



Proyecto: Dispensador Inteligente de Azúcar

Curso: Prototipos IoT con Raspberry

Alumno: Pere Martin

Fecha: 05/12/2024



## Introducción

El “**Dispensador Inteligente de Azúcar**” es un sistema diseñado para dispensar azúcar de manera automatizada y personalizada, utilizando tecnologías como el reconocimiento facial, sensores, motores controlados por PWM y una base de datos para registrar y gestionar las preferencias y actividades de los usuarios. Este proyecto busca no solo automatizar una tarea rutinaria, sino también explorar cómo la personalización y la tecnología pueden mejorar la experiencia del usuario en su vida diaria.

A lo largo del desarrollo, me encontré con diversos desafíos técnicos y de diseño, que abordé con un enfoque modular tanto en el hardware como en el software. El objetivo final fue crear un sistema que pudiera reconocer a los usuarios, identificar sus preferencias almacenadas y dispensar la cantidad exacta de azúcar deseada en función de esas configuraciones.

## Motivación del Proyecto

La idea detrás del proyecto fue llevar la personalización a un contexto práctico mediante el uso de tecnologías de inteligencia artificial e Internet de las cosas (IoT). En un mundo donde la experiencia personalizada se está convirtiendo en un estándar, integrar sistemas de reconocimiento facial con dispositivos cotidianos abre un abanico de posibilidades. Por ejemplo:

- Personalizar la cantidad de azúcar en bebidas según las preferencias individuales.
- Ajustar automáticamente la cantidad de café en una máquina.
- Extender este modelo a otras áreas como el control de porciones de alimentos o ingredientes en cocinas automatizadas.

Este proyecto no solo tiene un enfoque funcional, sino que también sirve como una plataforma para explorar cómo las tecnologías emergentes pueden integrarse en entornos cotidianos. Además, proporciona una base sólida para futuras mejoras, como incluir nuevos ingredientes o expandir las capacidades del sistema.

## **Estructura General del Sistema**

El sistema está compuesto por varios módulos principales, cada uno con una funcionalidad específica que, al combinarse, permiten el funcionamiento integral del dispensador. A continuación, se describen los elementos clave:

### **1. Reconocimiento Facial**

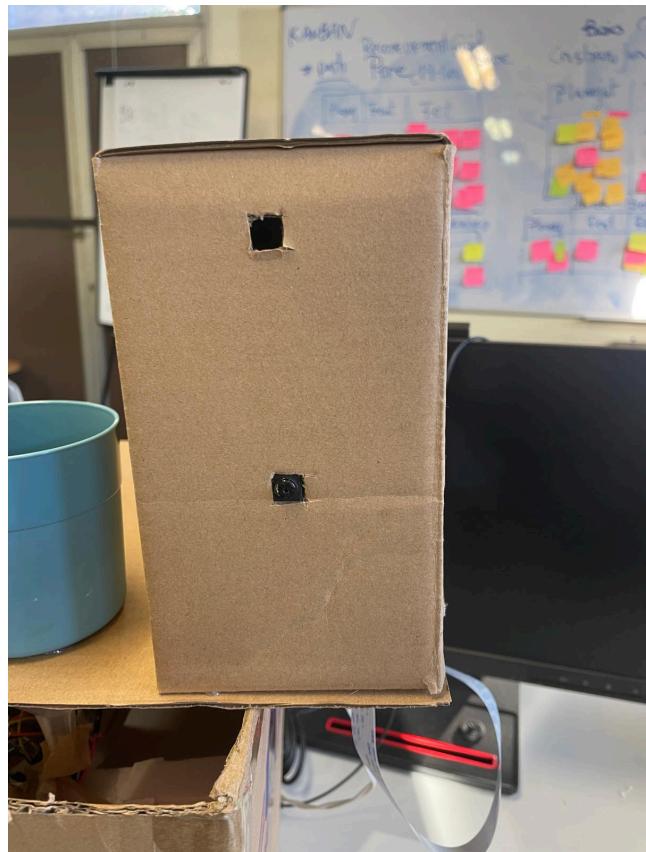
El reconocimiento facial es el núcleo del sistema. Utiliza una cámara conectada a una Raspberry Pi que identifica al usuario mediante librerías como **dlib** y **OpenCV**. Cada usuario tiene un perfil que incluye:

- Su encoding facial (generado por dlib).
- Sus preferencias de tipo de azúcar y cantidad (almacenadas en la base de datos).

