue una precisión global del conteo del 96,61% y para la biomasa una precisión del 86,09%.

- La aplicación web y aplicación móvil fueron desarrolladas mediante JavaScript. Al implementar un solo lenguaje de programación, permitió acelerar el proceso de creación. Ambas aplicaciones cuentan con la autenticación mediante correo electrónico y contraseña, mismas que son validadas mediante el servicio de Firebase. En caso de que sea necesario reestablecer la contraseña para un usuario existente, desde la opción de restauración se enviará un mensaje hacia el correo electrónico registrado con los pasos a seguir para la restauración de la clave.

4.2 Recomendaciones

- Las estructuras de los sistemas están desarrolladas con acero inoxidable, pero al incluir componentes que no están acondicionados para exteriores como bisagras y soportes estas pueden llegar a oxidarse. Por lo tanto, con fines se mejora se puede cambiar esos componentes por otros como plásticos o del mismo material de las estructuras como el acero inoxidable.
- El material más costoso para el desarrollo del dispensador de alimento fue el acero inoxidable, haciendo que su producción no sea atractiva. Para reemplazar el material se podría utilizar componentes como el plástico, cuyo costo es menor en comparación el acero.
- Debido que la ración de alimentación está en función de la temperatura del agua. Se recomienda al piscicultor llevar un registro mensual de la temperatura del agua, con el fin de ajustar la cantidad de alimento necesaria para cada temporada.

Referencias Bibliográficas

[1] J. Ramos, «Sistema de visión artificial para el conteo y medición de alevinos de trucha “Arcoíris”, para la Dirección Subregional de la Producción Andahuaylas, 2018.», 2018.

[2] C. X. Ortiz Malusin, «Diseño e implementación de una máquina automática contadora de alevines para optimizar el tiempo y la fiabilidad de la producción para la empresa ACUIMAGG de la parroquia “Manuel Cornejo Astorga” en la provincia de Pichincha.», feb. 2018, Accedido: 12 de julio de 2022. [En línea]. Disponible en: http://repositorio.espe.edu.ec/jspui/handle/21000/14000

[3] J. Osorio, F. Vallejo, y L. Echeverri, «Alimentador automático para perros con plataforma IOT», *Rev. Univ. Católica Oriente*, vol. 31, n.º 45, Art. n.º 45, sep. 2020.

[4] «Sistema de detección y clasificación de peces utilizando visión computacional». https://repositorio.ulima.edu.pe/handle/20.500.12724/11174 (accedido 12 de julio de 2022).

[5] P. Karningsih, R. Kusumawardani, N. Syahroni, Y. Mulyadi, y A. Saad, «Automated fish feeding system for an offshore aquaculture unit - IOPscience». https://iopscience.iop.org/article/10.1088/1757-899X/1072/1/012073/meta (accedido 12 de julio de 2022).

[6] Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO), «MANUAL PRÁCTICO PARA EL CULTIVO DE LA TRUCHA ARCOIRIS». 2014. [En línea]. Disponible en: https://www.fao.org/3/bc354s/bc354s.pdf

[7] A. González *et al.*, *TÉCNICAS Y ALGORITMOS BÁSICOS DE VISIÓN ARTIFICIAL*. [En línea]. Disponible en: https://publicaciones.unirioja.es/catalogo/online/VisionArtificial.pdf

[8] Infaimon, *INFAIMON SU ASESOR EN VISIÓN ARTIFICIAL*.

[9] A. F. Luna Camacho y N. E. Rodríguez Menjura, «Detección y conteo de personas en espacios cerrados utilizando estrategias basadas en visión artificial», Pontificia Universidad Javeriana, Bógota, 2017. Accedido: 12 de julio de 2022. [En línea]. Disponible en: http://repository.javeriana.edu.co/handle/10554/38771

[10] J. Pérez Nasser, «Análisis de tráfico vehicular mediante visión artificial», Universidad Técnica de Ambato, Ambato, 2019. Accedido: 12 de julio de 2022. [En línea]. Disponible en: https://repositorio.uta.edu.ec:8443/jspui/handle/123456789/29182

[11] M. C. Tobajas, *Energía solar fotovoltaica*. Cano Pina, 2018. Accedido: 6 de julio de 2022. [En línea]. Disponible en: https://elibro.net/es/ereader/uta/45047

[12] O. P. Lamigueiro, «Energía Solar Fotovoltaica», p. 186.

[13] E. SA, «¿Qué es un regulador? - Regulador de carga solar - Enercity SA», *Enercity S.A.*, 1 de septiembre de 2019. https://enercitysa.com/blog/regulador-de-carga-solar-2/ (accedido 8 de julio de 2022).

[14] «¿Qué inversor solar necesito para mi instalación fotovoltaica?», *tarifasgasluz.com*. https://tarifasgasluz.com/autoconsumo/componentes/inversor-solar (accedido 8 de julio de 2022).

[15] «1.3.1.- Determinación de la Potencia Pico del subsistema generador. | ISF06.- Diseño, dimensionado y selección de componentes de instalaciones fotovoltaica...» https://ikastaroak.ulhi.net/edu/es/IEA/ISF/ISF06/es_IEA_ISF06_Contenidos/website_131_determinacin_de_la_potencia_pico_del_subsistema_generador.html (accedido 12 de julio de 2022).

[16] J. Z. Jiménez, *Aplicaciones web*. Macmillan Iberia, S.A., 2013. Accedido: 6 de julio de 2022. [En línea]. Disponible en: https://elibro.net/es/ereader/uta/43262

[17] Pedro Salcedo Lagos, «Evolucion de la Web desde la 1.0 a la 7.0 - Dr Pedro Salcedo», 12:07:33 UTC. Accedido: 6 de julio de 2022. [En línea]. Disponible en: https://es.slideshare.net/PedroLagos1/evolucion-de-la-web-desde-la-10-a-la-70-dr-pedro-salcedo

[18] «Funcionamiento y evolución de las aplicaciones web | Blog | Hosting Plus Perú», *Hosting Plus*, 27 de septiembre de 2021. https://www.hostingplus.pe/blog/funcionamiento-y-evolucion-de-las-aplicaciones-web/ (accedido 6 de julio de 2022).

[19] R. V. Lerma-Blasco, *Aplicaciones web*. McGraw-Hill Espana, 2013. [En línea]. Disponible en: https://elibro.net/es/lc/uta/titulos/50244

[20] «Evolución de Las Aplicaciones Web | PDF | Red mundial | Internet y web», *Scribd*. https://es.scribd.com/document/566530607/EVOLUCION-DE-LAS-APLICACIONES-WEB-1 (accedido 6 de julio de 2022).

[21] «2.1 Arquitectura de las aplicaciones Web», *Programacion Web*, 14 de noviembre de 2013. https://programacionwebisc.wordpress.com/2-1-arquitectura-de-las-aplicaciones-web/ (accedido 3 de julio de 2022).

[22] N. Rodrigo y M.-R. Ramón, «TRABAJO FINAL DE GRADO», p. 47.

[23] «Qué es el Backend de una web y por qué es tan importante», *Rafa Arjonilla*. https://rafarjonilla.com/que-es/backend/ (accedido 30 de junio de 2022).

[24] M. L. Castañeda, «QUÉ SON LAS APPS Y TIPOS DE APPS», p. 3.

[25] G. Puetate y J. L. Ibarra, *APLICACIONES MÓVILES HÍBRIDAS*, Primera Edición. Centro de publicaciones PUCE. [En línea]. Disponible en: https://www.pucesi.edu.ec/webs2/wp-content/uploads/2021/02/Aplicaciones-M%C3%B3viles-H%C3%ADbridas-2020.pdf

[26] S. Cirani, G. Ferrari, M. Picone, y L. Veltri, *Internet of Things: Architectures, Protocols and Standards*. Newark, UNITED KINGDOM: John Wiley & Sons, Incorporated, 2018. Accedido: 23 de junio de 2022. [En línea]. Disponible en: http://ebookcentral.proquest.com/lib/uta-ebooks/detail.action?docID=5502966

[27] «¿Qué es IoT (Internet Of Things)?», *Deloitte Spain*. https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html (accedido 24 de junio de 2022).

[28] «Las 4 etapas de la arquitectura del IoT». https://es.digi.com/blog/post/the-4-stages-of-iot-architecture (accedido 24 de junio de 2022).

[29] N. C. Marcet y G. M. Pinzón, «IMPLEMENTACIÓN Y EVALUACIÓN DE PLATAFORMAS EN LA NUBE PARA SERVICIOS DE IoT», p. 62.

[30] EDITORIAL, «Concepto de gateway y ejemplos», *Cursos gratis*, 16 de mayo de 2013. https://conocimientosweb.net/dcmt/ficha3303.html (accedido 25 de junio de 2022).

[31] L. Hernández, «CANAL DE TELEVISIÓN DIGITAL PARA LA UNIVERSIDAD TÉCNICA DE AMBATO», UNIVERSIDAD TÉCNICA DE AMBATO, Ambato, 2015. [En línea]. Disponible en: https://repositorio.uta.edu.ec/bitstream/123456789/13069/1/Tesis_t1048ec.pdf

[32] «Protocolos de comunicación en IoT que deberías conocer», *Barbara IoT*, 29 de abril de 2021. https://barbaraiot.com/blog/protocolos-iot-que-deberias-conocer/ (accedido 26 de junio de 2022).

[33] B. Barbecho y E. Andrés, «Diseño e implementación de una plataforma basada en IoT para la gestión de promociones de artículos en establecimientos comerciales», p. 77.

[34] A. Palomino, Ed., «Manual de cultivo de trucha arcoíris en jaulas flotantes». 2004.

[35] F. Charris, «Efectividad de cinco métodos de enumeración de alevines de Tilapia (Oreochromis spp.)», p. 23, 1998.

[36] Fondo Nacional de Desarrollo Pesquero - FONDEPES, *Manual de cultivo de truchas en ambientes convencionales*. Lima. [En línea]. Disponible en: https://cdn.www.gob.pe/uploads/document/file/2496894/Manual-de-Cultivo-de-Trucha.pdf

[37] H. Choquehuayta, «MANUAL DE CRIANZA DE TRUCHAS EN ESTANQUES Y LOMBRICULTURA». 2008. [En línea]. Disponible en: https://corporacionbiologica.info/wp-content/uploads/2021/05/M-DE-CRIA-DE-TRU-EN-ESTAN-Y-LOMBR.pdf

[38] L. Arregui, «El cultivo de la trucha arco iris (Oncorhynchus mykiss)», p. 103, 2013.

[39] I. Valarezo y G. Vizuete, «DISEÑO DE UNA DOSIFICADORA DE JABONES DE GLICERINA CON CAPACIDAD DE SETENTA Y DOS

UNIDADES POR MINUTO», Quito, 2016. [En línea]. Disponible en: https://bibdigital.epn.edu.ec/bitstream/15000/15228/1/CD-7003.pdf

[40] Ingemecánica, «Cálculo de Transportadores de Tornillo Sin Fin». https://ingemecanica.com/tutorialsemanal/tutorialn143.html (accedido 12 de julio de 2022).

[41] L. Márquez, «Curso de maquinaria agrícola - Equipos para el aporte de fertilizantes». [En línea]. Disponible en: https://www.mapa.gob.es/es/ministerio/servicios/informacion/05_1-1abonadoras_tcm30-483063.pdf

[42] H. Jara, «DISEÑO DE UN SISTEMA DE BOMBEO SOLAR DIRECTO PARA RIEGO POR GOTEO EN EL DISTRITO DE GUADALUPITO – LA LIBERTAD», Universidad Señor de Sipán, Pimentel, 2021. [En línea]. Disponible en: https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/9159/Jara%20Toro%2C%20Henry%20James.pdf?sequence=1&isAllowed=y

[43] «1.4.- Subsistema de acumulación. | ISF06.- Diseño, dimensionado y selección de componentes de instalaciones fotovoltaica...» https://ikastaroak.ulhi.net/edu/es/IEA/ISF/ISF06/es_IEA_ISF06_Contenidos/website_14_subsistema_de_acumulacin.html (accedido 12 de julio de 2022).

[44] Corporación para la Investigación Energética, «ATLAS SOLAR DEL ECUADORCON FINES DE GENERACIÓN ELÉCTRICA», p. 51, 2008.

[45] «1.5.1.- Dimensionado del subsistema regulador en un proyecto. | ISF06.- Diseño, dimensionado y selección de componentes de instalaciones fotovoltaica...» https://ikastaroak.ulhi.net/edu/es/IEA/ISF/ISF06/es_IEA_ISF06_Contenidos/website_151_dimensionado_del_subsistema_regulador_en_un_proyecto.html (accedido 12 de julio de 2022).

[46] O. Perpiñán, *Energía Solar Fotovoltaica*. 2020. [En línea]. Disponible en: https://oscarperpinan.github.io/esf/ESF.pdf

[47] I. Jáuregui, «Reconstrucción de objetos en 3D mediante un sistema estéreo binocular», Universidad Pública de Navarra, Pamplona, 2018. [En línea]. Disponible en: https://academica-

e.unavarra.es/xmlui/bitstream/handle/2454/32849/MemoriaTFG.pdf?sequence=1&isAllowed=y

[48] «OpenCV: Camera Calibration». https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (accedido 12 de julio de 2022).

[49] «Simple object tracking with OpenCV - PyImageSearch». https://pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/ (accedido 2 de septiembre de 2022).

