CroqueFeed Sistemas de Alimentación Automatizada.

*Alumnos: *Gómez Hernández Alan Javier, Villegas Palacios David.*
Profesor: Úrsula Samantha Morales Rodríguez
**email: agomezh1901@alumno.ipn.mx*

Resumen – En la presente propuesta de proyecto final se propone el desarrollo de una aplicación web para mejorar la calidad de vida de las mascotas y simplificar la tarea de sus cuidadores. Utiliza un dispensador de croquetas con capacidades avanzadas y un servicio web para monitorear y controlar la alimentación de las mascotas. El sistema incorpora tecnología de sensor ultrasónico para medir el nivel de croquetas y notificar al usuario cuando es necesario rellenar el dispensador. También permite al usuario abrir remotamente la compuerta del dispensador para alimentar a la mascota. Se centra en el bienestar de las mascotas, el cuidado responsable, la facilitación de la vida cotidiana, el monitoreo en tiempo real, el control a distancia y la reducción del desperdicio de alimentos.

Palabras clave – Aplicación web, ESP32, Cuidado responsable, IoT, Alimentación automatizada.

1. Introducción

Automoción, administración, educación o salud son algunos de los sectores que están aprovechando las ventajas que ofrece el Internet de las Cosas (IoT) como es la reducción de costes, la optimización de procesos y la mejora de servicios. Ahora, estas avanzadas tecnologías aterrizan en FIEB (Fundación para la Investigación en Etología y Biodiversidad) para asegurar el bienestar de los animales que se alojan en su refugio CITES, situado a las afueras de Madrid.[1]

La creciente adopción de tecnologías de Internet de las Cosas (IoT) ha transformado numerosos aspectos de nuestra vida cotidiana, incluyendo la forma en que cuidamos a nuestras mascotas. La propuesta de este proyecto se centra en el desarrollo de una aplicación web que utiliza la tecnología IoT para mejorar la calidad de vida de las mascotas y simplificar la tarea de sus cuidadores.[2]

El cuidado de las mascotas es una responsabilidad que requiere atención y tiempo. Sin embargo, debido al ritmo de vida actual, a veces puede ser difícil mantener una rutina de alimentación constante para nuestras mascotas. Aquí es donde la tecnología IoT puede desempeñar un papel crucial. Los dispositivos IoT para el cuidado de mascotas pueden ayudar a los dueños a monitorear y controlar la alimentación de sus mascotas, incluso cuando no están en casa. [3]

El bienestar de nuestras mascotas es de suma importancia, y uno de los aspectos fundamentales para su salud es la alimentación adecuada y constante. Sin embargo, en la vida moderna, nuestras ajetreadas rutinas a veces pueden dificultar cumplir con esta responsabilidad de manera consistente.

Para abordar este desafío, hemos concebido un sistema inteligente que revolucionará la forma en que alimentamos a nuestras mascotas. Este proyecto consta de un dispensador de croquetas con capacidades avanzadas y una aplicación móvil que le permite al usuario monitorear y controlar la alimentación de sus mascotas de manera conveniente.

Este proyecto propone el uso de ESP32, un chip que proporciona conectividad Wi-Fi y Bluetooth para dispositivos integrados, lo que lo hace ideal para aplicaciones de IoT. El sistema incorporará un dispensador de croquetas con capacidades avanzadas y un sensor ultrasónico para medir el nivel de croquetas y notificar al usuario cuando sea necesario rellenar el dispensador. [5]

La funcionalidad permite al usuario abrir la compuerta del dispensador durante aproximadamente 3 segundos, permitiendo que la comida caiga en el plato de la mascota. Esta acción se puede realizar de forma remota a través de la aplicación móvil, lo que brinda un control total sobre la alimentación, incluso cuando el cuidador no está presente en casa.

En este documento, presentaremos una descripción detallada de las características, funcionalidades y beneficios de "CroqueFeed", además de los objetivos del proyecto, los recursos necesarios y un plan de implementación. Además, solicitaremos la autorización para avanzar con la ejecución de este emocionante proyecto que tiene el potencial de transformar la forma en que cuidamos y alimentamos a nuestras mascotas.

Estamos seguros de que "CroqueFeed" marcará un hito en el mundo de la alimentación de mascotas, y esperamos contar con su respaldo y aprobación para llevar a cabo este proyecto visionario.

Objetivo

Diseñar e implementar "CroqueFeed: Sistema de Alimentación Automatizada", una solución tecnológica innovadora destinada a elevar la calidad de vida de las mascotas y facilitar la responsabilidad de sus cuidadores. El objetivo principal es lograr un control detallado y remoto sobre el proceso de alimentación de las mascotas, proporcionando una experiencia integral que promueva el bienestar animal y simplifique la gestión alimentaria para los dueños. Este sistema integrará de manera eficiente la automatización a través de la placa ESP32, sensores ultrasónicos y actuadores para brindar un monitoreo en tiempo real, notificaciones claras y apertura remota de la compuerta del dispensador.[11] La solución también incorporará capas de seguridad, como la autenticación a través de Firebase, para garantizar un acceso seguro a la aplicación web asociada. Este objetivo se alinea con nuestra visión de transformar la forma en que las mascotas son alimentadas, brindando comodidad y tranquilidad a los cuidadores, y promoviendo un cuidado responsable y avanzado a través de la tecnología.

Objetivos Específicos

1. Diseñar y construir un dispensador de croquetas automatizado utilizando una placa ESP32, un sensor ultrasónico HC-SR04 y dos servomotores SG-90.

2. Desarrollar una aplicación web que permita a los usuarios monitorear y controlar remotamente la alimentación de sus mascotas. La aplicación se desarrollará utilizando React para la interfaz de usuario, Vite para el entorno de desarrollo, Tailwind CSS para el diseño, y Node.js y Express.js para el backend.
3. Implementar una funcionalidad de notificación en tiempo real que alerte a los usuarios cuando el nivel de croquetas en el dispensador sea bajo y necesite ser rellenado.
4. Integrar Firebase para la autenticación de usuarios, proporcionando una capa adicional de seguridad para la aplicación web.
5. Realizar pruebas de funcionalidad y usabilidad para garantizar que el sistema funciona correctamente y es fácil de usar para los cuidadores de mascotas.

Justificación

Desafortunadamente hoy en día millones de personas tienen al cuidado un animalito, sin embargo, una cantidad de estas personas no están al pendiente de alimentar o proveer las necesidades básicas de ellos, dado que los animalitos no tienen la facilidad de poder hacerlo por si solos se ve como un problema. Hoy en día con las tecnologías y el internet de las cosas podemos darle un cambio radical, disminuyendo el porcentaje de animales con mal cuidado, facilitando las cosas con una aplicación que pueda mandarle alertas para alimentar a sus mascotas. En el hecho de que muchas personas tienen muchos pendientes, o ocupaciones diferentes, se busca facilitar la vida cotidiana de ellos y sobre todo de tener un mejor cuidado de los animalitos en casa. [3]

La necesidad del Sistema de Alimentación Automatizada se basa en la creciente importancia de las mascotas en la vida de las personas y la evolución de la tecnología, que nos brinda la oportunidad de abordar sus necesidades de manera más efectiva. A continuación, se presentan las razones fundamentales que respaldan la implementación de este proyecto:

Bienestar de las Mascotas: Las mascotas son parte integral de nuestras vidas y merecen una alimentación constante y adecuada. "CroqueFeed" garantiza que las mascotas reciban comida en el momento adecuado, incluso cuando sus cuidadores no pueden estar presentes, lo que contribuye significativamente a su bienestar y salud.

Cuidado Responsable: La alimentación adecuada y consistente es esencial para el cuidado responsable de las mascotas. "CroqueFeed" empodera a los cuidadores con herramientas para garantizar que sus mascotas reciban la cantidad de comida necesaria y en el momento adecuado, sin importar su ubicación.