Cuando el sistema detecta a un usuario registrado, carga automáticamente sus preferencias, simplificando el proceso y eliminando la necesidad de introducir datos manualmente.

### **Desafíos en esta parte:**

Implementar dlib en la Raspberry Pi fue complejo debido a las limitaciones de recursos del dispositivo. Esto me llevó a crear una imagen Debian optimizada para incluir las dependencias necesarias y reducir la carga de procesamiento.



## 2. Interfaz de Usuario con Pulsadores y Pantalla LCD

La interacción con el usuario se realiza mediante pulsadores y una pantalla LCD conectada mediante el protocolo I2C. Esta pantalla muestra mensajes en tiempo real, como:

- Instrucciones iniciales: "Sistema listo. Reconociendo usuario."
- Solicitudes de entrada: "Ingrese número de cucharadas (en gramos): \_".
- Notificaciones: "Dispensando azúcar moreno..." o "Coloque una taza para continuar."

El sistema permite que el usuario ajuste manualmente la cantidad de azúcar utilizando los pulsadores si lo desea. Sin embargo, esta funcionalidad quedó parcialmente desarrollada para registrar nuevos usuarios y configuraciones completas desde la interfaz física.



### 3. Sensores Infrarrojos

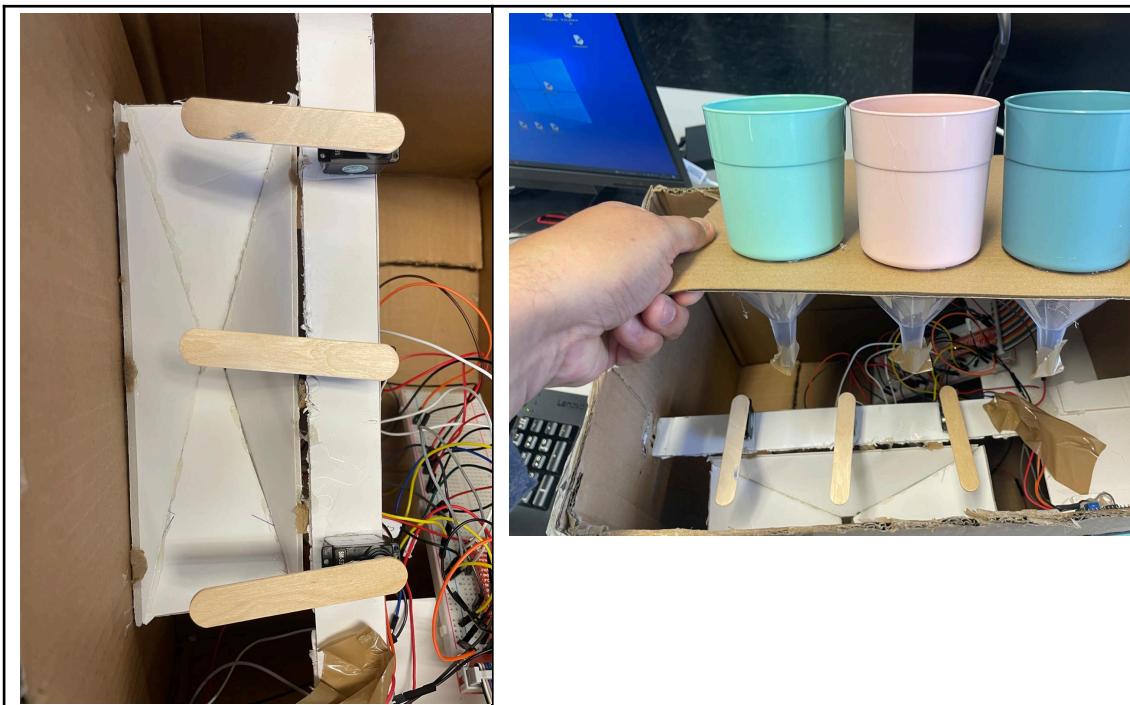
Para garantizar que el azúcar no se dispense sin un recipiente debajo, se integró un sensor infrarrojo. Este componente detecta la presencia de una taza bajo el dispensador y, si no está presente, bloquea el proceso y muestra un mensaje de error en la pantalla LCD. Esto asegura un uso eficiente del sistema y evita desperdicios.