[50] R. Krishnamurthi, A. Kumar, y S. S. Gill, *Autonomous and connected heavy vehicle technology*. 2022. Accedido: 12 de julio de 2022. [En línea]. Disponible en: https://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9780323907156

[51] D. M. D. Web, «¿Cómo Verificar Conexión a Base de Datos SOLIDWORKS Electrical?», *DMD*, 31 de marzo de 2017. https://dmd.com.mx/2017/03/31/como-verificar-conexion-a-base-de-datos-solidworks-electrical/ (accedido 13 de julio de 2022).

ANEXOS

Anexo A

Código del dispensador

```
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "ec.pool.ntp.org", -18000, 60000);
#include <math.h>

/*SENSOR DE PESO*/
#include <HX711_ADC.h>
#define DOUT  23
#define CLK  22
//HX711 constructor (dout pin, sck pin)
HX711_ADC LoadCell(DOUT, CLK);
long tpeso;
float obtener;

/*NEMA 23*/
#include <AccelStepper.h>
#define stepPin 13
#define motorInterfaceType 1
AccelStepper stepper = AccelStepper(motorInterfaceType, stepPin);
int pinDirection = 14;

/*SERVOMOTOR*/
#include <Servo.h>
Servo servoMotor;

/*DISTANCIA*/
#include "NewPing.h"
const int Trigger = 19;
const int Echo = 18;
NewPing sonar(Trigger, Echo);

/*DEEP SLEEP*/
#define uS_TO_S_FACTOR 1000000
#define TIME_TO_SLEEP  720        //En segundos
RTC_DATA_ATTR int bootCount = 0;


/*MEMORIA FLASH*/
#include <EEPROM.h>
#define EEPROM_SIZE 1


void setup() {
  EEPROM.begin(EEPROM_SIZE);

  if (EEPROM.read(0)==0||EEPROM.read(0)==207){
```

```
    ramdondato = 255;
  }else{
    ramdondato = EEPROM.read(0);
  }
 while(horario>=405 && horario <= 1080){
////////////////////////////////////////////////////////
    timeClient.update();
    HorasNew = timeClient.getHours();
    MinutesNew = timeClient.getMinutes();
    horario = (HorasNew*60) + MinutesNew;
    principal ();
  }

}
//////// MEDIR DISTANCIA/ALIMENTO TOLVA//////////////////
double medirDistancia(){
  float distance = sonar.ping_median(5);
  float valores = distance/59;
  return valores;
}
////////////////////////// MEDIR PESO/////////////////////

double medirPeso(){
  LoadCell.update();
 if (millis() > tpeso + 250) {
    obtener = fabsf(LoadCell.getData());
    tpeso = millis();
  }
  return obtener;
}
////////////////////////PROGRAMA PRINCIPAL//////////////
void principal(){

  timeClient.update();
  stringhora = String(timeClient.getHours());
  stringminutos = String(timeClient.getMinutes());
  stringTiempoAlimentar = stringhora + ":" + stringminutos;
  distancia = medirDistancia();
  nivelTolva = map(distancia, 0, 40, 100, 10);

  peso = medirPeso();

  compararSumaTiempo = intHora + intMinutos;

  if (compararTiempo == 0 || compararTiempo == compararSumaTiempo){
    if (millis() - compararTiming > 5000){
      compararTiming = millis();
      compararTiempo = compararSumaTiempo;

    }
 }else{
        if (millis() - compararTiming > 5000){
          compararTiming = millis();
          compararTiempo = 0;

        }
      intVueltas = 0;
      intTiempoTotal = 0;
      sumaIntervalos = 0;
  }
```

```
////////////////ENCENDIDO ACTUADORES AL SIGUIENTE DIA/////
intTiempoTotalAcuadores = intHora + intMinutos - 1;
finalHoraActuadores = intTiempoTotalAcuadores/60;
finalauxiliarActuadores = intTiempoTotalAcuadores/60;
finalMinutosActuadores = (round((finalauxiliarActuadores-
finalHoraActuadores)*60));

  if (habilitarActuadores == true){
    Serial.println("Los actuadores estan encendidos");
    digitalWrite(rele, HIGH);
  }

  if (((finalHoraActuadores) == timeClient.getHours()) &&
(finalMinutosActuadores == timeClient.getMinutes())){
    Serial.println("Los actuadores estan encendidos");
    habilitarActuadores = true; //HABILITA ACTUADORES
    cicloActuadores = false; // APAGAA ACTUADORES
    repeticionActuadores = true; // APAGA ACTUADORES
  }

////////////////////////////////////////////////////////
intTiempoTotalCiclo = (finalHora*60) + finalMinutos - 1;
finalHoraCiclo = intTiempoTotalCiclo/60;
finalauxiliarCiclo = intTiempoTotalCiclo/60;
finalMinutosCiclo = (round((finalauxiliarCiclo-finalHoraCiclo)*60));

  if (cicloActuadores == true){
     repeticionActuadores = false;
     habilitarActuadores = false;
     actuadoresHorario =false;
    }

  if (repeticionActuadores == false){
    delay(2000);
    Serial.println("Se ha apagado los actuadores");
    digitalWrite(rele, LOW);
  }else{
    //Serial.println("Los actuadores estan encendidos");
    //digitalWrite(rele, HIGH);
  }

  if(actuadoresHorario == true){
    Serial.println("Se ha encendido los actuadores");
    digitalWrite(rele, HIGH);
  }

  if (((finalHoraCiclo) == timeClient.getHours()) &&
(finalMinutosCiclo == timeClient.getMinutes())){
    Serial.println("Los actuadores estan encendidos");
    repeticionActuadores = true; // ACTIVA ACTUADORES EN TRUE
    cicloActuadores = false; //APAGA EN TRUE
    actuadoresHorario =true; //HABILITA ACTUADORES EN TRUE
  }
```

Anexo B

Código del contador

```
import math
from RPLCD import *
from RPLCD.i2c import CharLCD
import imutils
from imutils.video import VideoStream
from imutils.video import FPS
from imutils.video import FileVideoStream
from Rastreador.rastreador_centroide import RastreadorCentroide
from Alevines.objetoAlevin import ObjetoAlevin
from CalculoRacion.calculo_racion import Racion
import numpy as np
import cv2
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
import datetime
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(21, GPIO.IN)
GPIO.setup(16, GPIO.IN)
#############################################################################
lcd = CharLCD('PCF8574', 0x27)
lcd = CharLCD(i2c_expander='PCF8574', address=0x27, port=1,
              cols=20, rows=4, dotsize=8,
              charmap='A02',
              auto_linebreaks=True,
              backlight_enabled=True)
#############################################################################
fecha
hora = ""
sumaid = 0
#############################################################################
ob = RastreadorCentroide()
cap = VideoStream(usePiCamera=1 > 0).start()
time.sleep(2.0)
prev_frame_time = 0
new_frame_time = 0
fgbg = cv2.bgsegm.createBackgroundSubtractorMOG()
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
AlevinRastreado = {}
ContadorAlevines = 0
masa = 0
ss = 0
```

```
pesos_alevines = [ ]
tallas_alevines = [ ]
tallas_promedios = 0
#############################################################################
firebase_credenciales = credentials.Certificate('react-native.json')
firebase_admin.initialize_app(firebase_credenciales,{})
ref = db.reference()
#############################################################################
distCoeffs = np.array([[0.27837767,  -1.36213724,
0.00644272,  0.00773746, 1.65877368]])

cameraMatrix = np.array([[595.57348633,   0,         324.5738219],
                         [  0,         594.17730713, 239.79392217],
                         [  0,           0,           1        ]])
#############################################################################
while True:
    fecha = "%s/%s/%s" %(e.day, e.month, e.year)
    hora = "%s:%s:%s" %(e.hour, e.minute, e.second)
    sumaid = e.hour + e.minute + e.second
    lcd.cursor_pos = (0, 0)
    lcd.write_string('Cantidad: ' + str(ContadorAlevines))
    lcd.cursor_pos = (1, 0)
    lcd.write_string('Biomasa: ')
    lcd.cursor_pos = (1, 9)
    lcd.write_string(str(round(masa,2)))
    pulsador = GPIO.input(21)
    reiniciar = GPIO.input(16)
    corregido = cap.read()
    frame = cv2.undistort(corregido, cameraMatrix, distCoeffs)
    frame = cv2.undistort(corregido, cameraMatrix, distCoeffs, None)
    grises = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rectangulos = []
    area_pts = np.array([[0,0], [0,320],[320,320], [320,0]], np.int32)
    imagen_negra = np.zeros(shape=(frame.shape[:2]), dtype=np.uint8)
    imagen_negra = cv2.drawContours(imagen_negra, [area_pts], -1,
(255), -1)
    imagen_are_interes = cv2.bitwise_and(grises, grises,
mask=imagen_negra)
    bgmask = fgbg.apply(imagen_are_interes)
    erosion =  cv2.erode(bgmask, kernel, iterations=2)
    dilatacion = cv2.dilate(erosion, kernel, iterations=11)
    cnts = cv2.findContours(dilatacion, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]
    for i in cnts:
        if cv2.contourArea(i) > 1000:
            x,y,w,h = cv2.boundingRect(i)
            endX = x+w
            endY = y+h
```

```
            rectangulos.append([x,y,endX,endY])
            #print(rectangulos)
            cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0),2)
            tamaño = w
            formateado = "{0:.2f}".format(tamaño)
            cv2.putText(frame, str(formateado), (x, y-5),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,255,255), 2)
            cv2.line(frame, (x,y), (x+w,y), (255,0,0),1)
##########################################################################
    objetos = ob.actualizar(rectangulos)
    for (objectID, centroid) in objetos.items():
##########################################################################
        # Comprueba si hay un alevin rastreado asociado a un ID
        to = AlevinRastreado.get(objectID, None)
        # Si no hay un alevin rastreado, se registra
        if to is None:
            to = ObjetoAlevin(objectID)
        # Si hay alevines rastrados se sigue con el conteo
        else:
            # Verifica si el objeto ha sido contado
            if not to.contado:
                if 150< centroid[0][1] < 220:
                    cv2.line(frame, (0,150), (320,150), (0,255,0), 4)
                    to.contado = True
                    ContadorAlevines += 1
##########################################################################
                    t = np.double(centroid[1]/25)
                    regresion = 0.2203*math.pow(t,2)-1.1415*t+1.7506
                    masa += regresion
                    pesos_alevines.append(regresion)
                    pesos_promedios = np.mean(pesos_alevines)
                    tallas_alevines.append(t)
                    tallas_promedios = np.mean(tallas_alevines)
                    ss = round(tallas_promedios,0)
##########################################################################
        # Almacena el objeto rastreado el diccionario
        AlevinRastreado[objectID] = to
        calculo = Racion(masa,ss,ContadorAlevines)
################Pulsador Para enviar datos a Firebase##################
    if pulsador == 1:
        ref = db.reference('boton/estado')
        ref.update({"booleano":False})
        ref = db.reference('Base de Datos-
Automatico/{}'.format(sumaid))
        ref.update({"descripcion":"El sistema se ejecutará a las {}:{}
suministrando una cantidad de {} gramos cada {}:{} hora durante {}
vez/ces al día".format(calculo.horas,calculo.minutoss,
```

```
round(calculo.raciones,2),calculo.intervaloHora,calculo.intervaloMinuto
s,calculo.repeticiones),
                    "fechaactual":fecha,
                    "horaactual":hora,
                    "usuario":"Automatico"
                    })
        time.sleep(.3)
        lcd.clear()
        lcd.cursor_pos = (0, 0)
        lcd.write_string('Datos Enviados')
        while reiniciar == 0:
            reiniciar = GPIO.input(16)
    time.sleep(.01)
################Pulsador Para Reiniciar Contador#######################

    if reiniciar == 1:
        lcd.clear()
        lcd.cursor_pos = (0, 0)
        lcd.write_string('Reiniciando.....')
        AlevinRastreado = {}
        ContadorAlevines = 0
        masa = 0
        ss = 0
        pesos_alevines = [ ]
        tallas_alevines = [ ]
        tallas_promedios = 0
        time.sleep(.3)
        lcd.clear()
    time.sleep(.01)
#########################################################################
    k = cv2.waitKey(20) & 0xFF
    if k == 27:
        break
cap.release()
cap.destroyAllWindows()
```

Anexo C

Código de la aplicación móvil

Código de la ventana 'Login'

```
import React, { useState, useContext } from 'react';
import { View, Text, StyleSheet, TouchableOpacity, ScrollView, Alert,
Image } from 'react-native';
import FormButton from '../components/FormButton';
import FormInput from '../components/FormInput';

export default function LoginScreen({ navigation }) {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const { login } = useContext(AuthContext);
  const handleSubmit =  () => {
    if(email == '' && password ==''){
    Alert.alert('Introduce tu correo electrónico y tu contraseña')
    }else
      if(email == ''){
        Alert.alert('Introduce  el correo electrónico')
    }else
      if(password == ''){
        Alert.alert('Introduce  tu contraseña')
      }else
      login(email, password)
     }
  return (
<ScrollView showsVerticalScrollIndicator={false}>
    <View style={styles.root}>
      <Text style={{letterSpacing: 2.43}}></Text>
      <Text style={{letterSpacing: 2.43}}></Text>
      <Image style={{ width: 50, height: 50 }}
source={require('../images/cand.png')} />
      <Text style={styles.title}>Iniciar sesión</Text>
      <Text style={{letterSpacing: 2.43}}></Text>
      <FormInput
        value={email}
        placeholderText='Correo electrónico'
        onChangeText={userEmail => setEmail(userEmail)}
        autoCapitalize='none'
        keyboardType='email-address'
        autoCorrect={false}
      />
      <FormInput
        value={password}
        placeholderText='Contraseña'
```

```
        onChangeText={userPassword => setPassword(userPassword)}
        secureTextEntry={true}
      />
        <Text style={{letterSpacing: 2.43}}></Text>
      <FormButton buttonTitle='Iniciar sesión' onPress={handleSubmit}
/>
      <View>
      <FormNavButton
    text="¿Olvidaste tu contraseña?"
    onPress={() => navigation.navigate('ForgotPassword')}
    type="TERTIARY"
      />
      <FormNavButton
    text="Crear cuenta nueva"
    onPress={() => navigation.navigate('Signup')}
    type="TERTIARY"
      />
    </View>
    </View>
</ScrollView>
  );
}
```