Facilitación de la Vida Cotidiana: Las agendas ocupadas y los compromisos diarios pueden dificultar la atención inmediata a las necesidades de las mascotas. El sistema "CroqueFeed" alivia a los cuidadores de la preocupación constante de la alimentación de sus mascotas, permitiéndoles centrarse en otros aspectos de su vida cotidiana.

Monitoreo en Tiempo Real: La función de notificación basada en sensores ultrasónicos proporciona una capacidad de monitoreo en tiempo real del nivel de croquetas en el dispensador. Esto asegura que el usuario esté al tanto de cuánta comida queda y pueda rellenar el dispensador de manera oportuna.

Control a Distancia: La aplicación web asociada a "CroqueFeed" permite a los usuarios controlar la alimentación de sus mascotas desde cualquier lugar con una conexión a Internet. Esta funcionalidad ofrece comodidad y flexibilidad, lo que es especialmente útil para personas que viajan con regularidad.

Tecnología Innovadora: "CroqueFeed" utiliza tecnología de vanguardia, como sensores ultrasónicos y control remoto a través de una aplicación móvil. Esto muestra nuestro compromiso con la innovación y la mejora continua de las soluciones de alimentación para mascotas.

Reducción de Desperdicio de Alimentos: Al proporcionar una alimentación precisa y controlada, "CroqueFeed" ayuda a reducir el desperdicio de comida, lo que a su vez tiene un impacto positivo en el medio ambiente.

Problemática

La problemática abordada del proyecto se centra en las dificultades que enfrentan los dueños de mascotas para proporcionar una alimentación constante y adecuada debido a las ajetreadas rutinas diarias. La falta de tiempo y la distracción cotidiana pueden llevar a situaciones en las que las mascotas no reciben la atención alimentaria necesaria. Estudios han demostrado que un patrón de alimentación irregular puede llevar a un aumento en los comportamientos relacionados con el estrés en las mascotas.[10]

Además, la alimentación inadecuada puede tener un impacto significativo en la salud y el bienestar de las mascotas. Por ejemplo, la sobrealimentación puede llevar al sobrepeso, que a su vez puede causar una serie de problemas de salud. Por otro lado, la subalimentación puede resultar en desnutrición y sus asociados problemas de salud. [4]

A pesar de la existencia de alimentadores automáticos de mascotas en el mercado, muchos dueños de mascotas todavía enfrentan desafíos. Algunos alimentadores automáticos pueden atascarse durante el proceso de dispensación de alimentos, mientras que otros pueden sobrealimentar a las mascotas. Además, muchos de los alimentadores automáticos existentes no ofrecen la capacidad de monitorear y controlar la alimentación de las mascotas de manera eficiente, especialmente cuando los dueños de mascotas no están presentes físicamente. [5]

Esta problemática se agrava por la ausencia de una solución tecnológica integral que permita a los cuidadores monitorear y controlar la alimentación de sus mascotas de manera eficiente. Aquí es donde el proyecto “CroqueFeed: Sistema de Alimentación Automatizada” entra en juego, proponiendo una solución innovadora para abordar estos desafíos.

Propuesta de solución

Para abordar el problema de la alimentación de mascotas, se propone el proyecto “CroqueFeed: Sistema de Alimentación Automatizada”. Este sistema se basa en una serie de componentes y tecnologías que trabajan en conjunto para proporcionar una solución integral**.**

1. Hardware: El corazón del sistema es una placa ESP32, que permite la conexión a internet vía Wi-Fi para transmitir datos1. Un sensor ultrasónico HC-SR04 se utiliza para proporcionar mediciones en tiempo real del nivel de croquetas en el dispensador. Dos servomotores SG-90 actúan como actuadores para la compuerta del dispensador. Todo esto se monta en una estructura hecha de tubos de PVC, con conexiones realizadas con jumpers y una tabla de prototipo de conexión. La programación del ESP32 se realiza utilizando el IDE de Arduino.
2. Aplicación web: La aplicación web se desarrolla utilizando React para la interfaz de usuario, Vite para el entorno de desarrollo y Tailwind CSS para el diseño. El backend de la aplicación se desarrolla utilizando Node.js y Express.js, proporcionando una API para interactuar con el hardware6. Se utiliza Firebase para la autenticación de usuarios, proporcionando una capa adicional de seguridad.

Este sistema permite a los usuarios tener un control preciso sobre la alimentación de sus mascotas, incluso cuando no pueden estar presentes físicamente. La alimentación se automatiza a través de la apertura de una compuerta del dispensador, que se puede activar remotamente desde la aplicación web. Además, el sensor ultrasónico proporciona mediciones en tiempo real del nivel de croquetas en el dispensador y envía notificaciones claras sobre la necesidad de rellenarlo. De esta manera, se aborda la falta de monitoreo en tiempo real, un problema común entre los cuidadores que desean estar seguros de que sus mascotas están siendo alimentadas adecuadamente.

Estado del arte

Nombre del dispensador.	Características	Precio
Yuposl Comedero Automático para Gatos	Puede establecer porciones por alimentación y establecer las comidas de 1 a 6 para sus mascotas. Configurado para que suelte croquetas cada cierto tiempo.	$800
Comederos Automáticos Para Gatos Con Cámara, Pumpkii 4l Come.	Establece horario de alimentación preferido de forma remota con la aplicación móvil gratuita (solo admite red Wi-Fi de 2,4 GHz) | Cámara HD para mascotas.	$3700
Alimentador Automático Perros Y Gatos 6l Wifi App	Aplicación de control remoto El comedero automático para gatos WiFi se puede conectar a una red de 2,4 GHz, y puede organizar y controlar las comidas de su mascota en cualquier momento y en cualquier lugar a través de la aplicación "TUYA" en su teléfono inteligente iOS o teléfono Android.	$1689
CroqueFeed Sistemas de Alimentación Automatizada	Servicio web para alimentar a tu mascota de manera remota, al igual que una interfaz de seguridad. Usando Wifi, y monitoreo desde cualquier lugar.	$804

Diseño y Arquitectura del sistema

En la Figura 1 podemos ver que la arquitectura del sistema para el prototipo consta de varios componentes interconectados. El

ESP32 actúa como centro de control y recopila datos de los sensores. Estos datos se muestran en tiempo real en una pantalla LCD de 16x2. Además, los datos se envían a través de una conexión Wi-Fi a una aplicación web, donde los usuarios pueden acceder y visualizar la calidad del aire en su ubicación. La aplicación web nos dará las dos funciones, la primera es la alerta de que esta apunto las croquetas del dispensador y la segunda el botón de interacción para liberar las croquetas al plato. En la figura uno podemos ver la interacción entre el ESP32 con los componentes que tendrá el dispensador, al igual que el dispensador solo va a interactuar con la aplicación mandándole señales para que actúen los dispositivos que están conectados con base a la aplicación. Para la figura 2 tenemos el diagrama visto desde la implementación real y como lo llevamos al MVC con los dispositivos usados en el esp32 en el modelo que aquí funge como un servidor aparte ya que espera peticiones http para hacer la instrucción deseada que es "abrir compuerta" o "obtener el valor de croquetas en el contendor" mandada desde el controlador con GET, finalmente devuelve en JSON la respuesta que puede ser el valor del contenedor lleno o vacío, en caso de la compuerta que ya fue abierta. Esta información es recibida por el controlar que es el encargado de gestionarla para mandarla a la vista que es el servicio web responsivo. El controlador es quien interactúa por ambas partes para que la información sea gestionada, se visualice y mande instrucciones al dispositivo ES32, es decir que es quien maneja las peticiones HTTP. Finalmente, la vista es lo que ve el usuario de primera instancia un login con firebase para que verifique es una persona real y evitar las peticiones de cualquier dispositivo, después de este login tiene la visualización de los datos del contenedor de croquetas y puede abrir la compuerta con un botón, básicamente es la interfaz lo que ve el usuario, las peticiones que se hacen aquí de consultas y de interacción pasan al controlador para verificarlas, y finalmente para que lleguen al modelo.