#### 4. Motores DC y Control de Dispensado

Cada tipo de azúcar está asociado a un motor DC independiente, controlado mediante señales PWM. Estos motores regulan una trampilla que dispensa el azúcar desde un embudo hacia el recipiente. La cantidad dispensada se calcula en función del tiempo de activación del motor, lo que permite una dosificación precisa.

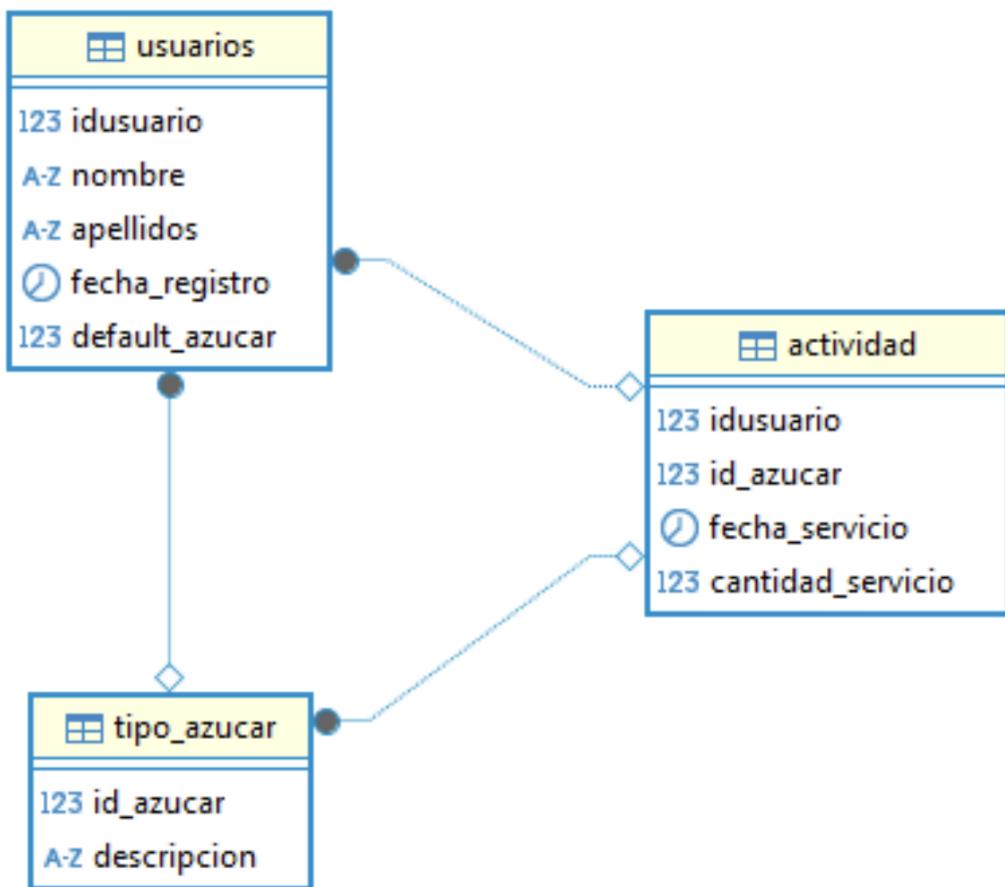
Sin embargo, uno de los mayores problemas fue lograr la alineación exacta entre el embudo y la trampilla del motor. La estructura inicial, hecha de cartón, no proporcionaba la rigidez necesaria para mantener un espacio constante de **1 mm**, lo que resultaba en pérdidas de azúcar o cantidades inexactas. Este problema subrayó la necesidad de usar materiales más robustos en futuras versiones.