Código de la ventana 'Register'

```
import React, { useState, useContext } from 'react';
import { View, Text, StyleSheet, ScrollView, Alert, Image } from
'react-native';
import FormButton from '../components/FormButton';
import FormInput from '../components/FormInput';
import FormNavButton from '../components/FormNavButton';
import { AuthContext } from '../navigation/AuthProvider';
export default function SignupScreen({navigation}) {
 const [email, setEmail] = useState('');
 const [password, setPassword] = useState('');
 const {register} = useContext(AuthContext)
 const handleSubmit =  () => {
//register(email, password)
if(email == '' && password ==''){
Alert.alert('Introduce el correo electrónico y tu contraseña')
}else
  if(email == ''){
    Alert.alert('Introduce  el correo electrónico')
  }else
  if(password == ''){
    Alert.alert('Introduce  tu contraseña')
  }else
  register(email, password)
 }
```

```
 return (
  <ScrollView showsVerticalScrollIndicator={false}>
  <View style={styles.root}>
  <Text style={{letterSpacing: 2.43}}></Text>
  <Text style={{letterSpacing: 2.43}}></Text>
  <Image style={{ width: 50, height: 50 }}
source={require('../images/cand.png')} />
     <Text style={styles.title}>Registrarte</Text>
     <Text>Es rápido y fácil.</Text>
     <Text style={{letterSpacing: 2.43}}></Text>
      <FormInput
        value={email}
        placeholderText='Correo electrónico'
        onChangeText={userEmail => setEmail(userEmail)}
        autoCapitalize='none'
        keyboardType='email-address'
        autoCorrect={false}
      />
      <FormInput
        value={password}
        placeholderText='Contraseña'
        onChangeText={userPassword => setPassword(userPassword)}
        secureTextEntry={true}
      />
        <Text style={{letterSpacing: 2.43}}></Text>
      <FormButton
        buttonTitle='Registrarte'
        onPress={handleSubmit}
      />
        <Text style={{letterSpacing: 2.43}}></Text>
      <FormNavButton
    text="¿Ya tienes una cuenta? Iniciar sesión"
    onPress={() => navigation.navigate('Login')}
    type="TERTIARY"
      />
    </View>
    </ScrollView>
  );
}
```

Código de la ventana ‘forgot password’

```
import FormButton from '../components/FormButton';
import FormInput from '../components/FormInput';
import { AuthContext } from '../navigation/AuthProvider';
export default function ForgotPasswordScreen({ navigation }) {
const [email, setEmail] = useState('');
const {forgotpassword, logout} = useContext(AuthContext)
const handleSubmit =  () => {
```

```
  if(email == ''){
    Alert.alert('Introduce una dirección de correo electrónico')
  }else{
    forgotpassword(email) & navigation.navigate('Login')  }
   }
return (
<ScrollView showsVerticalScrollIndicator={false}>
<View style={styles.root}>
  <Text style={{letterSpacing: 2.43}}></Text>
  <Text style={{letterSpacing: 2.43}}></Text>
  <Image style={{ width: 50, height: 50 }}
source={require('../images/cand.png')} />
  <Text style={styles.title}>Restablece tu contraseña</Text>
      <FormInput
        value={email}
        placeholderText='Correo electrónico'
        onChangeText={userEmail => setEmail(userEmail)}
        autoCapitalize='none'
        keyboardType='email-address'
        autoCorrect={false}
      />
        <Text style={{letterSpacing: 2.43}}></Text>
      <FormButton
      buttonTitle='Restablecer contraseña'
      onPress={handleSubmit}
      />
    </View>
    </ScrollView>
  );
}
```

Código de la ventana ‘Home’

```
import FormNavButton from '../components/FormNavButton';
import { AuthContext } from '../navigation/AuthProvider';
import icon from '../images/icono.png';
import database from '@react-native-firebase/database' ;
import Boton from '../components/Boton';
import BooleanNavButton from '../components/BooleanNavButton';
export default function HomeScreen({ navigation }) {
  const {height} = useWindowDimensions();
  const { user, logout } = useContext(AuthContext);
  const [value, setValue] = useState('');
  const [booleano, setBooleano] = useState('');
  useEffect(() => {
    database().ref('/Tolva/Nivel/Value')
    .on('value', snapshot => {
      console.log('User data: ', snapshot.val());
      setValue(snapshot.val())
```

```
    });
  }, []);
  useEffect(() => {
    database().ref('/boton/estado/booleano')
    .on('value', snapshot => {
    console.log('User data: ', snapshot.val());
    setBooleano(snapshot.val())
    });
  }, []);
  const handleSubmit =  () => {

  if (booleano == false){
    Alert.alert('Al dar click en Guardar no tendra acceso al sistema
automatico')
    navigation.navigate('Manual')
       }else{
        navigation.navigate('Manual')
       }
      }
      const hAuto =  () => {
        navigation.navigate('Automatico')
              }
      const [stylea, setStylea] = useState("");
      useEffect(() => {
        if (booleano == true){
        setStylea("BotonH1");
      } else
      setStylea("BotonH");
      }, );
  return (
    <ScrollView showsVerticalScrollIndicator={false}>
    <View style={styles.root}>
    <Text style={{letterSpacing: 2.43}}></Text>
    <Text style={{letterSpacing: 2.43}}></Text>
    <Text style={{fontSize: 24, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>Bienvenido al Sistema de</Text>
    <Text style={{fontSize: 24, alignSelf:'center',  fontWeight:
'bold',   color: 'black',}}>Alimentación</Text>
    <Boton text="Historial de comida" onPress={() =>
navigation.navigate('Historialdecomida')} />
    <Text style={styles.tolva}> {value}% </Text>
    <Image
    source = {icon}
    style ={[styles.nico, {height: height * 0.3}]}
    resizeMode="contain"/>
    <Text style={{fontSize: 17, alignSelf:'center',  color: 'black',}}>
{user.email}</Text>
    <Text style={{letterSpacing: 2.43}}></Text>
```

```
    <Text style={{fontSize: 17, alignSelf:'center',  color: 'black',}}>
{user.uid}</Text>
    <Text style={{letterSpacing: 2.43}}></Text>
    <Text style={{fontSize: 24, alignSelf:'center',  fontWeight:
'bold',   color: 'black',}}>Eliga el tipo de control</Text>
      <View style={styles.alineado}>
      <BooleanNavButton
      text="Automático"
      onPress={hAuto}
      type={stylea}
      />
      <View style={{flex:0.1}}/>
      <FormNavButton
      text="Manual"
      onPress={handleSubmit}
      type={"BotonH"}
      />
      </View>
      <FormNavButton
      text="Salir"
      onPress={() => logout()}
      type="BotonS"
/>
    </View>
    </ScrollView>
  );
}
```

Código de la ventana ‘Manual’

```
import InputManual from '../components/InputManual';
import Puntos from '../components/Puntos';
import { AuthContext } from '../navigation/AuthProvider';
import database from '@react-native-firebase/database' ;
import SelectDropdown from 'react-native-select-dropdown'
import { color } from 'react-native-reanimated';
export default function ManualScreen({ navigation }) {
    const horass = ["00", "01", "02", "03", "04", "05", "06", "07",
"08", "09", "10", "11", "12","13", "14", "15", "16", "17", "18", "19",
"20", "21", "22", "23",]
  const minutoss = ["00", "01", "02", "03", "04", "05", "06", "07",
"08", "09", "10", "11", "12","13", "14", "15", "16", "17", "18", "19",
"20", "21", "22", "23","24", "25","26", "27", "28", "29", "30", "31",
"32", "33", "34", "35", "36", "37", "38","39", "40", "41", "42", "43",
"44", "45", "46", "47","48", "49", "50", "51","52","53","54","55","56",
"57","58", "59",]
  const [cantidad, setCantidad] = useState('');
  const [slt, setSlt] = useState('');
  const [hora, setHora] = useState('');
```

```
  const [minutos, setMinutos] = useState('');
  const [intervaloHora, setIntervaloHora] = useState('');
  const [intervaloMinutos, setIntervaloMinutos] = useState('');
  const [repeticiones, setRepeticiones] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const { user } = useContext(AuthContext);
  const [currentId, setCurrentId] = useState('')
  function create  ()  {
    database()
    .ref('/Datos RealTime-Manual/')
    .set({
      cantidad: cantidad,
      hora: hora,
      minutos: minutos,
      intervaloHora: intervaloHora,
      intervaloMinutos: intervaloMinutos,
      repeticiones: repeticiones,
      usuario: user.email,
      descripcion:"El sistema se ejecutará a la/s " +hora+ ":"
+minutos+ " suministrando una cantidad de " +cantidad+ " gramos, con
intervalos de " +intervaloHora+ ":" +intervaloMinutos+ " hora/s durante
" +repeticiones+ " vez/ces al día",
      horaactual: new Date().toLocaleTimeString(),
      fechaactual: new Date().toLocaleDateString(),

    })
    .then(() => console.log('Data set.'))
    .catch(error => {
      console.log(error)
    })
  };
  function create1  ()  {
    if (currentId == '')
    database()
    .ref('/Base de Datos-Manual/')
    .push({
      usuario: 'Manual',
      descripcion:"El sistema se ejecutará a la/s " +hora+ ":"
+minutos+ " suministrando una cantidad de " +cantidad+ " gramos, con
intervalos de " +intervaloHora+ ":" +intervaloMinutos+ " hora/s durante
" +repeticiones+ " vez/ces al día",
      horaactual: new Date().toLocaleTimeString(),
      fechaactual: new Date().toLocaleDateString(),
    })
    .then(() => console.log('base de datos creada.'))
    .catch(error => {
      console.log(error)
```

```
    })
  };
   const handleSubmit =  () => {
    if(cantidad == '' && hora ==''  &&minutos == '' && intervaloHora
==''  &&intervaloMinutos == '' && repeticiones ==''  ){
    Alert.alert('Todos los campos se encuentran vacíos')
  }else
      if(cantidad == ''){
        Alert.alert('introduce la cantidad de comida')
      }else
      if(hora == ''){
        Alert.alert('Introduce  la hora (numeros del 1 - 24/ las 24
horas es igual que 00)')
      }else
      if(minutos == ''){
        Alert.alert('Introduce  los minutos (numeros del 0 - 59)')
      }else
      if(intervaloHora == ''){
        Alert.alert('Introduce  la hora de los intervalos')
      }else
      if(intervaloMinutos == ''){
        Alert.alert('Introduce  los minutos de los intervalos')
      }else
      if(repeticiones == ''){
       Alert.alert('Introduce el numero de repeticiones que desas')
       }else
       {
        create();
        create1();
        estado();
       }
            }
       function estado  ()  {
              database()
              .ref('/boton/estado/')
              .set({
                booleano: true,
              })
              .then(() => console.log('Data set.'))
              .catch(error => {
                console.log(error)
              })
            };
  return (
<ScrollView showsVerticalScrollIndicator={false}>
<View style={styles.root}>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{letterSpacing: 2.43}}></Text>
```

```
<Text style={{fontSize: 24, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>Sistema de</Text>
<Text style={{fontSize: 24, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>alimentación Manual</Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 17, alignSelf:'center',  color: 'black',}}>
{user.email}</Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Boton text="Datos Almacenados" onPress={() =>
navigation.navigate('DatosM')} />
<Text style={{alignSelf:'center',fontWeight: 'bold',  color:
'black',}}>Comida</Text>
      <InputManual
      placeholder="Cantidad de comida en gr."
      value={cantidad}
      setValue={setCantidad}
      onChangeText = {(cantidad) => {setCantidad(cantidad)}}
      type={Number}
      />
<Text style={{alignSelf:'center',fontWeight: 'bold',  color:
'black',}}>Inicio</Text>
<View style={{flexDirection: 'row'}}>
<SelectDropdown
defaultButtonText="Inicio hora"
buttonTextStyle={{
  fontSize:15,
  textAlign: 'left',
}}
buttonStyle={{
backgroundColor:'white',
maxWidth:'48%',
borderRadius: 5,
borderColor:'red'
}}
rowStyle={{
backgroundColor: 'white',
width: '100%',
borderWidth: 1,
borderRadius: 4,
borderColor:'#D3D3D3',
borderBottomColor:'#D3D3D3',
}}
  data={horass}
  onSelect={(selectedItem) => {
    console.log(selectedItem)
    setHora(selectedItem)
  }}
  buttonTextAfterSelection={(selectedItem) => {
```

```
    return selectedItem
  }}
  rowTextForSelection={(item) => {
    return item
  }}
/>
<Text style={styles.puntos}>:</Text>
<SelectDropdown
defaultButtonText={ ('Inicio minutos')}
buttonTextStyle={{
  fontSize:15,
  textAlign: 'left',
}}
buttonStyle={{
backgroundColor:'white',
maxWidth:'48%',
borderRadius: 5,
}}
rowStyle={{
backgroundColor: 'white',
width: '100%',
borderWidth: 1,
borderRadius: 4,
borderColor:'#D3D3D3',
borderBottomColor:'#D3D3D3',
}}
  data={minutoss}
  onSelect={(selectedItem) => {
    console.log(selectedItem)
    setMinutos(selectedItem)
  }}
  buttonTextAfterSelection={(selectedItem) => {
    return selectedItem
  }}
  rowTextForSelection={(item) => {
    return item
  }}
/>
</View>
<Text style={{alignSelf:'center',fontWeight: 'bold',  color:
'black',}}>Intervalos</Text>
<View style={{flexDirection: 'row'}}>
<SelectDropdown
defaultButtonText={ ('Intervalos Hora')}
buttonTextStyle={{
  fontSize:15,
  textAlign: 'left',
}}
```

```
buttonStyle={{
backgroundColor:'white',
maxWidth:'48%',
borderRadius: 5,
borderColor:'red'
}}
rowStyle={{
backgroundColor: 'white',
width: '100%',
borderWidth: 1,
borderRadius: 4,
borderColor:'#D3D3D3',
borderBottomColor:'#D3D3D3',
}}
  data={horass}
  onSelect={(selectedItem) => {
    console.log(selectedItem)
    setIntervaloHora(selectedItem)
  }}
  buttonTextAfterSelection={(selectedItem) => {
    return selectedItem
  }}
  rowTextForSelection={(item) => {
    return item
  }}
/>
<Text style={styles.puntos}>:</Text>
<SelectDropdown
defaultButtonText={ ('Intervalos Minutos')}
buttonTextStyle={{
  fontSize:15,
  textAlign: 'left',
}}
buttonStyle={{
backgroundColor:'white',
maxWidth:'48%',
borderRadius: 5,
}}
rowStyle={{
backgroundColor: 'white',
width: '100%',
borderWidth: 1,
borderRadius: 4,
borderColor:'#D3D3D3',
borderBottomColor:'#D3D3D3',
}}
  data={minutoss}
  onSelect={(selectedItem) => {
```

```
    console.log(selectedItem)
    setIntervaloMinutos(selectedItem)
  }}
  buttonTextAfterSelection={(selectedItem) => {
    return selectedItem
  }}
  rowTextForSelection={(item) => {
    return item
  }}
/>
</View>
<Text style={{alignSelf:'center',fontWeight: 'bold',  color:
'black',}}>Repeticiones</Text>
<InputManual
placeholder="Repeticiones"
value={repeticiones}
setValue={setRepeticiones}
onChangeText = {(repeticiones) => {setRepeticiones(repeticiones)}}
/>
<View style={styles.alineado}>
<FormNavButton
      text="Guardar"
      onPress={ handleSubmit }
      type="BotonH"
      />
</View>
</View>
    </ScrollView>

  );
}
```