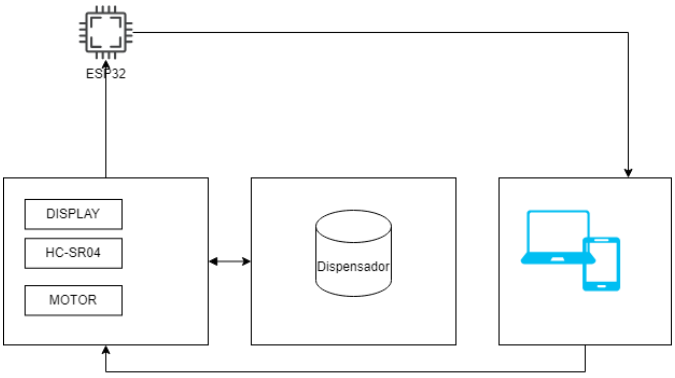


Ilustración 1. Diagrama del sistema prototipo.

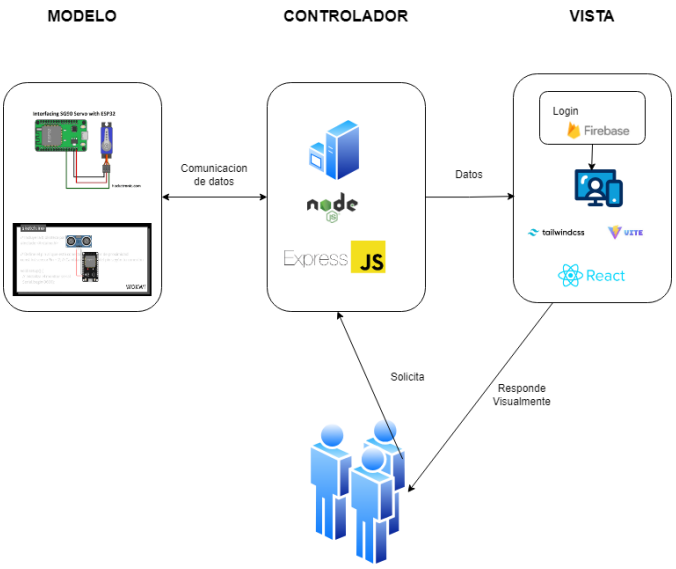


Ilustración 2. MVC.

Racionalidad de la solución

La racionalidad de la solución propuesta, "CroqueFeed: Sistema de Alimentación Automatizada", se basa en la necesidad de abordar los desafíos inherentes a la alimentación de mascotas y mejorar la calidad de vida tanto de los animales como de sus cuidadores. La alimentación constante y adecuada es esencial para el bienestar de las mascotas, y cualquier cambio inesperado en la rutina de alimentación puede causar ansiedad en las mascotas. "CroqueFeed" permite mantener un control continuo sobre la alimentación, asegurando que las mascotas reciban la cantidad de comida necesaria, incluso en ausencia de sus cuidadores. Esto contribuye a la salud y el bienestar de las mascotas, lo que a su vez mejora la relación entre las mascotas y sus dueños.

En cuanto a la optimización de recursos, el desperdicio de alimentos para mascotas es un problema común4. La solución "CroqueFeed" establece una alimentación precisa y controlada, lo que disminuye significativamente el desperdicio de comida y, en última instancia, reduce los costos asociados con la alimentación de mascotas. Además, "CroqueFeed" ayuda a prevenir la obesidad y los trastornos relacionados en las mascotas.

8. Costos

DISPOSITIVOS O MATERIAL	DESCRIPCION	COSTO
Tubos de PVC	Para el armado del dispensador	$150
ESP32	Placa Wifi microcontrolador para la conexión con la app.	$250
Jumper de conexión	Para las conexiones del sensor, actuador y el microcontrolador.	$50

Tabla de prototipo de conexión	Para las conexiones de nuestro circuito.	$159
hc-sr04	Sensor de distancia o ultrasónico para medir la cantidad de croquetas	$75
2 Servomotores sg-90	Para la compuerta será el actuador	$120
		Total: 804

Marco teórico.

A continuación, se presentan conceptos teóricos relevantes para la comprensión del proyecto:

1. **Internet de las Cosas (IoT):**
   - *Definición:* IoT se refiere a la interconexión de dispositivos físicos mediante la integración de sensores, actuadores y conectividad a Internet para recopilar, transmitir y procesar datos. En el contexto de "CroqueFeed", el ESP32 actúa como el nodo central que se conecta a la red, permitiendo la comunicación entre el dispensador de croquetas y la aplicación web.
2. **Placa ESP32:**
   - *Concepto:* La placa ESP32 es un microcontrolador que combina Wi-Fi y Bluetooth, adecuado para proyectos de IoT. En "CroqueFeed", la ESP32 sirve como el cerebro del sistema, permitiendo la conectividad a Internet y la comunicación con la aplicación web. [7]
3. **Sensores Ultrasónicos (HC-SR04):**
   - *Función:* Los sensores ultrasónicos emiten ondas acústicas y miden el tiempo que tarda en recibir el eco, proporcionando información sobre distancias. En este proyecto, el sensor HC-SR04 se utiliza para medir el nivel de croquetas en el dispensador, facilitando el monitoreo en tiempo real.
4. **Actuadores (Servomotores SG-90):**
   - *Definición:* Los actuadores son dispositivos que generan movimientos controlados en respuesta a señales eléctricas. Los servomotores SG-90 en "CroqueFeed" controlan la apertura de la compuerta del dispensador, permitiendo la liberación controlada de croquetas.
5. **Aplicación Web con React, Vite y Tailwind CSS:**
   - *Descripción:* React es una biblioteca de JavaScript para construir interfaces de usuario interactivas. Vite es un entorno de desarrollo rápido y Tailwind CSS facilita el diseño. La aplicación web "CroqueFeed" utiliza estas tecnologías para ofrecer una interfaz amigable y receptiva.
6. **Node.js y Express.js:**
   - *Definición:* Node.js es un entorno de ejecución para JavaScript fuera del navegador, mientras que Express.js es un marco de aplicación web para Node.js. Ambos se utilizan en el backend de la aplicación web para gestionar las rutas y la lógica del servidor.
1. **Firebase para Autenticación de Usuarios:**
   - *Concepto:* Firebase es una plataforma de desarrollo de aplicaciones móviles y web adquirida por Google. En "CroqueFeed", Firebase se implementa para la autenticación de usuarios, proporcionando capas adicionales de seguridad para el acceso a la aplicación web.
2. **Modelo-Vista-Controlador (MVC):**
   - *Descripción:* MVC es un patrón de diseño arquitectónico que separa la aplicación en tres componentes principales: Modelo (datos y lógica), Vista (interfaz de usuario) y Controlador (manejo de la interacción del usuario). Este enfoque se implementa en "CroqueFeed" para organizar y estructurar el desarrollo del sistema.
3. **Seguridad en Aplicaciones IoT:**
   - *Importancia:* La seguridad en aplicaciones IoT es crucial para proteger datos sensibles y garantizar un funcionamiento seguro. La implementación de Firebase y otras medidas de seguridad en "CroqueFeed" demuestra el compromiso con la privacidad y la protección de la información del usuario.

Planificación

En el diagrama de Gantt 1 se muestra el plan para el desarrollo del proyecto en los tres meses aproximados del mismo.