## Base de Datos en MariaDB

La base de datos es el corazón del sistema, donde se almacena toda la información relacionada con los usuarios, los tipos de azúcar y las actividades realizadas. Utilicé MariaDB en la Raspberry Pi y la gestioné desde Python mediante mysql-connector.

### Estructura de la Base de Datos



#### 1. **usuarios:**

Almacena los datos básicos de los usuarios, como su nombre, apellidos, fecha de registro y su tipo de azúcar predeterminado.

#### 2. **tipo\_azucar:**

Contiene los tipos de azúcar disponibles en el sistema (Blanco, Moreno, Edulcorante).

#### 3. **actividad:**

Registra cada acción realizada en el dispensador, incluyendo el usuario que lo utilizó, el tipo de azúcar dispensado, la fecha y la cantidad servida.

## **Clases Utilizadas en el Proyecto**

El diseño del software es modular, lo que facilita el mantenimiento y las futuras expansiones. A continuación, se describen las principales clases implementadas:

### **1. FaceRecognitionClass:**

- Gestiona el reconocimiento facial utilizando dlib y OpenCV.
- Genera y almacena encodings faciales en la base de datos.
- Se encarga de identificar al usuario al inicio del proceso.

### **2. LCD\_I2C:**

- Controla la pantalla LCD conectada mediante el protocolo I2C.
- Muestra mensajes en tiempo real y actúa como el principal medio de comunicación con el usuario.

### **3. SensorInfrarrojo:**

- Detecta la presencia de un recipiente bajo el dispensador.
- Envía señales al sistema para permitir o bloquear el dispensado.

### **4. MotorDC:**

- Controla los motores responsables de abrir y cerrar las trampillas de los embudos.
- Utiliza PWM para dosificar con precisión la cantidad de azúcar.

### **5. ContenedorClass:**

- Supervisa el estado lleno/vacío de los contenedores de azúcar.
- Registra cada dispensado en la base de datos y emite alertas cuando el nivel es bajo.

### **6. EntradaDatos:**

- Permite la interacción del usuario mediante pulsadores.
- Facilita la introducción de datos como el tipo y cantidad de azúcar deseada.

## Problemas y Desafíos Durante el Desarrollo

### 1. Instalación de Dlib

Uno de los primeros retos importantes del proyecto fue la instalación de **dlib** en la Raspberry Pi, una librería esencial para gestionar el reconocimiento facial. Aunque sabía que la Raspberry Pi 3 podía ser una limitación en términos de recursos, no anticipé la cantidad de problemas que surgirían.

El proceso inicial de instalación requería la compilación manual de **dlib** y sus dependencias. Aquí enfrenté varios problemas:

- **Falta de memoria RAM:** La Raspberry Pi no tenía suficiente memoria para completar la compilación en una sola ejecución. Esto obligó a realizar configuraciones adicionales, como habilitar el uso de memoria swap, lo cual ralentizó significativamente el proceso.
- **Dependencias rotas:** Algunas librerías necesarias para dlib no estaban disponibles directamente en la imagen predeterminada de Debian para Raspberry Pi, lo que requería instalar manualmente versiones específicas o compilar dependencias adicionales.
- **Errores de configuración:** Incluso después de completar la instalación, surgían problemas de compatibilidad entre las versiones de dlib, OpenCV y Python.

Después de varios intentos fallidos, opté por una solución más robusta: **crear una nueva imagen Debian personalizada**. Esta nueva imagen incluyó:

- Solo las librerías esenciales para reducir la carga del sistema operativo.
- Configuración manual del entorno de desarrollo para garantizar la compatibilidad entre OpenCV y dlib.
- Una reducción en la resolución de las imágenes procesadas (640x480 px) para aligerar el uso de recursos y optimizar el rendimiento del reconocimiento facial.

Este proceso no solo retrasó el proyecto, sino que también me enseñó una lección importante: **la preparación del entorno es crucial en proyectos que involucran tecnologías de alto rendimiento**. Dedicar más tiempo inicialmente a evaluar las necesidades y limitaciones del hardware habría evitado muchas de estas complicaciones.