Código de la ventana ‘Automático’

```
import { View, Text, StyleSheet, ScrollView } from 'react-native';
import Boton from '../components/Boton';
import FormButton from '../components/FormButton';
import { AuthContext } from '../navigation/AuthProvider';
import database from '@react-native-firebase/database' ;
export default function AutomaticoScreen({ navigation }) {
const { user, logout } = useContext(AuthContext);
//biomasa
  const [biomasa, setBiomasa] = useState('');
  useEffect(() => {
  database().ref('/Datos RealTime-Automatico/sistema
automatico/biomasa')
  .on('value', snapshot => {
      console.log('biomasa: ', snapshot.val());
      setBiomasa(snapshot.val())
```

```
    });
  }, []);
//comida
const [comida, setComida] = useState('');
useEffect(() => {
    database().ref('/Datos RealTime-Automatico/sistema
automatico/comida')
    .on('value', snapshot => {
      console.log('comida: ', snapshot.val());
      setComida(snapshot.val())
    });
  },[])
//intervaloHora
const [intervaloHora, setIntervaloHora] = useState('');
useEffect(() => {
    database().ref('/Datos RealTime-Automatico/sistema
automatico/intervaloHora')
    .on('value', snapshot => {
      console.log('Hora intervalo: ', snapshot.val());
      setIntervaloHora(snapshot.val())
    });
  },[])
//intervalominutos
const [intervaloMinutos, setIntervaloMinutos] = useState('');
useEffect(() => {
    database().ref('/Datos RealTime-Automatico/sistema
automatico/intervaloMinutos')
    .on('value', snapshot => {
      console.log('Hora intervalo: ', snapshot.val());
      setIntervaloMinutos(snapshot.val())
  });
},[])
//hora
const [hora, setHora] = useState('');
useEffect(() => {
    database().ref('/Datos RealTime-Automatico/sistema
automatico/hora')
    .on('value', snapshot => {
      console.log('Hora intervalo: ', snapshot.val());
      setHora(snapshot.val())
    });
  },[])


//minutos
const [minutos, setMinutos] = useState('');
useEffect(() => {
```

```
    database().ref('/Datos RealTime-Automatico/sistema
automatico/minutos')
    .on('value', snapshot => {
      console.log('Hora intervalo: ', snapshot.val());
      setMinutos(snapshot.val())
    });
  },[])
//repeticiones
const [repeticiones, setRepeticiones] = useState('');
useEffect(() => {
    database().ref('/Datos RealTime-Automatico/sistema
automatico/repeticiones')
    .on('value', snapshot => {
      console.log('Hora intervalo: ', snapshot.val());
      setRepeticiones(snapshot.val())
    });
  },[])
const [peces, setPeces] = useState('');
useEffect(() => {
   database().ref('/Datos RealTime-Automatico/sistema
automatico/peces')
    .on('value', snapshot => {
      console.log('Hora intervalo: ', snapshot.val());
      setPeces(snapshot.val())
    });
  },[])
  return (
<ScrollView showsVerticalScrollIndicator={false}>
<View style={styles.root}>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 24, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>Sistema de</Text>
<Text style={{fontSize: 24, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>alimentación Automático</Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 17, alignSelf:'center',  color: 'black',}}>
{user.email}</Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Boton text="Datos Almacenados" onPress={() =>
navigation.navigate('DatosA')} />
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 17, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>Biomasa de los alevines</Text>
<Text style={{fontSize:14}}> {biomasa} gramos</Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 17, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>Número de alevines</Text>
```

```
<Text style={{fontSize:14}}> {peces} alevines </Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 17, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>Cantidad de comida</Text>
<Text style={{fontSize:14}}> {comida} gramos</Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 17, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>Hora de inicio</Text>
<Text style={{fontSize:14}}> {hora} : {minutos}</Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 17, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>Intervalos</Text>
<Text style={{fontSize:14}}>
{intervaloHora} : {intervaloMinutos}
</Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 17, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>Repeticiones</Text>
<Text style={{fontSize:14}}> {repeticiones} veces al día </Text>
    </View>
</ScrollView>
  );
}
```

Código de la ventana 'Historial de comida'

```
import React, { useEffect, useState, useContext } from "react";
import {
  View,
  Text,
  SafeAreaView,
  StyleSheet,
  ScrollView
} from "react-native";
import { DataTable } from 'react-native-paper';
import database from '@react-native-firebase/database' ;
import { AuthContext } from "../navigation/AuthProvider";
const Historialdecomida = () => {
  const { user, logout } = useContext(AuthContext);
  //comida
const [comida, setComida] = useState({});
useEffect(() => {
    database().ref('Base de Datos-Comida')
    .on('value', snapshot => {
      console.log(snapshot.val())
      if (snapshot.val() != null)
      setComida({
          ...snapshot.val()
      })
```

```
    });
  },[])
  return (
<ScrollView showsVerticalScrollIndicator={false}>
<View style={styles.root}>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 24, alignSelf:'center',fontWeight:
'bold',  color: 'black',}}>Historial de comida</Text>
<Text style={{letterSpacing: 2.43}}></Text>
<Text style={{fontSize: 17, alignSelf:'center',  color: 'black',}}>
{user.email}</Text>
        <DataTable>
        {
             Object.keys(comida).map(id => {
             return <DataTable.Row key={id}>
             <ScrollView horizontal={true}>
             <DataTable.Cell >{comida[id].comida}</DataTable.Cell >
             </ScrollView>
        </DataTable.Row>
                                          })
        }
        </DataTable>
      </View>
      </ScrollView>
  );
};
```

Anexo D

Código de la página web

Código de la ventana 'login

```
import Button from '@mui/material/Button';
import CssBaseline from '@mui/material/CssBaseline';
import TextField from '@mui/material/TextField';
import Paper from '@mui/material/Paper';
import Box from '@mui/material/Box';
import Grid from '@mui/material/Grid';
import LockOutlinedIcon from '@mui/icons-material/LockOutlined';
import Typography from '@mui/material/Typography';
import { createTheme, ThemeProvider } from '@mui/material/styles';
const theme = createTheme();
export const Login = () => {
  const { login } = useAuth();
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);
  const history = useHistory();
  console.log(loading);
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const handleEmail = e => setEmail(e.target.value);
  const handlePassword = e => setPassword(e.target.value);
  const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true);
    try {
      await login(email, password);
      setLoading(false);
      history.push('/');
    } catch (error) {
      setLoading(false);
      setError('Introduce tu correo electrónico y tu contraseña');
      setTimeout(() => setError(''), 1500);
    }
  }
  return (
  <ThemeProvider theme={theme}>
    <Grid container component="main" sx={{ height: '100vh' }}>
      <CssBaseline />
        <Grid
        item
        xs={false}
        sm={4}
        md={7}
```

```
        sx={{
        backgroundImage: 'url(fish.jpg)',
        backgroundRepeat: 'no-repeat',
        backgroundColor: (t) =>
        t.palette.mode === 'light' ? t.palette.grey[50] :
t.palette.grey[900],
        backgroundSize: 'cover',
        backgroundPosition: 'center',
        }}
        />
        <Grid item xs={12} sm={8} md={5} component={Paper}
elevation={6} square>
        <Box
        sx={{
        my: 8,
        mx: 4,
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'center',
           }}
        >
        <Avatar sx={{ m: 1, bgcolor: 'secondary.main' }}>
        <LockOutlinedIcon />
        </Avatar>
        <Typography component="h1" variant="h5">
        Iniciar sesión
        </Typography>
        <div>
        <div>
        {error && <p className='error' >{error}</p>}
        </div>
        </div>
        <Box component="form" noValidate onSubmit={handleSubmit} sx={{
mt: 1 }}>
        <TextField
                margin="normal"
                required
                fullWidth
                id="email"
                label="Correo electrónico"
                name="email"
                autoComplete="email"
                autoFocus
                onChange={handleEmail}
              />
        <TextField
                margin="normal"
                required
```

```
                fullWidth
                name="password"
                label="Contraseña"
                type="password"
                id="password"
                autoComplete="current-password"
                onChange={handlePassword}
              />
        <Button
                type="submit"
                fullWidth
                variant="contained"
                sx={{ mt: 3, mb: 2 }}
              >
                Iniciar sesión
        </Button>
        <Grid container>
                <Grid item xs>
                <Link to='/forgot-password' variant="body2">
                ¿Olvidaste tu contraseña?
                </Link>
                </Grid>
                <Grid item>
                <Link to='/signup' variant="body2">
                Crear cuenta nueva
                </Link>
                </Grid>
        </Grid>
        </Box>
        </Box>
        </Grid>
        </Grid>
  </ThemeProvider>
  )
}
```

Código de la ventana 'Register'

```
import Avatar from '@mui/material/Avatar';
import Button from '@mui/material/Button';
import CssBaseline from '@mui/material/CssBaseline';
import TextField from '@mui/material/TextField';
import Grid from '@mui/material/Grid';
import Box from '@mui/material/Box';
import LockOutlinedIcon from '@mui/icons-material/LockOutlined';
import Typography from '@mui/material/Typography';
import Container from '@mui/material/Container';
import { createTheme, ThemeProvider } from '@mui/material/styles';
const theme = createTheme();
```

```
export const SignUp = () => {
   const { signup } = useAuth();
   const [error, setError] = useState('');
   const [loading, setLoading] = useState(false);
   const history = useHistory();
   const [email, setEmail] = useState('');
   const [password, setPassword] = useState('');
   const [confirmPassword, setConfirmPassword] = useState('');
   const handleEmail = e => setEmail(e.target.value);
   const handlePassword = e => setPassword(e.target.value);
   const handleConfirmPassword = e =>
setConfirmPassword(e.target.value);
   const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true);
    if(password !== confirmPassword){
      setError('Las contraseñas no coinciden');
      setTimeout(() => setError(''), 1500);
    }else{
      try{
        await signup(email, password);
        history.push('/');
      }catch(error){
        setError('Introduce una dirección de correo electrónico');
        setTimeout(() => setError(''), 1500);
      }
         }
        setLoading(false);
                                      }
  return (
    <ThemeProvider theme={theme}>
      <Container component="main" maxWidth="xs">
        <CssBaseline />
          <Box
            sx={{
              marginTop: 8,
              display: 'flex',
              flexDirection: 'column',
              alignItems: 'center',
            }}
          >
            <Avatar sx={{ m: 1, bgcolor: 'secondary.main' }}>
              <LockOutlinedIcon />
            </Avatar>
            <Typography component="h1" variant="h5">
              Registro
            </Typography>
```

```
            <Typography component="h1" variant="h5">
            {error && <p className='error' >{error}</p>}
            </Typography>
              <Box autoComplete='off' component="form" noValidate
onSubmit={handleSubmit} sx={{ mt: 3 }}>
              <Grid container spacing={2}>
              <Grid item xs={12}>
                <TextField
                required
                fullWidth
                id="email"
                label="Dirección de email"
                name="email"
                onChange={handleEmail}
                />
              </Grid>
              <Grid item xs={12}>
                <TextField
                required
                fullWidth
                name="password"
                label="Contraseña"
                type="password"
                id="password"
                onChange={handlePassword}
                />
              </Grid>
              <Grid item xs={12}>
                <TextField
                  required
                  fullWidth
                  name="password"
                  label="Repetir Contraseña"
                  type="password"
                  id="password"
                  onChange={handleConfirmPassword}
              />
              </Grid>
              </Grid>
              <Button
                  type="submit"
                  fullWidth
                  variant="contained"
                  sx={{ mt: 3, mb: 2, }}
                >
                  Registrarte
              </Button>
```

```
                <Box
                  sx={{
                    my:4,
                    mx: 24
                  }}
                >
                  {loading && <img src={Spinner} alt='Loading' />}
                </Box>
                <Box
                  sx={{
                    my:4,
                    mx: 5
                  }}
                >
                  <Grid container >
                    <Grid item xs={12} sm={6}>
                        <Typography component="h10" variant="h10">¿Ya
tienes una cuenta?</Typography>
                    </Grid>
                    <Grid item xs={12} sm={6}>
                        <Link to='/login'>Iniciar sesión</Link>
                    </Grid>
                  </Grid>
                </Box>
            </Box>
          </Box>
        </Container>
      </ThemeProvider>
    )
}
```