Actividades	**October**				**NovemBer**				**Diciembre**			
Montaje del dispositivo con el prototipo.												
Integración del sensor de distancia y actuadores.												
Integración Wi-Fi en el dispositivo.												
Creación de la aplicación del usuario.												
Aplicación Pruebas de integración.												
Corrección de errores y mejora en el												

circuito físico.												
Visualización mejorada de la aplicación.												

Tabla 1. Diagrama de Gantt.

Implementación y Desarrollo.

En esta sección se presentará el historial de cambios para lograr que nuestro proyecto funcionará en su totalidad.

Empezado por la idea principal del proyecto antes de nuestro diagrama de Gantt ya que la finalidad de nuestro proyecto era construir un dispensador que funcionara con un dispositivo bluetooth el poder hacer la conectividad iba ser de ese modo y la aplicación seria móvil construida desde el "MIT app inventor "y l interfaz montada en el mismo, la primera construcción del prototipo fuer totalmente funcional es decir que hacia las funciones del proyecto, darle la notificación al usuario de que el dispensador estaba vacío o lleno según el caso y abrir la compuerta como se presenta en la Ilustración 4 y 5. Para ser concisos el proceso fue cómodo para nosotros ya que el diagrama de bloques, la lógica y la función de codificación es fácil de entender en la interfaz por su sistema de colores y el armado de la misma, lo que fue un reto fue implementar el módulo bluetooth ya que los pines de RX y TX iban intercambiados para la compilación y cargar el código dentro del Arduino y tenían que estar desconectados para que fuera correcto el cargar el código dentro del micro controlador Arduino, la codificación y la comunicación con el dispositivo fue gracias a la velocidad de conexión entre ellos, emparejarlos con el bloque de MIT app inventor "Listpicker" con esto pudimos realizar el primer paso. En este proceso teníamos dos caminos que seguir el primero era mejorar el diseño de la aplicación al igual que la interfaz fuera más agradable, incluso poder implementar la lógica en otros sistemas operativos ya que este solo funcionaria con Android y el pasarlo a IOS nos presentaba un nuevo reto. Por otro lado, era encaminar el proyecto para cualquier persona que tuviera wifi, pudiera usar la aplicación y que de igual forma tuviera la seguridad, y comodidad para usarla.

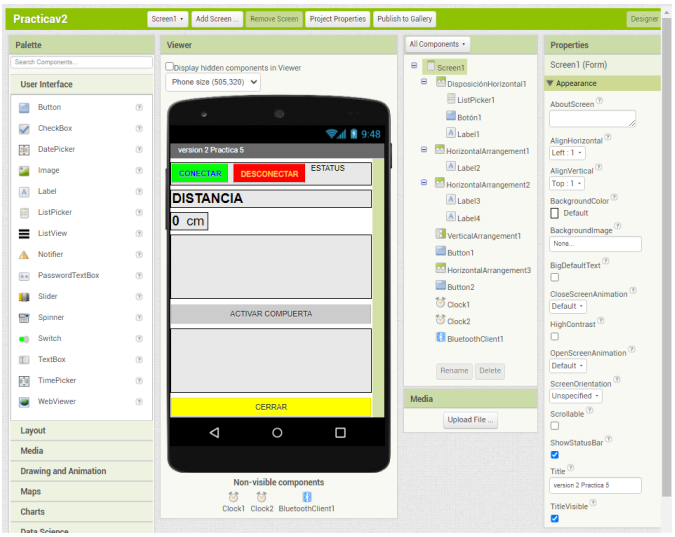


Ilustración 3. Vista del proyecto desde "MIT app inventor"

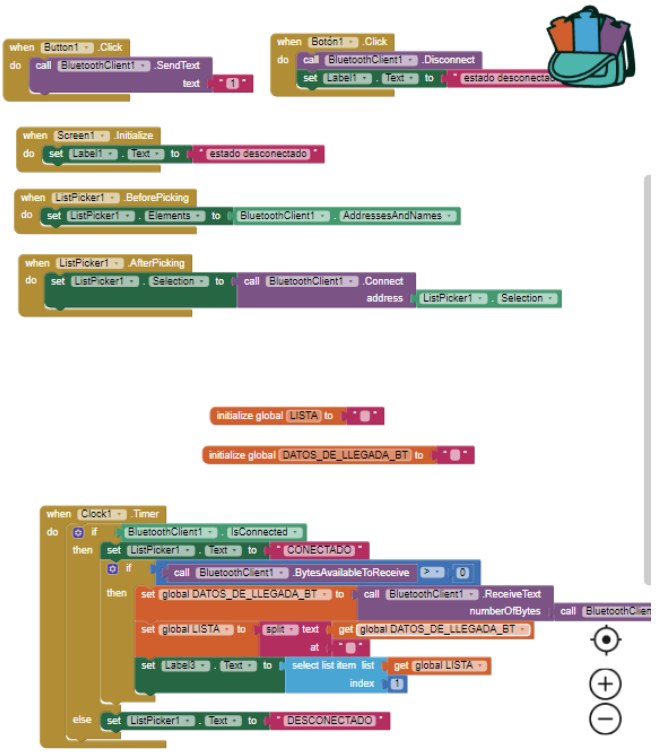


Ilustración 4. Lógica de bloques

Bien para el mes de octubre tomamos una decisión para nuestro diagrama de gannt de seguir con el proyecto como llevarlo a nuestro objetivo y resolver nuestra problemática este fue el de usar wifi y una interfaz web para que cualquier persona pudiera tener acceso, aprovechar el potencial de IOT con diferentes tecnologías. Por ende, la actividad siguiente fue "*Montaje del dispositivo con el prototipo", "integración del sensor de distancia y actuadores" "Integración WIFI en el dispositivo".* Para este caso el camino que se optó de un principio fue usar un motor paso a paso para girar las compuertas (nuestro actuador) y seguir usando el sensor de distancia HC-SR04 como en la primera solución, sin embargo aquí optamos por usar el sensor wifi del modulo ESP32 ya que al intentar usar el dispositivo ESP01 con el Arduino la

comunicación con nuestro servicio web montado en el mismo no funciono, se opto por esta alternativa ya que nos percatamos que se quemo de los pines de conexión por una mala conexión de nuestra parte. Cabe destacar que cuando empezamos a usar el ESP32 tuvimos complicaciones para poder instalar los drivers ya que eran diferentes a los que se tienen en línea, pero se logró encontrar los correspondientes a la placa y con esto tener el uso completo de ella para programarla. Bien el montaje en la placa del prototipo fue bueno ya que teníamos las conexiones acordes a la información de la placa quedando de la siguiente manera:

Motor paso a paso	
PIN 18 DIG	IN1
PIN 19 DIG	IN2
PIN 12 DIG	IN3
PIN 13 DIG	IN4
VIN or 3.3v	VCC
GND	GND
sensor ultrasónico HC-SR04	
PIN DIG 2	TRIG 2
PIN DIG 15	ECHO 3
GND	GND
VIN	VCC

Con estos pines pudimos hacer correcta funcionalidad con la ESP32 y probarla con algunos códigos sencillos para ver si giraba el motor al igual que nos mostrara la distancia. No tuvimos mayor conflicto hasta que integramos con nuestro dispensador de pvc ya que el motor no tenia la suficiente fuerza para girar y cerrar la compuerta por eso mismo la solución que encontramos fue el usar servomotores sg-90 para que fueran ambos los que ayudaran a abrir y cerrar la compuerta como se muestra en la ilustración 5. Al igual que se uso una tapa y unos palos para el movimiento de la compuerta y un palo en el tope para medir la distancia de las croquetas podemos decir también que las mejoras fueron hasta diciembre donde incorporamos los tubos de pvc de esta manera para sostener el contenedor principal.

Ilustración 5 Montaje del prototipo

Concluyendo así el mes de octubre con la instalación del prototipo únicamente con los actuadores, sensor y la placa ESP32 ya las mejores para que funcionaran en conjunto fueron en diciembre.