## 2. Entrada de Datos

Otro aspecto del proyecto que quedó incompleto fue la implementación de una entrada de datos completamente funcional mediante los pulsadores y la pantalla LCD. La idea inicial era que el usuario pudiera introducir información como:

- Su nombre, durante el registro de nuevos perfiles.
- Preferencias de tipo y cantidad de azúcar.
- Ajustes personalizados directamente desde la interfaz física.

Aunque se lograron avances básicos, como la selección de cantidades y tipos de azúcar mediante los pulsadores, la lógica necesaria para registrar nuevos usuarios o modificar configuraciones quedó sin completar. Las razones principales fueron:

- **Falta de tiempo:** La implementación de otras partes del sistema, como el reconocimiento facial y la integración de la base de datos, demandaron mucho más tiempo del previsto.
- **Complejidad de la interfaz:** Diseñar una interfaz fluida utilizando solo pulsadores y la pantalla LCD fue un desafío. Al no contar con un teclado físico, la entrada de texto era especialmente complicada, y la lógica necesaria para navegar entre opciones y validar datos no se completó.

Este problema me enseñó a priorizar las funcionalidades clave del proyecto. Aunque hubiera sido ideal completar la entrada de datos desde los pulsadores, fue más importante asegurar el correcto funcionamiento del dispensador y del reconocimiento facial. En futuras iteraciones, podría considerar agregar una pantalla táctil para simplificar la interacción y ofrecer una experiencia más intuitiva.

### 3. Diseño Mecánico

El desafío más grande, y posiblemente el más frustrante, fue la construcción física del prototipo. Inicialmente, utilicé cartón para crear la estructura del dispensador debido a su bajo costo y facilidad de manipulación. Sin embargo, este material resultó inadecuado para las necesidades del sistema.

El problema principal fue la falta de **rigidez** del cartón, que provocó:

- **Desalineación entre los embudos y los motores:** La estructura no era capaz de mantener el espacio de **1 mm** necesario entre el embudo y la trampilla del motor. Esto generaba pérdidas de azúcar y dificultades para dispensar cantidades precisas.
- **Deformación durante las pruebas:** A medida que realizaba pruebas, el cartón se deformaba, lo que hacía que la estructura perdiera estabilidad y afectaba directamente el rendimiento de los motores.
- **Dificultades en el montaje:** Aunque inicialmente parecía sencillo trabajar con cartón, reforzar la estructura y ajustar manualmente los componentes consumió mucho más tiempo del previsto.

Este problema evidenció la importancia de elegir los materiales correctos, incluso en las etapas iniciales del proyecto. Para futuras versiones, usar madera o plástico sería una solución más adecuada, ya que estos materiales ofrecen mayor resistencia y estabilidad. Además, podría considerar diseñar soportes específicos para los motores y embudos, asegurando una alineación precisa sin depender del material estructural.

A pesar de estos contratiempos, la experiencia fue extremadamente valiosa. Me permitió comprender mejor los desafíos prácticos de integrar hardware y software en un proyecto real, así como la importancia de equilibrar los aspectos técnicos con los físicos. Este reto destacó la necesidad de planificar más detalladamente la parte mecánica en proyectos futuros y no subestimar el tiempo necesario para el diseño y construcción de prototipos físicos.

## **Reflexión General sobre los Problemas**

Cada uno de estos desafíos fue una oportunidad para aprender y mejorar mis habilidades tanto técnicas como de planificación. La instalación de dlib reforzó la importancia de preparar adecuadamente el entorno de desarrollo. El diseño incompleto de la entrada de datos me mostró la necesidad de priorizar las funciones más importantes y reservar tiempo suficiente para la implementación de características secundarias. Finalmente, las dificultades en la construcción física subrayaron la relevancia de elegir los materiales adecuados desde el principio, especialmente en proyectos donde la precisión mecánica es fundamental.

Aunque estas áreas presentaron limitaciones en el desarrollo del proyecto, también dejaron claro el potencial para futuras mejoras. Este proyecto no solo me ayudó a aplicar conocimientos técnicos, sino también a reflexionar sobre cómo abordar problemas similares en el futuro de una manera más eficiente y planificada.