Código de la ventana ‘forgost pssword’

```
import Avatar from '@mui/material/Avatar';
import CssBaseline from '@mui/material/CssBaseline';
import Box from '@mui/material/Box';
import LockOutlinedIcon from '@mui/icons-material/LockOutlined';
import Typography from '@mui/material/Typography';
import Container from '@mui/material/Container';
import { createTheme, ThemeProvider } from '@mui/material/styles';
const theme = createTheme();
export default function ForgotPassword() {
const email2Ref = useRef()
const { resetPassword1 } = useAuth()
const [error, setError] = useState('')
const [loading, setLoading] = useState(false)
const [message, setMessage] = useState('')
const history = useHistory();
console.log(message);
```

```
async function handleSubmit(e) {
e.preventDefault();
try {
  setMessage('Verifica tu bandeja de entrada y sigue las
instrucciones')
  setMessage('')
  setError('')
  setLoading(true)
  history.push('/');
  await resetPassword1(email2Ref.current.value)
  setMessage('Verifica tu bandeja de entrada y sigue las
instrucciones')
    }catch{
      setError('Falló al restaurar tu contraseña')
          }
      setLoading(false)
                                  }
  return (
    <ThemeProvider theme={theme}>
      <Container component="main" maxWidth="xs">
      <CssBaseline />
      <Box
      sx={{
      marginTop: 8,
      display: 'flex',
      flexDirection: 'column',
      alignItems: 'center',
         }}
      >
      <Avatar sx={{ m: 1, bgcolor: 'secondary.main' }}>
      <LockOutlinedIcon />
      </Avatar>
      <Typography component="h1" variant="h5">
      {error && <p className='error' >{error}</p>}
      </Typography>
      <Typography component="h1" variant="h5">
      Restablece tu contraseña
      </Typography>
      <Box component="form" onSubmit={handleSubmit} noValidate sx={{
mt: 1 }}>
      <form onSubmit={handleSubmit}>
      <Box
      sx={{mx:9}}
      >
      <input style={{fontFamily: 'Arial',}} type='email'
placeholder='Correo electrónico' autoFocus required ref={email2Ref} />
      </Box>
      <Box
```

```
      sx={{mx:9}}
      >
      <input style={{fontFamily: 'Arial',}} type='submit'
disabled={loading} value='Restablecer contraseña' />
      </Box>
      </form>
      <Box
      sx={{mx:20, mt: 2}}
      >
      <Link to='/login' style={{fontFamily: 'Arial',}}>Regresar</Link>
      </Box>
      </Box>
      </Box>
      </Container>
      </ThemeProvider>
  )
}
```

Código de la ventana 'Manual'

```
import Button from '@mui/material/Button';
import CssBaseline from '@mui/material/CssBaseline';
import Paper from '@mui/material/Paper';
import Box from '@mui/material/Box';
import Grid from '@mui/material/Grid';
import Typography from '@mui/material/Typography';
import { createTheme, ThemeProvider } from '@mui/material/styles';
const theme = createTheme();
export const Manual = () => {
  //ACTUALIZAR DATOS DESDE FIREBASE
  const deshabilitar = () => {
    firebaseDb.child('boton/estado').update({
      booleano: true
    })
  }
  const { currentUser } = useAuth();
  const initialFieldValues = {
  usuario: '',
  cantidad: '',
  hora: '',
  minutos: '',
  intervaloHora: '',
  intervaloMinutos: '',
  repeticiones: '',
  fechaactual: '',
  descripcion: '',
  horaactual: ''
  }
  const [values, setValues] = useState(initialFieldValues)
```

```
const [currentId, setCurrentId] = useState('')
const handleInputChange = e => {
const { name, value } = e.target
setValues({...values,[name]: value})
                                  }
const handleFormSubmit =  (e) => {
e.preventDefault();
addOrEdit(values)
addOrEdit1(values)
                                    }
const addOrEdit = obj => {
if (currentId == 'ss'){
}else{
    firebaseDb.child(`Datos RealTime-Manual/${currentId}`)
    .set(
    obj,
    err => {
    if (err)
    console.log(err)
    else
    setCurrentId('')
     })
     }}
const addOrEdit1 = obj => {
if (currentId == ''){
    firebaseDb.child('Base de Datos-Manual')
    .push(
    obj,
    err => {
    if (err)
    console.log(err)
    else
    setCurrentId('')
     }) }
         }

const history = useHistory();
const [error, setError] = useState('');
const hAtras = async () => {
      try {
        history.push('/');
      }catch(error) {
        setError('Server Error')
      }
    }
const hDatosM = async () => {
      try {
        history.push('/DatosM');
```

```
        }catch(error) {
          setError('Server Error')
        }
      }
  return (
    <ThemeProvider theme={theme}>
      <Grid container component="main" sx={{ height: '100vh' }}>
        <CssBaseline />
          <Grid
            item
            xs={false}
            sm={4}
            md={7}
            sx={{
              backgroundImage: 'url(fish.jpg)',
              backgroundRepeat: 'no-repeat',
              backgroundColor: (t) =>
                t.palette.mode === 'light' ? t.palette.grey[50] :
t.palette.grey[900],
              backgroundSize: 'cover',
              backgroundPosition: 'center',
            }}
          />
          <Grid item xs={12} sm={8} md={5} component={Paper}
elevation={6} square>
          <Box
            sx={{
              my: 8,
              mx: 4,
              display: 'flex',
              flexDirection: 'column',
              alignItems: 'center',
            }}
          >

          <Typography align='center' component="h1" variant="h4">
              Sistema de Alimentación Manual
          </Typography>

          <Button
                onClick={hDatosM}
                variant="contained"
                sx={{ mt: 3, mb: 2 }}
              >
                Registro de Actividades
          </Button>

    <form autoComplete="off" onSubmit={handleFormSubmit}>
```

```
        <Typography align = "center" component="h5" variant="inherit">
            Cantidad de Alimento (gramos)
        </Typography>

            <input
            name="cantidad"
            type="number"
            step="0.01"
            placeholder="Cantidad de comida"
            value={values.cantidad}
            onChange={handleInputChange}
            style={{height:30}}
            />

        <Typography align = "center" component="h5" variant="inherit">
            Inicio (HH:MM)
        </Typography>

        <div class="row" >
          <div class="col-xs-12 col-md-3 input-group input-group-sm" >
            <input
            type="number"
            class="form-control"
            step="1"
            placeholder="Inicio hora"
            name="hora"
            value={values.hora}
            onChange={handleInputChange}
            style={{height:30}}
            />
            <span class="input-group-addon" style={{padding: "0px
10px"}}> : </span>
            <input
            type="number"
            class="form-control"
            name="minutos"
            step="1"
            placeholder="Inicio Minutos"
            value={values.minutos}
            onChange={handleInputChange}
            style={{height:30}}
            />
          </div>
        </div>
        <Typography align = "center" component="h5" variant="inherit">
             Intervalos (HH:MM)
        </Typography>
```

```
        <div class="row" >
          <div class="col-xs-12 col-md-3 input-group input-group-sm" >
           <input
           type="number"
           class="form-control"
           step="1"
           placeholder="Intervalos hora"
           name="intervaloHora"
           value={values.intervaloHora}
           onChange={handleInputChange}
           style={{height:30}}
           />
           <span class="input-group-addon" style={{padding: "0px
10px"}}>:</span>
           <input
           type="number"
           class="form-control"
           name="intervaloMinutos"
           step="1"
           placeholder="Intervalos minutos"
           value={values.intervaloMinutos}
           onChange={handleInputChange}
           style={{height:30}}
           />
          </div>
        </div>
        <Typography align = "center" component="h5" variant="inherit">
           Repeticiones
       </Typography>
           <input name="repeticiones" type="number" step="1"
placeholder="Repeticiones"
           value={values.repeticiones}
           onChange={handleInputChange}
           style={{height:30}}
           />
        <br />
        <Box
         sx={{py:0, mx: 18}}
       >
        <button className='btn-primary'
        usuario={values.usuario = 'Manual'}
        value={values.fechaactual = new Date().toLocaleDateString()}
        hora={values.horaactual =new Date().toLocaleTimeString()}
        descripcion={values.descripcion = ("El sistema se ejecutará a
la/s " +values.hora+ ":" +values.minutos+ " suministrando una cantidad
de " +values.cantidad+ " gramos, con intervalos de "
```

```
+values.intervaloHora+ ":" +values.intervaloMinutos+ " hora/s durante "
+values.repeticiones+ " vez/ces al día")} onClick={deshabilitar}
        >Guardar</button>
        </Box>
    </form >
        <Button
        color='warning'
        onClick={hAtras}
        variant="contained"
        sx={{ mt: 3, mb: 2, }}
        >
        Atrás
        </Button>
        </Box>
        </Grid>
        </Grid>
        </ThemeProvider>
        )}
```

Código de la ventana 'Historial de comida'

```
import Paper from '@mui/material/Paper';
import Box from '@mui/material/Box';
import Grid from '@mui/material/Grid';
import Typography from '@mui/material/Typography';
import { createTheme, ThemeProvider } from '@mui/material/styles';
const theme = createTheme();
export const Historialdecomida = () => {
  var [contactObjects, setContactObjects] = useState({})
  useEffect(() => {
       firebaseDb.child('Base de Datos-Comida')
       .on('value', snapshot => {
       if (snapshot.val() != null)
       setContactObjects({...snapshot.val()})
       else
       setContactObjects({})
       })}
       ,[])
  return (
    <ThemeProvider theme={theme}>
      <Grid container component="main" sx={{ height: '100vh' }}>
        <CssBaseline />
          <Grid
            item
            xs={false}
            sm={4}
            md={7}
            sx={{
            backgroundImage: 'url(river1.jpg)',
```

```
            backgroundRepeat: 'no-repeat',
            backgroundColor: (t) =>
            t.palette.mode === 'light' ? t.palette.grey[50] :
t.palette.grey[900],
            backgroundSize: 'cover',
            backgroundPosition: 'center',
            }}
          />
          <Grid item xs={12} sm={8} md={5} component={Paper}
elevation={6} square>
          <Box
            sx={{
            display: 'flex',
            flexDirection: 'column',
            alignItems: 'center',
            }}
          >
           <table className="table" border="2"  style={{background:"
#FFFFFF"}}>
            <thead className="thead-light">
            <tr style={{textAlign:"right" }}><Link to='/'
style={{color:'red' }}>X</Link></tr>
            <tr><th>
            <Typography align="center" component="h1" variant="inherit">
            Historial de comida
            </Typography>
            </th></tr>
            </thead>
            <tbody>{
            Object.keys(contactObjects).map(id => {
            return <tr
key={id}><td>{contactObjects[id].comida}</td></tr>
             })
            }</tbody>
            </table>
            </Box>
        </Grid>
      </Grid>
    </ThemeProvider>
    );
}
```

Anexo E

Manual de usuario

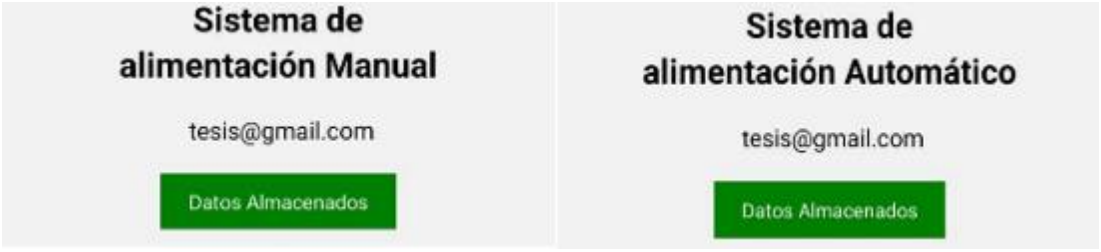
Paso1: Abrir la aplicación y registrarse con un correo que tenga acceso a recibir mensajes de entrada.