Para el mes de noviembre fue la "*Creación de la aplicación del usuario*" empezando por el diseño del servicio web responsivo Se diseño como se muestra en la ilustración 6. Y en el mes de diciembre se diseña la pagina de login y register como en la ilustración 7 y 8. En conjunto con el diseño del logotipo de la aplicación mostrada en la ilustración 9..

Ilustración 6 Diseño del "home"

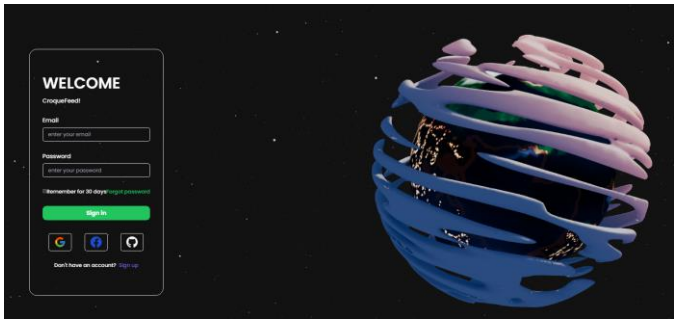


Ilustración 7 Diseño del "login"

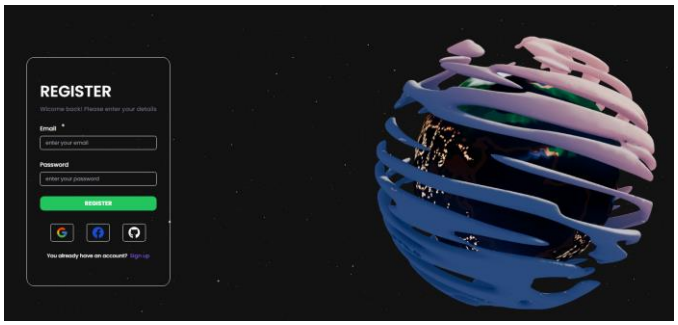


Ilustración 8 Diseño del "register"

Ilustración 9 Diseño del "logotipo"

Por esta parte no tuvimos inconvenientes para el diseño ya que desde un principio cuando nos hablaron sobre IoT lo

primero que se nos venia a la menta era el mundo cambiante que vivimos donde el internet funge alrededor de todas las personas, es inevitable no interactuar con ella por ende el diseño del espacio y estrellas para el modelo responsivo planeábamos solo dejar de primera vista el lado izquierdo para el usuario y le fuera cómodo abrirlo en el celular donde se encuentre como si fuera una app móvil. El reto viene después para la integración del servicio MVC ya que nuestro prototipo cambia totalmente, la primera dificultad que encontramos fue el entender los principios y pasarlos a nuestra planeación ya que no entendíamos como funcionaba el controlador que sería el intermediario, después los servidores de la ESP32 ya que para nosotros era funcional un sitio web en la ESP32 como en la ilustración 10. Aquí podemos observar que la pagina es sencilla pero funcional cumple con el propósito sin embargo el montar la pagina dentro de la ESP32 presentaba varias desventajas, numero 1 la conectividad era algo tardada si no encontrabas cerca del dispositivo, numero 2 las peticiones se hacían ahí mismo entonces si llegaban muchas se trababa, numero 3 no era factible para un futuro si quisiéramos mas dispositivos o si crecía la aplicación porque no aguantaría todo el diseño de la pagina web. Lo que nos lleva al diseño del servicio visto como MVC como en la ilustración 2.

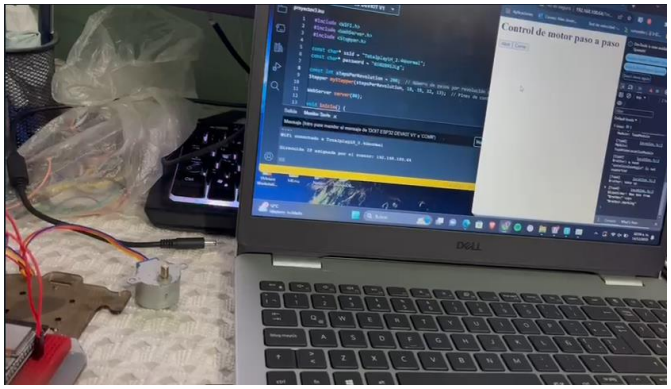


Ilustración 10"Aplicacion del ESP32 como servidor y mostrando la página web"

Las complicaciones que tuvimos en el diseño e integración del MVC fue principalmente el montar el controlador ya que nuestra primera tecnología para realizarlo era Flask como backend sin embargo la integración con está usando sockets, y rutas http no lograban vincularse con el ESP32 es decir que no recibía nada a pesar de que los servicios estaban montados en diferentes lugares. Por ende, optamos cambiar de tecnología y usar un framework cómodo para nosotros que fue node js y express js, al igual que sockets para las rutas, la facilidad que nos da express es poner las rutas de una manera sencilla, manejar las peticiones dentro de funciones, validarlas y finalmente pasarlas al servidor del ESP32 desde la vista que igual declaramos en el controlador es una ruta fija.

```
app.use(express.static(path.join(__dirname,
'../view/public/')));
app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname,
'../view/public/index.html'));
});
```

Por eso cuando abre el servicio lo que nos aparece es lo de esa ruta que es el index.html, en este índex se mandan las peticiones del usuario de la siguiente manera:

```
// Esta función hace una solicitud GET al
servidor Express
async function getData() {
    try {
        const response = await
fetch('/getDistance'); // Realiza la
solicitud GET al servidor
        const data = await response.json();
// Obtiene los datos en formato JSON

        // Muestra los datos en la página
HTML

document.getElementById('Distancia').innerTex
t = `${data.quality} cm`;
    } catch (error) {
        console.error('Error al obtener los
datos del sensor de distancia:', error);
    }

}
```

Con js mandamos la peticion a express con la ruta /getDistance ya establecida en el controlador, por eso esta dentro de una promesa que si no hay ningún error en procesar los datos nos responde con los datos en tipo JSON.

```
// Rutas GET
app.get('/getDistance', async (req, res) => {
    let time = moment().format('YYYY-MM-DD
HH:mm:ss'); // Formato de fecha y hora
deseado
    console.log(`[${time}] METHOD: GET
${req.path}`);
    try {
        const response = await
axios.get('http://192.168.100.5:8080' +
req.path);
        const qualityData = response.data; //
Suponiendo que la respuesta contiene los
datos de calidad
        time = moment().format('YYYY-MM-DD
HH:mm:ss');
        console.log(`[${time}] Distancia:
${qualityData} cm\n`);

        // Enviarlo como respuesta a la
solicitud GET
        res.json({ quality: qualityData });
    } catch (error) {
```

```
        console.error('Error al obtener la
distancia:', error);
        res.status(500).json({ error:
'Ocurrió un error al obtener la distancia'
});
    }
});
```

Y así manejos nuestras rutas establecidas dentro de estas tienen su función en particular entre estas obtener la informacion que viene del ESP32 en una ip ya fijada por la ESP32 en la red donde se encuentre en automático cuando se conecte se establece esa ip y si llegara a cambiar de red a una ditinta se tendrían que hacer las modificaciones en esta ip a la correspondiente de la ESP32. A continuación el pedazo de código mas importante de la ESP32 para el manejo de las peticiones que le llegan y como las regresa.

```
int handleWebRequests() {
  client = server.available();

  if (!client) return 0;

  Serial.println("Nuevo cliente");
  String buffer = "";
  while (client.connected()) {
    if (client.available()) {
      request = client.readStringUntil('\r');

      if (request.indexOf("GET /getDistance")
!= -1) {
        if (distanciaEnable)
sendHTTPResponse(String(readDistance()));
        else sendHTTPResponse("Not Enable");
      }

      else if (request.indexOf("GET
/getGirar") != -1) {
        if (motorEnable)
sendHTTPResponse("Girar: " +
String(abrirCompuerta()));
        else sendHTTPResponse("Not Enable");
      }
}
```