Paso2: Elegir el tipo de automatización ya sea Automático o Manual

Paso3: En caso de elegir la automatización de manera Manual, el usuario debe ingresar la cantidad de comida, hora, intervalos y repeticiones que desea que el dispensador ejecute.

Paso4: En caso de elegir la automatización de manera Automática, la cantidad de comida, hora, intervalos y repeticiones vienen dadas según la biomasa de los alevines que envía el sistema de conteo.

Paso5: Para poder verificar que los datos ingresados tanto de manera Manual como Automático, cada ventana tiene su botón denominado ‘Datos almacenados’ que permite verificar que los datos se han almacenado en la plataforma de Firebase.



Paso6: Por último, hay el botón ‘Historial de comida’ que se utiliza para poder verificar que el dispensador de comida ya haya ejecutado lo que el usuario haya ingresado en el sistema manual o automático, este botón se encuentra en la ventana ‘Home’.

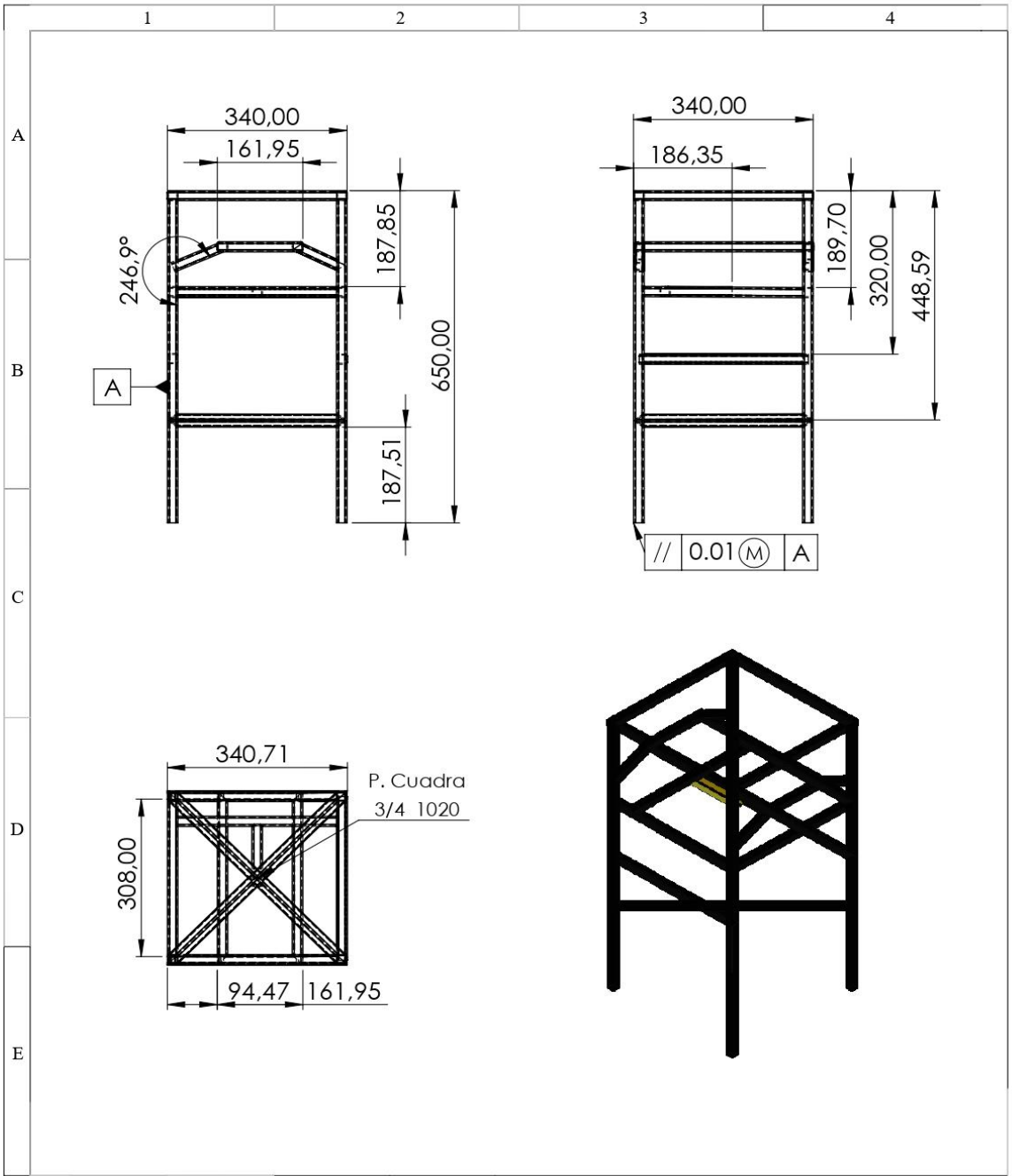
Anexo F

Registro de temperatura del agua del estanque durante el mes de mayo mediante el sensor DS18B20

Dia	Horario de registro de temperatura ℃					Promedio diario ℃
	8:00	10:00	12:00	14:00	16:00	
1	13,05	13,44	13,88	15,06	15,19	14,12
2	14,52	15,25	16,38	15,88	15,06	15,42
3	13,12	14,01	14,28	14,56	14,38	14,07
4	12,01	12,05	12,58	12,81	12,45	12,38
5	11,12	12,47	12,98	14,52	15,45	13,31
6	10,51	11,74	12,47	13,15	14,22	12,42
7	9,47	9,51	10,41	11,69	11,58	10,53
8	10,77	11,47	12,01	12,54	13,05	11,97
9	11,14	11,45	11,84	12,58	14,15	12,23
10	13,44	13,85	14,57	14,64	13,44	13,99
11	13,84	14,51	15,43	16,22	16,32	15,26
12	14,21	14,47	15,23	16,41	16,38	15,34
13	14,22	14,38	15,15	16,01	16,42	15,24
14	14,77	14,35	14,98	15,24	16,01	15,07
15	13,25	13,87	14,11	15,27	13,41	13,98
16	7,45	9,25	11,21	14,32	12,41	10,93
17	8,47	8,42	9,14	9,37	10,14	9,11
18	10,14	11,21	12,44	13,22	14,11	12,22
19	8,69	10,11	11,11	12,14	12,02	10,81
20	12,14	12,74	12,36	15,25	15,71	13,64
21	13,11	13,45	13,85	14,72	13,22	13,67
22	13,45	14,33	14,51	15,32	15,04	14,53
23	12,74	13,47	13,98	14,52	14,39	13,82
24	13,21	14,85	15,12	15,46	15,82	14,89
25	13,69	13,87	14,23	14,56	15,01	14,27
26	14,02	14,38	14,22	14,85	15,03	14,50
27	13,27	13,35	13,74	13,78	13,85	13,60
28	13,85	14,11	14,52	15,02	15,16	14,53
29	13,22	13,52	13,41	13,54	13,74	13,49
30	13,01	13,11	13,45	13,49	13,13	13,24
Promedio mensual ℃						13,42

Anexo G

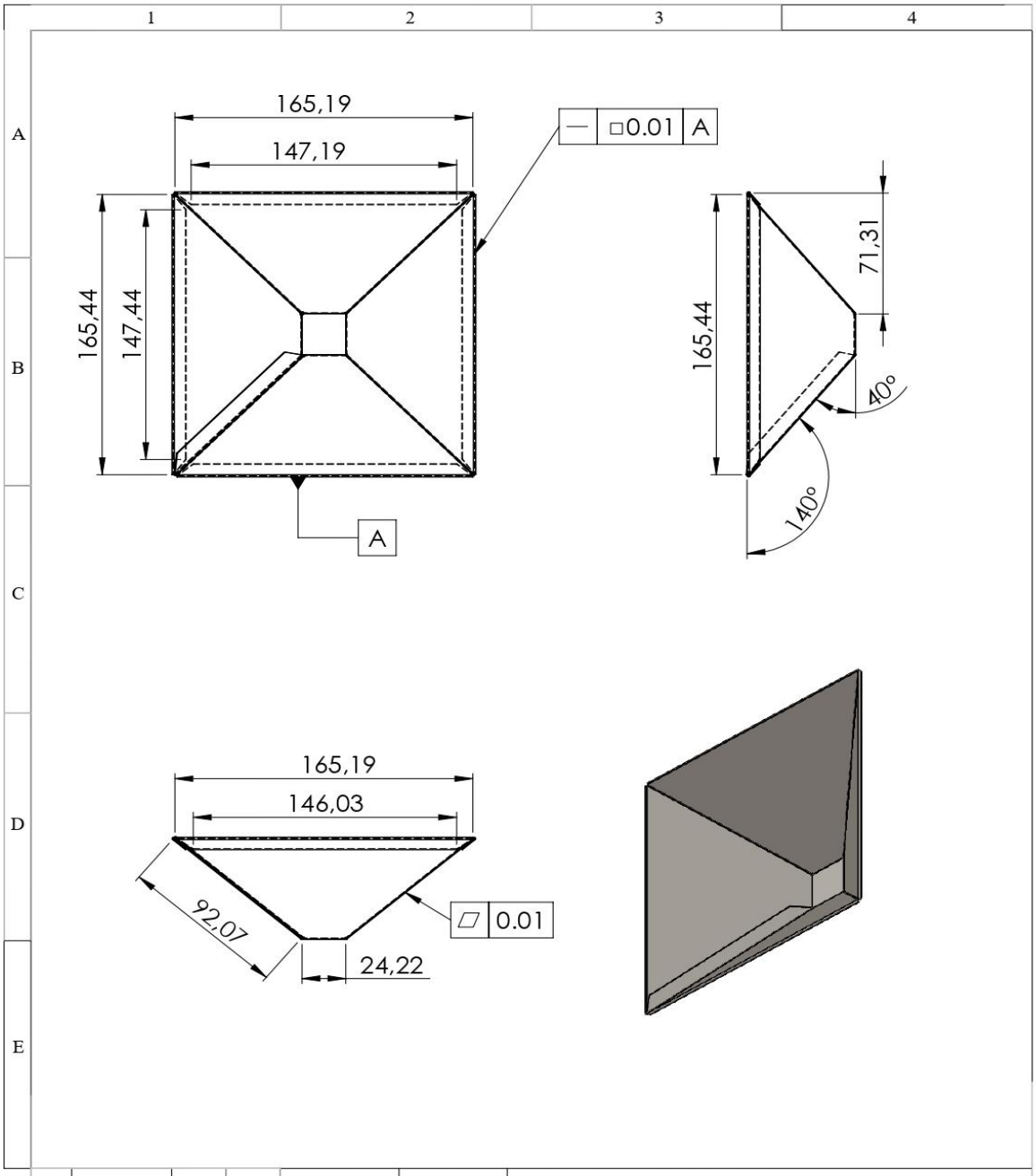
Estructura Base - Alimentador



Tolerancia: ±0.01	Peso: 0.4 Kg	Materiales y tratamiento Termico. : **AISI 1020**				
	Fecha:	Nombre:	Estructura Base - Alimentador			Escala: 1:10
Dib.		Basantes M.				
Rev.		Ing. Altamirano S.				
Apro.		Basantes M.				
UTA FISEI			**CAD**			
Edi_ción	Modificación	Fecha	Nombre			

Anexo H

Estructura Tolva inferior – Alimentador



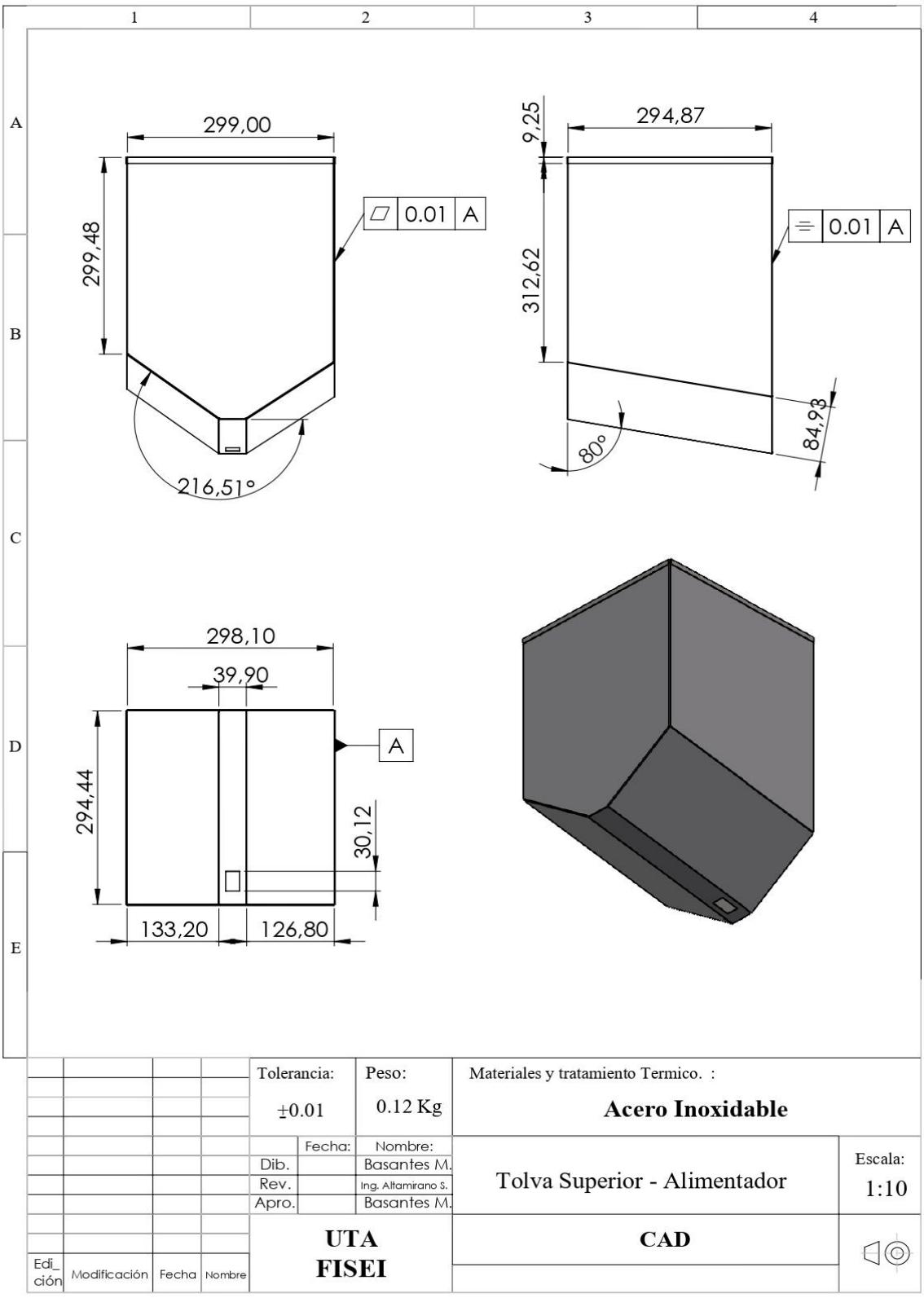
				Tolerancia: ±0.01	Peso: 0.06 Kg	Materiales y tratamiento Termico. : **Acero Inoxidable**
					Fecha:	Nombre:
				Dib.		Basantes M.
				Rev.		Ing. Altamirano S.
				Apro.		Basantes M.
Edi_ción	Modificación	Fecha	Nombre	**UTA FISEI**		**CAD**