Esta función se ejecuta continuamente en un bucle y se encarga de manejar las solicitudes web que llegan al dispositivo. Verificación de Cliente Nuevo:Comprueba si hay un cliente web nuevo esperando a ser atendido. Un cliente web podría ser un navegador u otro dispositivo que se conecta al ESP32.Manejo del Cliente:Si hay un cliente, se inicia el manejo y se imprime en la consola "Nuevo cliente". Lectura de la Solicitud del Cliente: Se lee la solicitud completa del cliente hasta que se encuentra un retorno de carro ('\r'), lo que indica el final de la solicitud. Procesamiento de Solicitudes:

La función verifica el contenido de la solicitud para determinar qué acción realizar.Si la solicitud contiene "GET /getDistance", y la variable distanciaEnable está activada, envía una respuesta HTTP con la distancia medida por un sensor ultrasónico.Si la solicitud contiene "GET /getGirar", y la variable motorEnable está activada, envía una respuesta HTTP informando sobre el giro de un motor o servo.

```
 while (WiFi.status() != WL_CONNECTED &&
counterConnection < 10) {
    delay(500);
    Serial.print(".");
    counterConnection++;
  }

  if (counterConnection < 10) {
    IPAddress ip(192,168,100,5);
    IPAddress gateway(192,168,100,1);
    IPAddress subnet(255,255,255,0);
    WiFi.config(ip, gateway, subnet);

    Serial.println("");
    Serial.println("Conectado al WiFi");

    server.begin();
    Serial.print("API ESP32 desplegado en la
ip 'http://");
    Serial.print(WiFi.localIP());
    Serial.println(":8080'");
  } else {
    Serial.println("");
  }
```

Intento de Conexión WiFi: La instrucción WiFi.status() != WL_CONNECTED verifica si el ESP32 no está conectado a una red WiFi. counterConnection es un contador que se utiliza para limitar el número de intentos de conexión. La condición counterConnection < 10 asegura que no se intente la conexión más de 10 veces.
Bucle de Conexión:
Mientras el ESP32 no esté conectado y no se haya intentado conectar más de 10 veces, el código se queda en un bucle. En cada iteración, espera 500 milisegundos (delay(500)) y muestra un punto en la consola (Serial.print(".")). También incrementa el contador de conexiones (counterConnection++).
Manejo del Resultado de Conexión: Después de salir del bucle, se verifica si el contador de conexiones (counterConnection) es menor que 10.
Si es menor, significa que la conexión fue exitosa en algún momento de los intentos anteriores.
En este caso, se configuran manualmente la dirección IP, puerta de enlace y máscara de subred utilizando WiFi.config.
Se imprime en la consola que el dispositivo está conectado y se inicia un servidor con server.begin().
Se muestra la dirección IP asignada al ESP32 en la consola.
Fracaso de Conexión: Si el contador de conexiones es igual o

mayor que 10, se imprime un mensaje indicando que la conexión ha fallado.
Bien con esto logramos que funcionara correctamente nuestro MVC, de principio la implementación fue demasiado tardada por la documentación de las librerías, al igual que en la ESP32 para la integración del servo nosotros teníamos una biblioteca general que funcionaba para el arduino sin embargo después esta librería no fue soportada por la ESP32 por lo que tuvimos que buscar alternativas de bibliotecas entre ellas había una exclusiva del microcontrolador por lo que la lógica seguía siendo la misma entonces pudimos solucionar esa parte, después en la integración del servicio en express y esp32 tuvimos errores ya que cuando configuramos la ip la mascara por defecto no nos salía pero después de reiniciar nuestro modem pudimos identificar cual era y con esto lograr la comunicación en ambos por lo que fue funcional como en la ilustración 11.

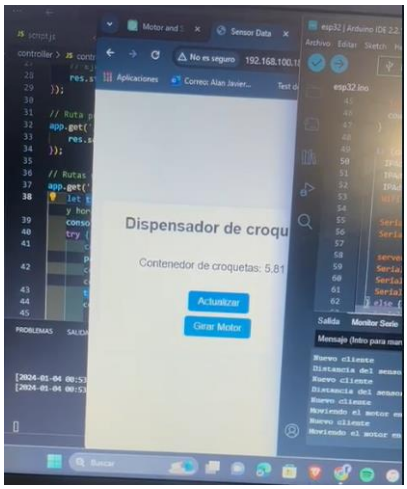
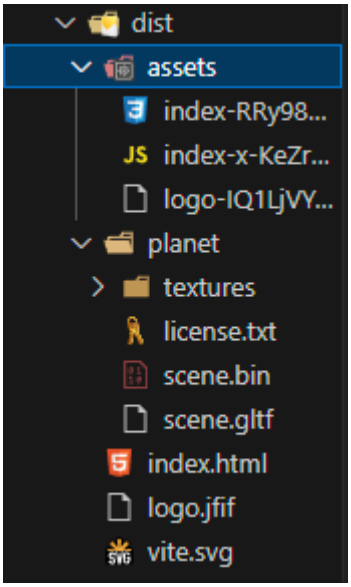


Ilustración 11 MVC version 1 sin css

Sin embargo, en nuestra opción era muy simple para nuestros usuarios la vista ya que teníamos un diseño ya previsto decidimos adaptarlo en la vista pero requeríamos algo mas estructurado para poder plasmarla por lo que las herramientas en este caso fue React y tailwind css, con lo que anteriormente ya habíamos trabajado nos daba la seguridad de lógralo, cuando creamos el proyecto quisimos optimizarlo con vite y que no estuviera tan pesado para subirlo a producción. Entre algunos problemas que nos encontramos al principio de la programación fue el que fuera responsivo ya que la idea de meter el planeta como un diseño 3D fue hacer la plantilla de este y animarlo para que se hiciera bola. Sin embargo, el diseño era tan pesado que usaba todos los recursos, para solucionarlo únicamente se renderizo a una calidad menor, las estrellas realmente eran estáticas, pero se implementó una función para que se movieran de manera aleatoria. Después para que fuera responsiva lo dividimos en dos si están en pantallas grandes se mostrará el planeta y si están en dispositivos móviles únicamente la funcionalidad de los botones y una serie de instrucciones como en la ilustración 6. Así logramos que fuera responsiva aparte de usar flex en la parte izquierda, aquí no tuvimos mayor problema ya que la lógica que teníamos de los botones era la misma con las rutas para las acciones, y los errores las manejábamos en la misma. De primera instancia se veía bien sin embargo nuestra lógica del contenedor aun no estaba implementada donde el usuario le saltaría la alerta de cuantas croquetas tiene el contenedor por es mismo optamos por implementar un modal que saltara como notificación, usamos una categoría de colores donde rojo es que esta casi vacío o vacío, amarillo si esta bien el contendor, y verde que es lleno. La lógica fue el dato lo guardamos en una variable local con useState el hook de react pudimos interactuar con el y pasar el valor al modal para que hiciera las validaciones, cambiara el color y finalmente su respectivo botón de cerrar, la complicación con el modal fue sobre todo el que no salía en pantalla con el botón es decir que como estaba en la promesa de que el dato del dispensador llegaría no la mostraba por ende por uso de pruebas lo pusimos fuera de la promesa para verificar y la sorpresa que nos llevamos es que no cambiaba de color pero fue fácil de solucionar ya que como el diseño estaba únicamente en blanca lo hicimos dinámico como si pasara un valor a la clase, es decir que el blanco ya no seria fijo para el modal, también por comodidad la lógica del modal se implementó en otro archivo. Usamos la misma lógica de este modal para la notificación de que ya se abrió la compuerta y esto mas que nada para lo lógica de nuestro servicio, para que no se trabaran los motores de tanto presionar el botón de abrir se les dio un time de espera para que no pudieran interactuar con la pagina esto lo logramos con un contado y un bloqueo total de la ventana a unos segundos con eso solucionamos nuestra lógica de servicio.
Finalmente el reto también era ponerle esta capa de seguridad extra a nuestra página web por eso implementamos nuestro login y register con firebase de Google ya que es muy sencillo de usar, las credenciales que nos da al crear el servicio como tokens solo había que ponerlas en un archivo y instalar algunas librerías extras que nos proporcionan una lógica más simple para la interacción del inicio de sesión aquí el reto fue el redireccionamiento de la pagina en caso de que se validara o simplemente fuera un error, pero a pesar de que nos llevo tiempo pudimos hacerlo por las librerías solo teníamos que validar los datos. Dado que teníamos una plantilla del home usamos la misma para el login y register no hubo margen de error mas que acomodar bien nuestro formulario. El reto más complicado fue implementar la vista desde el controlador ya que a pesar de que la ruta era muy parecida, tal cual el index de la primera versión, no mostraba nada. Para solucionar esta parte de que la vista no mostraba nada con react buscamos que era porque no estaba en producción y había demasiados archivos para cargar entonces tuvimos que compilar y crear una carpeta dist que tuviera todo lo funcional de la pagina esto con un comando de node, una ves creada esta ruta tuvimos que cambiar la del controlador es decir la vista ya no iba ser bublic si no dist como a continuación:

```
app.use(express.static(path.join(__dirname,
'../view/public/dist/')));
```