Tolva Inferior - Alimentador

Escala: 1:2

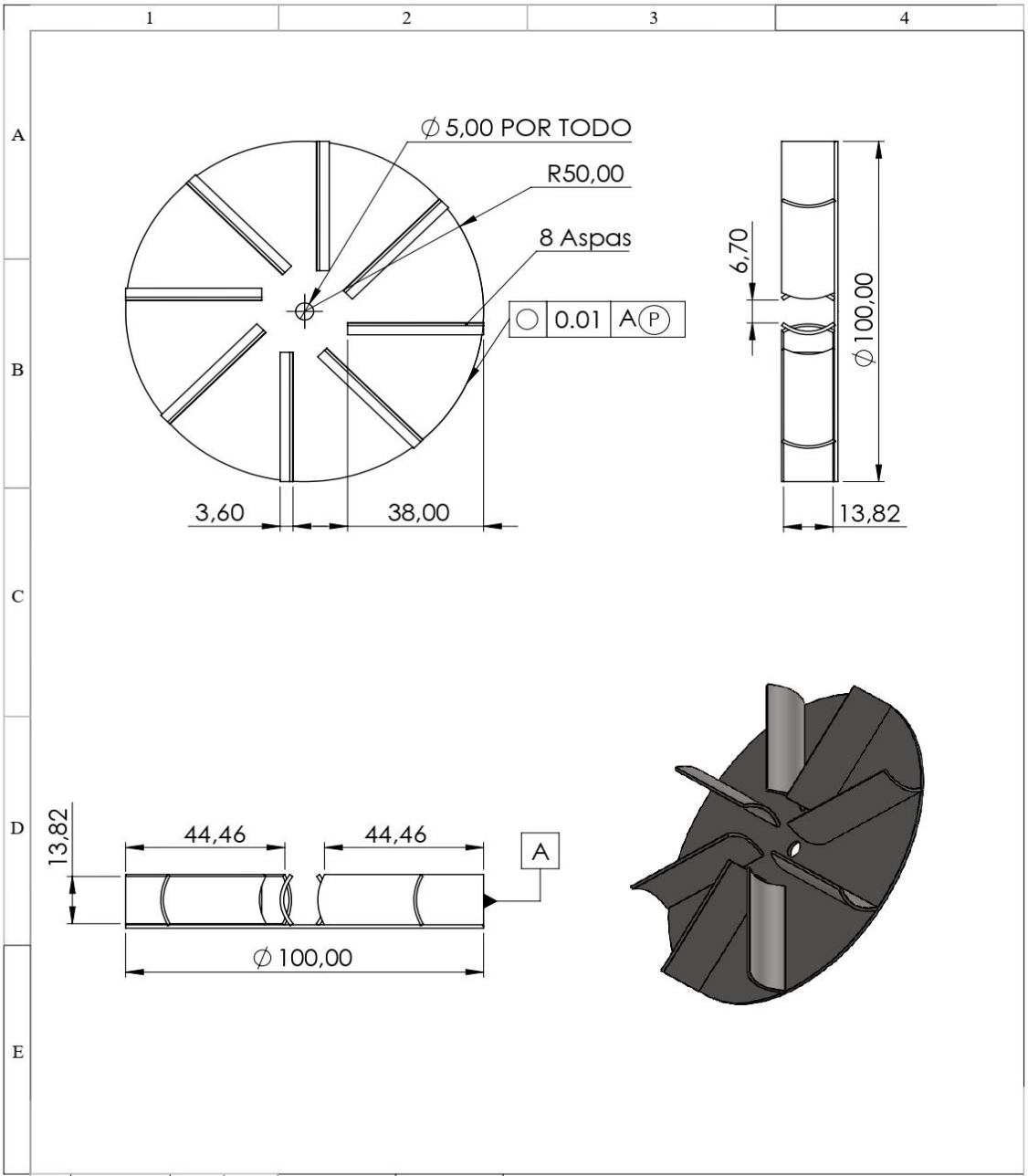
Anexo I

Estructura Tolva superior – Alimentador



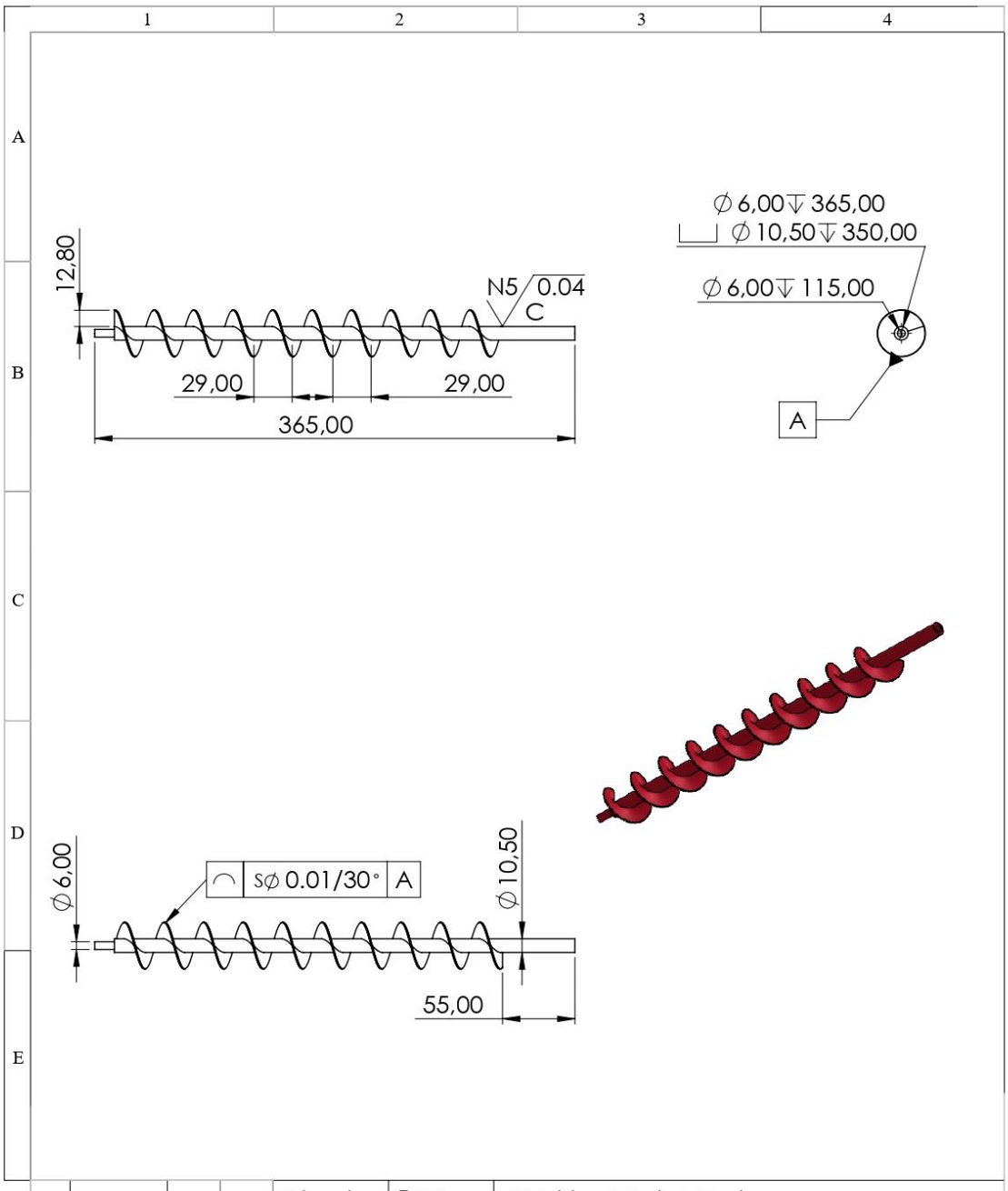
Anexo J

Disco Centrífugo – Alimentador



Anexo K

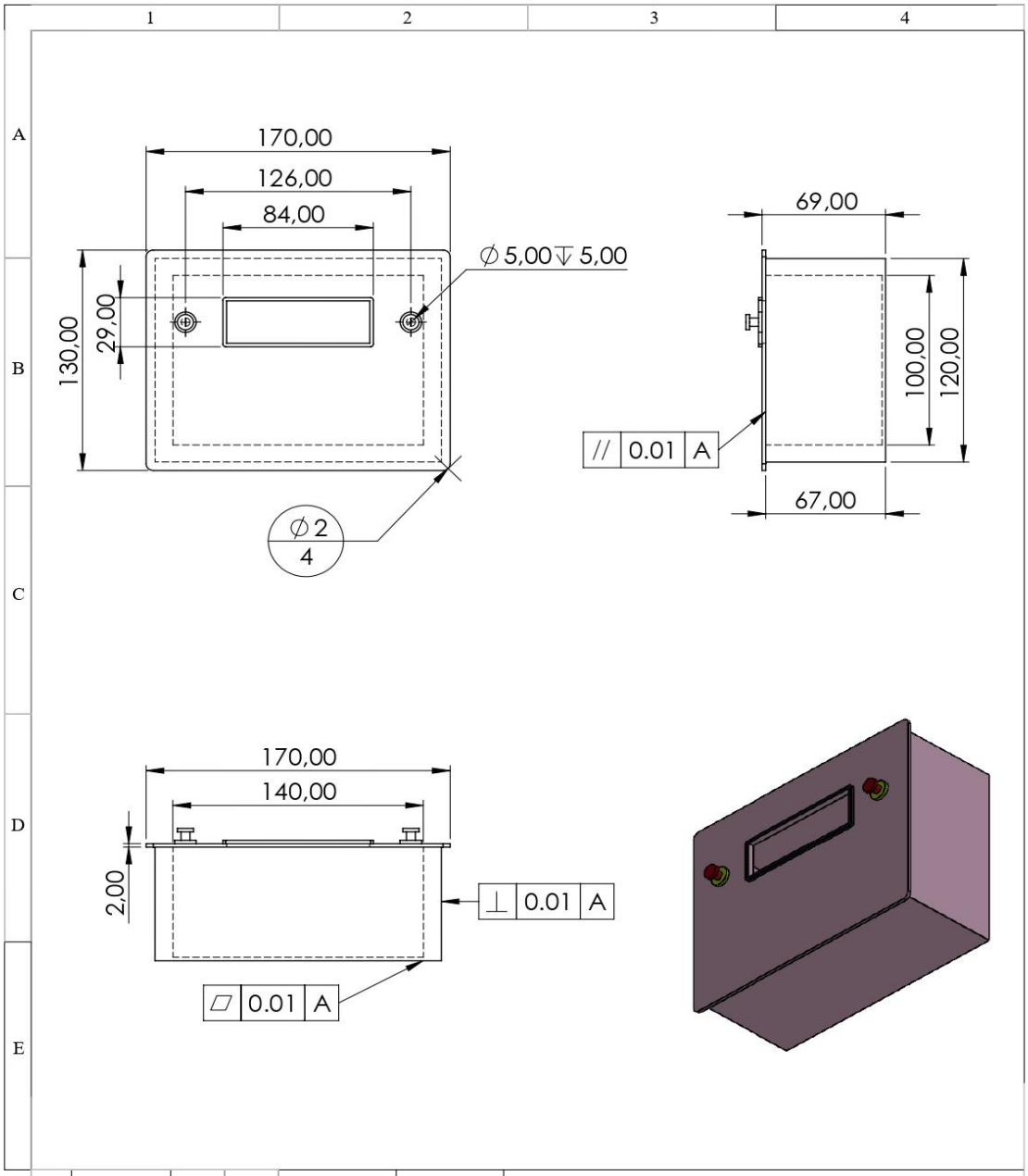
Tornillo sin fin – Alimentador



				Tolerancia: ±0.01	Peso: 0.07 Kg	Materiales y tratamiento Termico. : **Acero Inoxidable**
				Fecha:	Nombre:	Escala: 1:5
				Dib.	Basantes M.	Tornillo Sin Fin - Alimentador
				Rev.	Ing. Altamirano S.	
				Apro.	Basantes M.	
Edi_ción	Modificación	Fecha	Nombre	**UTA FISEI**		**CAD**