En esa carpeta se encontraría la vista compilada lista para mostrarse con los siguientes archivos:



Donde podemos encontrar el planeta que usamos para la animación 3D, nuestro logo y las carpetas necesarias de lógica, diseño y el index que extrae esa información compilada como si estuviera comprimida y cifrada hasta cierto punto, esto nos permitió cambiar la vista simple a algo más amigable para los usuarios. A continuación, en la ilustración 12 se muestra todo el proyecto de funcionamiento.

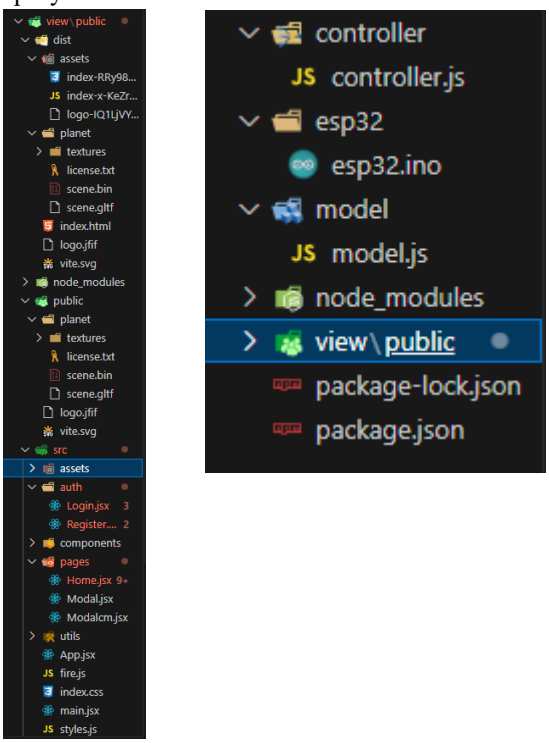


Ilustración 12 MVC version 2 del proyecto completo

Pruebas y Validación

Para nuestras pruebas primero validamos el registro correcto de usuarios y el login esto registrando dos usuarios a firebase como se muestra en la ilustración 13, se logra apreciar dos usuarios que registramos desde el servicio web al igual que con sus contraseñas cifradas esto quiere decir que la configuración de la identificación fue correcta sin embargo al principio el registrase había más parámetros, pero los delimitamos por tiempos de implementación.

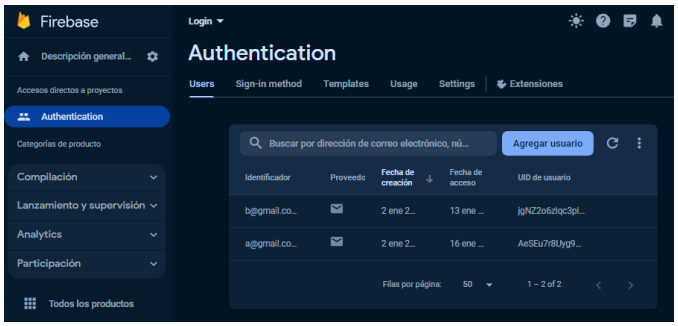


Ilustración 13 validación de usuarios

Para la Pruebas de hardware: en el funcionamiento del Dispensador: Verificamos que el dispensador realizara correctamente las funciones de notificación de nivel y apertura de compuerta. Se realizaron pruebas exhaustivas para garantizar la correcta sincronización entre el sensor de distancia HC-SR04, los motores paso a paso y los servomotores SG-90.
Aseguramos que la compuerta se abriera y cerrara adecuadamente según las señales del sistema. También aquí mismo se realizaron pruebas de conexión Wifi con el módulo ESP32 para asegurar la estabilidad y la correcta integración con el servicio web.
Pruebas de Software:

Aplicación del Usuario:
Se llevaron a cabo pruebas en la aplicación móvil para verificar la correcta emisión de notificaciones al usuario sobre el estado del dispensador.
Se validó la interfaz de usuario en busca de posibles mejoras en diseño y usabilidad.
Servicio Web y MVC:
Realizamos pruebas de integración del servicio web en el ESP32 utilizando el framework Express.js y Node.js.
Validamos la funcionalidad del Modelo-Vista-Controlador (MVC) para garantizar la comunicación efectiva entre el frontend y el backend.
Se realizaron pruebas de manejo de solicitudes HTTP para confirmar la obtención correcta de datos del dispensador y la capacidad de girar la compuerta.
Pruebas de Usabilidad:

Interfaz Responsiva:
Verificamos la adaptabilidad de la interfaz web en diferentes dispositivos, asegurándonos de que fuera fácil de usar tanto en pantallas grandes como en dispositivos móviles.
Pruebas de Notificación:
Alertas y Modalidades:
Realizamos pruebas en la lógica de notificación para confirmar que las alertas sobre el nivel del dispensador se mostraran correctamente en el modal de la interfaz web.

Validamos la implementación de modalidades de colores para indicar el estado del dispensador (rojo para casi vacío, amarillo para nivel adecuado, verde para lleno).
Pruebas de Rendimiento:
Optimización de Recursos:
Implementamos pruebas de rendimiento para optimizar la carga de recursos en la interfaz web, especialmente al utilizar React y Tailwind CSS.
Verificamos que la compilación de la vista en la carpeta "dist" permitiera una carga eficiente de la página.
Pruebas de Escalabilidad:
Manejo de Volumen de Usuarios:
Validamos la capacidad del sistema para manejar múltiples conexiones y solicitudes simultáneas, garantizando que fuera escalable para un mayor número de usuarios.
El conjunto de estas pruebas y validaciones contribuyó al desarrollo de un proyecto robusto, funcional y confiable, asegurando una experiencia positiva tanto para los usuarios finales como para los desarrolladores y responsables del mantenimiento.

Resultados y Conclusiones

La funcionalidad de notificación en tiempo real ha demostrado ser efectiva al alertar a los usuarios cuando el nivel de croquetas es bajo. Esto permite una respuesta inmediata por parte del cuidador, asegurando que las mascotas reciban alimentación adecuada y oportuna.
Seguridad y Control de Acceso
La implementación de Firebase para la autenticación garantiza un nivel adicional de seguridad en el acceso a la aplicación web. Los usuarios pueden tener la tranquilidad de que solo personas autorizadas pueden monitorear y controlar la alimentación de sus mascotas.
Consumo Eficiente de Recursos
El sistema ha demostrado un consumo eficiente de recursos, tanto en términos de hardware como de ancho de banda. La optimización en el uso de recursos contribuye a un funcionamiento fluido y confiable del sistema.

Logros del Proyecto

- **Automatización Efectiva:** La automatización del dispensador de croquetas mediante el uso de la placa ESP32, sensor ultrasónico y servomotores ha demostrado ser eficiente y confiable.
- **Control Remoto y Monitoreo:** La aplicación web permite a los usuarios tener un control remoto preciso sobre la alimentación de sus mascotas y monitorear el nivel de croquetas en tiempo real.
- **Notificaciones Oportunas:** Las notificaciones en tiempo real han mejorado significativamente la capacidad de los cuidadores para responder rápidamente a las necesidades alimentarias de sus mascotas.

Reflexiones sobre Mejoras Futuras

- **Expansión de Funcionalidades:** Se podría considerar la expansión de funcionalidades, como la programación de horarios de alimentación y la integración con dispositivos de monitoreo de salud para proporcionar un cuidado más completo.
- **Optimización de Energía:** Explorar opciones para optimizar el consumo de energía, especialmente en el modo de espera, para hacer el sistema aún más eficiente y sostenible.
- **Interfaz de Usuario Avanzada:** Mejorar la interfaz de usuario con características avanzadas, como informes de consumo de alimentos a lo largo del tiempo y sugerencias nutricionales basadas en patrones de alimentación.
- **Integrar una base de datos:** Para futuras consultas y que ayuden a mas personas a integrarse al proyecto y hacer estudios sobre la viabilidad del proyecto para mas personas con mascotas.

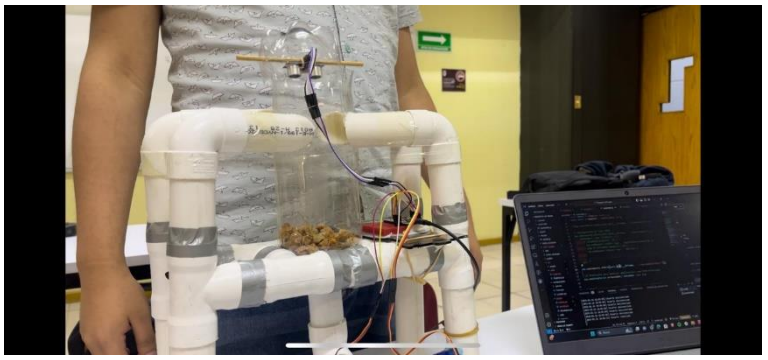


Ilustración 14. Proyecto terminado con las pruebas

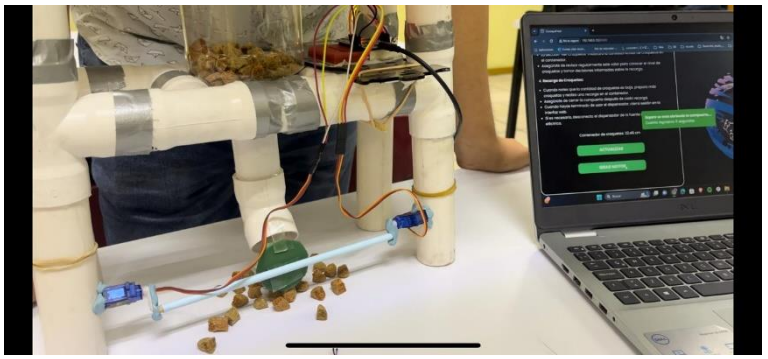


Ilustración 15. Proyecto terminado con las pruebas

Para este proyecto se tuvieron varios conflictos desde que iniciamos , sin embargo a nuestro tiempo se fueron solucionando con investigación adicional, al igual que podemos decir que toma caminos distintos a la idea principal que teníamos sin embargo cumple con nuestro objetivo , podemos decir que el proyecto "CroqueFeed: Sistema de Alimentación Automatizada" representa un paso significativo hacia la mejora de la calidad de vida de las mascotas y la simplificación de las responsabilidades de sus cuidadores. La solución implementada ha demostrado eficacia en varios aspectos clave, proporcionando resultados alentadores. La eficiencia operativa del sistema es evidente en la automatización precisa del dispensador de croquetas. La integración de la placa ESP32, el sensor ultrasónico HC-SR04 y los servomotores SG-90 ha permitido una dispensación

controlada y confiable de alimentos. Esta eficiencia se traduce en una alimentación constante y adecuada para las mascotas, incluso en ausencia de sus cuidadores.

La capacidad de acceso remoto y monitoreo en tiempo real a través de la aplicación web ha sido fundamental. La interfaz desarrollada con tecnologías modernas ha proporcionado a los usuarios un control intuitivo y una visibilidad instantánea sobre el nivel de croquetas. La respuesta ágil de la aplicación contribuye a una experiencia de usuario fluida y amigable.

La implementación de notificaciones en tiempo real, basadas en el sensor ultrasónico, aborda una preocupación fundamental: la falta de monitoreo constante. Los cuidadores reciben alertas oportunas sobre la necesidad de rellenar el dispensador, garantizando una respuesta inmediata y mitigando la posibilidad de interrupciones en la alimentación de las mascotas.

En términos de seguridad y privacidad, la inclusión de Firebase para la autenticación ha fortalecido la protección de datos sensibles. La arquitectura del sistema ha sido diseñada con un enfoque en la seguridad, asegurando que solo usuarios autorizados tengan acceso a la aplicación web y, por ende, al control del dispensador.

Además de los logros alcanzados, el proyecto deja espacio para futuras mejoras. La expansión de funcionalidades, como la programación de horarios de alimentación y la integración con dispositivos de monitoreo de salud, podría enriquecer aún más la experiencia del usuario. Asimismo, se podría explorar la optimización de la eficiencia energética y la mejora continua de la interfaz de usuario.

En conclusión, "CroqueFeed" no solo cumple con su objetivo de ofrecer una solución tecnológica para el cuidado de mascotas, sino que también sienta las bases para innovaciones futuras.

10. Referencias

[1]Actuadores eléctricos | ¿Qué son y cómo funcionan? | SDI. (s.f.). SDI. https://sdindustrial.com.mx/blog/introduccion-a-los-actuadores-electricos-motores-electricos/

[2]El Internet de las Cosas en el mundo animal para la protección de la biodiversidad. (s.f.). Blogthinkbig.com. https://blogthinkbig.com/internet-de-las-cosas-en-el-mundo-animal

[3] "Benefits of an Automatic Pet Feeder," PetSafe®, 2015. [Online]. Available.

[4] R. Escott, "Top 5 benefits of automatic pet feeders," Closer Pets, Mar. 17, 2021. [Online]. Available:.

[5] "What Are The Benefits Of An Automatic/Smart Pet Feeder?," INSTACHEW, Feb. 9, 2023. [Online]. Available.

[6] "ESP32 for IoT: A Complete Guide," Nabto. [Online]. Available.

[7] S. Boral, "What is ESP32 and Why Is It Best for IoT Projects?," IoT Tech Trends, May 25, 2020. [Online]. Available.

[8] "IoT Pet Tech Solutions: the Future in Smart Technologies for Pets," Cogniteq, Mar. 30, 2022. [Online]. Available.

[9] R. Isaacs, M. Diaz, and A. Murray, "The best automatic pet feeders of 2024," ZDNET, Dec. 7, 2023. [Online]. Available.

[10] "The best automatic pet feeders in 2023," Popular Science. [Online]. Available: .

[11] "Automatic pet feeder," AIP Conference Proceedings, AIP Publishing, Jul. 24, 2023. [Online]. Available.