Anexo L

Caja de componentes – Contador



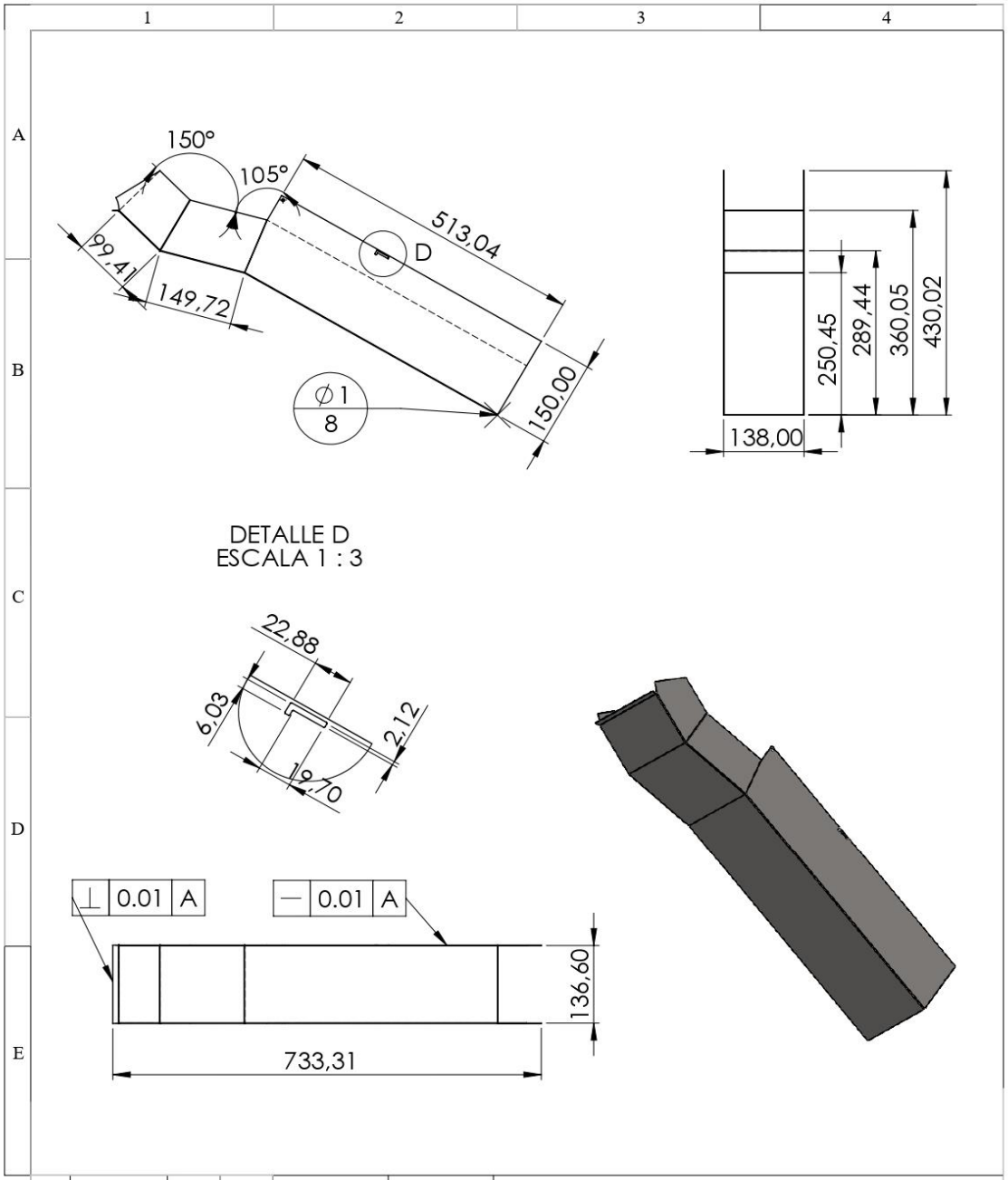
				Tolerancia: ±0.01	Peso: 0.02 Kg	Materiales y tratamiento Termico. : **Acrilico Blanco 3mm**
					Fecha:	Nombre:
				Dib.		Basantes M.
				Rev.		Ing. Altamirano S.
				Apro.		Basantes M.
Edi_ción	Modificación	Fecha	Nombre	**UTA FISEI**		**CAD**

Caja Display - Alimentador

Escala: 1:2

Anexo M

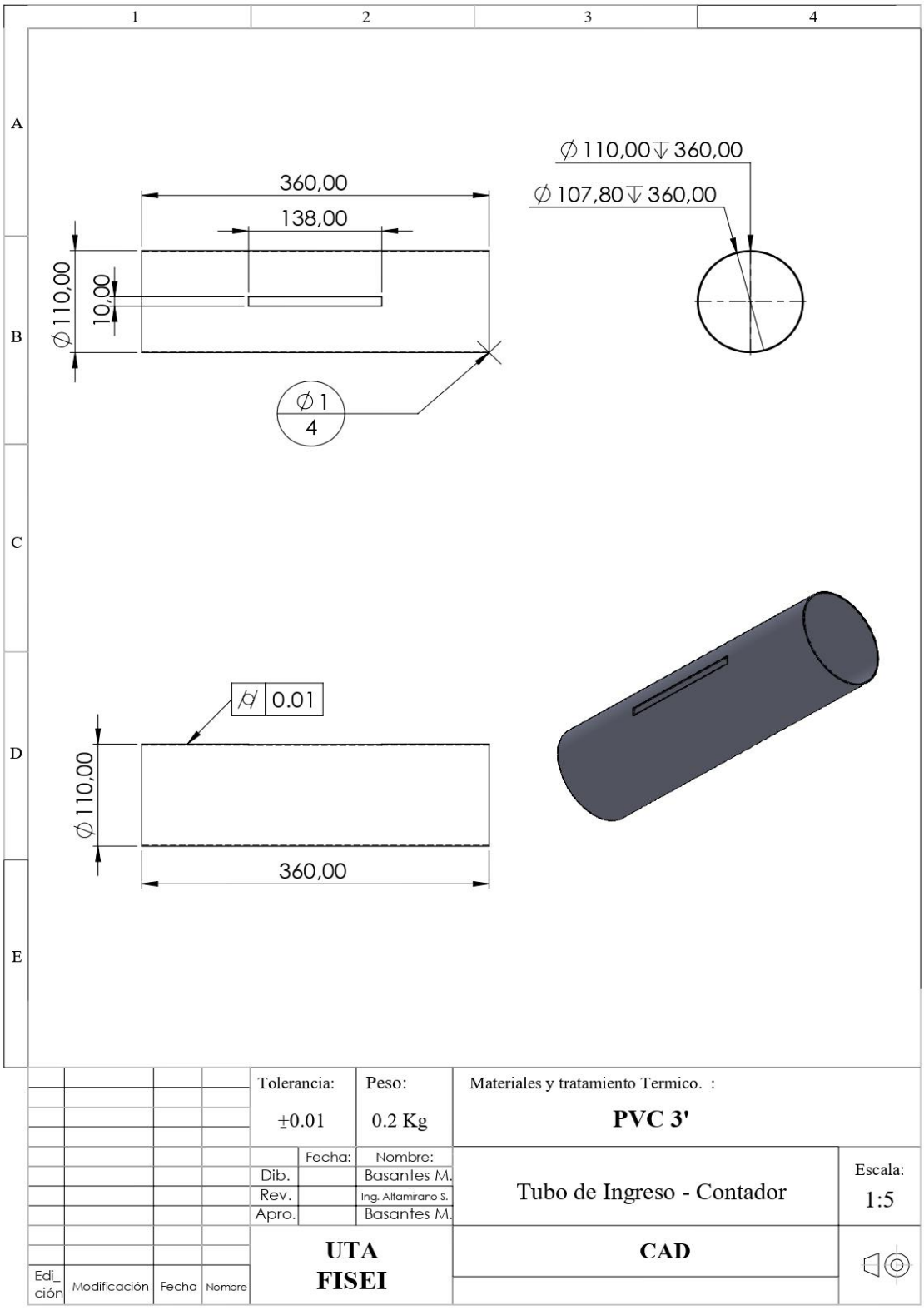
Estructura – Contador



				Tolerancia: ±0.01	Peso: 0.21 Kg	Materiales y tratamiento Termico. : **Asero Inoxidable Inox Cepillado**
					Fecha: Nombre:	
				Dib.	Basantes M.	Estructura Base - Contador — Escala: 1:10
				Rev.	Ing. Altamirano S.	
				Apro.	Basantes M.	
Edi_ción	Modificación	Fecha	Nombre	**UTA FISEI**		**CAD**

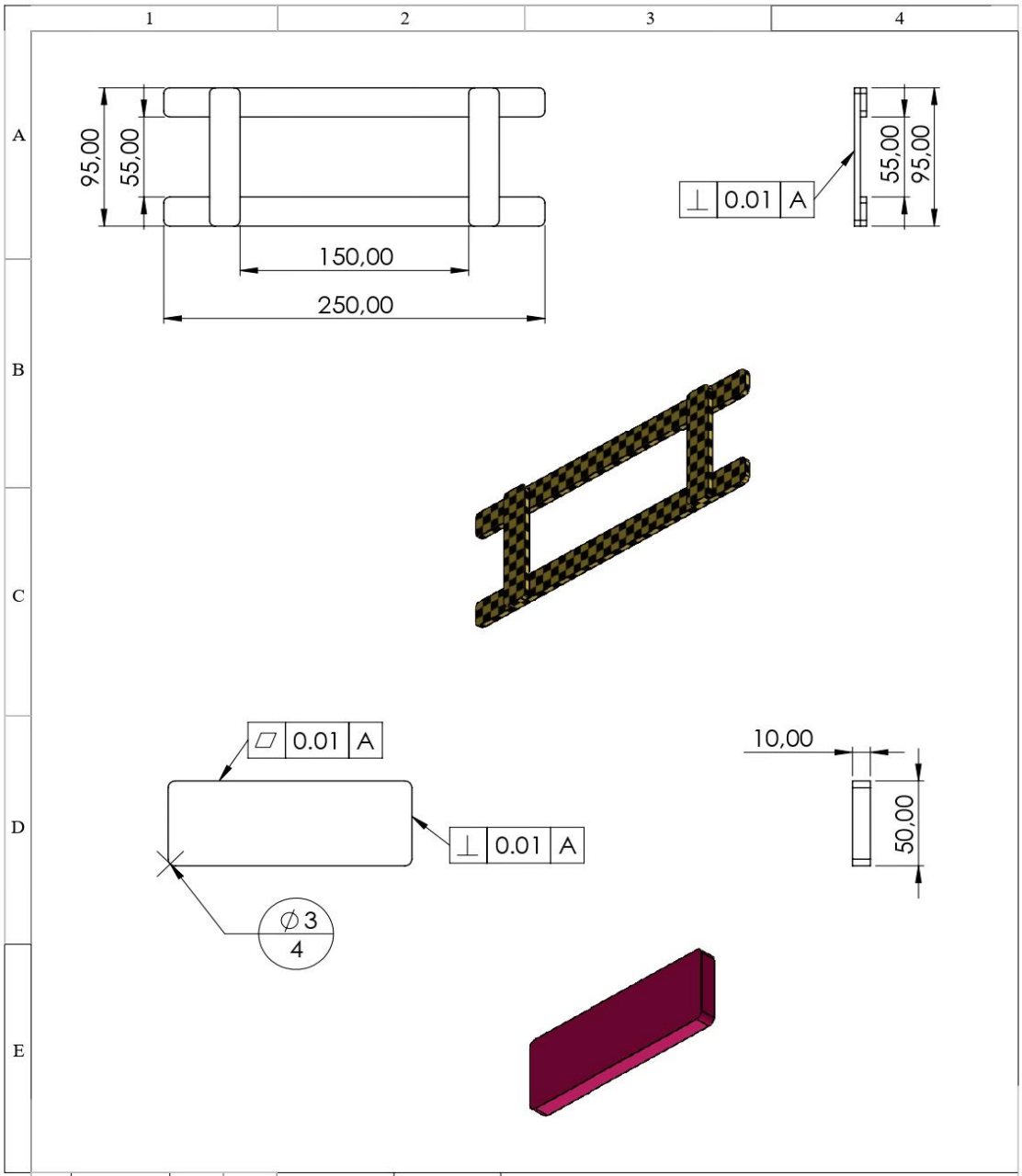
Anexo N

Tubo de ingreso – Contador



Anexo Ñ

Base de iluminación – Contador



Tolerancia: ±0.01	Peso: 0.1 Kg	Materiales y tratamiento Termico. : MDF Prensado 4mm	
	Fecha:	Nombre:	
Dib.		Basantes M.	Base Superior - Contador — Escala: 1:2
Rev.		Ing. Altamirano S.	
Apro.		Basantes M.	
UTA FISEI			CAD

Edi_ción	Modificación	Fecha	Nombre