Índice

[Índice 1](#_Toc151135736)

[Tablas 1](#_Toc151135737)

[Ilustraciones 1](#_Toc151135738)

[Esquemas 2](#_Toc151135739)

[Glosario de Términos: 2](#_Toc151135740)

[**CAPÍTULO III – “**DISEÑO E IMPLEMENTACIÓN DEL SISTEMA**”** 2](#_Toc151135741)

[3.1 Diseño Sistema General 2](#_Toc151135742)

[3.2 Dispositivo de Laboratorio 3](#_Toc151135743)

[3.2.1 Comunicación con el Sensor 4](#_Toc151135744)

[3.2.2 Arquitectura de la Interfaz Visual 4](#_Toc151135745)

[3.2.3 Arquitectura Base de Datos 6](#_Toc151135746)

[3.2.4 Transferencia de datos 7](#_Toc151135747)

[~~3.2.5 Despliegue de modelos en la nube~~ 7](#_Toc151135748)

[3.3 Dispositivo de Campo 7](#_Toc151135749)

[3.3.1 Interacción con el Sensor 8](#_Toc151135750)

[3.3.2 Arquitectura de Clases 8](#_Toc151135751)

[3.3.3 Pagina Web 10](#_Toc151135752)

[~~3.3.4 Almacenamiento permanente~~ 11](#_Toc151135753)

[~~3.3.5 Actualización de modelos~~ 11](#_Toc151135754)

[3.4 Utilización de Redes Neuronales Artificiales 11](#_Toc151135755)

[3.4.1 Arquitectura de Red Neuronal 11](#_Toc151135756)

[3.4.2 Pruebas en Laboratorio 11](#_Toc151135757)

[3.4.3 Pruebas en Campo 11](#_Toc151135758)

[Referencias 11](#_Toc151135759)

[Anexos 11](#_Toc151135760)

# Tablas

[Tabla 1: Formato CSV 7](#_Toc151135761)

# Ilustraciones

[Ilustración 1: Vita de usuarios 5](#_Toc151135762)

[Ilustración 2: Interfaz visual con barral lateral 6](#_Toc151135763)

[Ilustración 3: Página Web Desplegada por ESP32 10](#_Toc151135764)

[Ilustración 4: Interfaces de actualización de modelo en ESP32 10](#_Toc151135765)

# Esquemas

[Esquema 1: Arquitectura General del Sistema. 3](#_Toc151135766)

[Esquema 2: Conexión DL-Sensor 4](#_Toc151135767)

[Esquema 3: Interfaces Web del dispositivo de laboratorio 5](#_Toc151135768)

[Esquema 4: Arquitectura de Base de Datos Relacional 7](#_Toc151135769)

[Esquema 5: Conexión DC-Sensor 8](#_Toc151135770)

[Esquema 6: Flujo de Código del ESP32 9](#_Toc151135771)

Glosario de Términos:

* DL: Dispositivo de Laboratorio
* DC: Dispositivos de Campo
* CSV (Comma Separated Values): Valores Separados por Coma. Es un estándar de formato de documento para representar en forma de tabla la información

# **CAPÍTULO III – “**DISEÑO E IMPLEMENTACIÓN DEL SISTEMA**”**

El diseño e implementación de un sistema para detectar adulteraciones en la leche requiere de una correcta implementación tanto del hardware como del software. Ambos componentes son fundamentales para lograr resultados precisos y confiables en la detección de adulterantes en este importante producto.

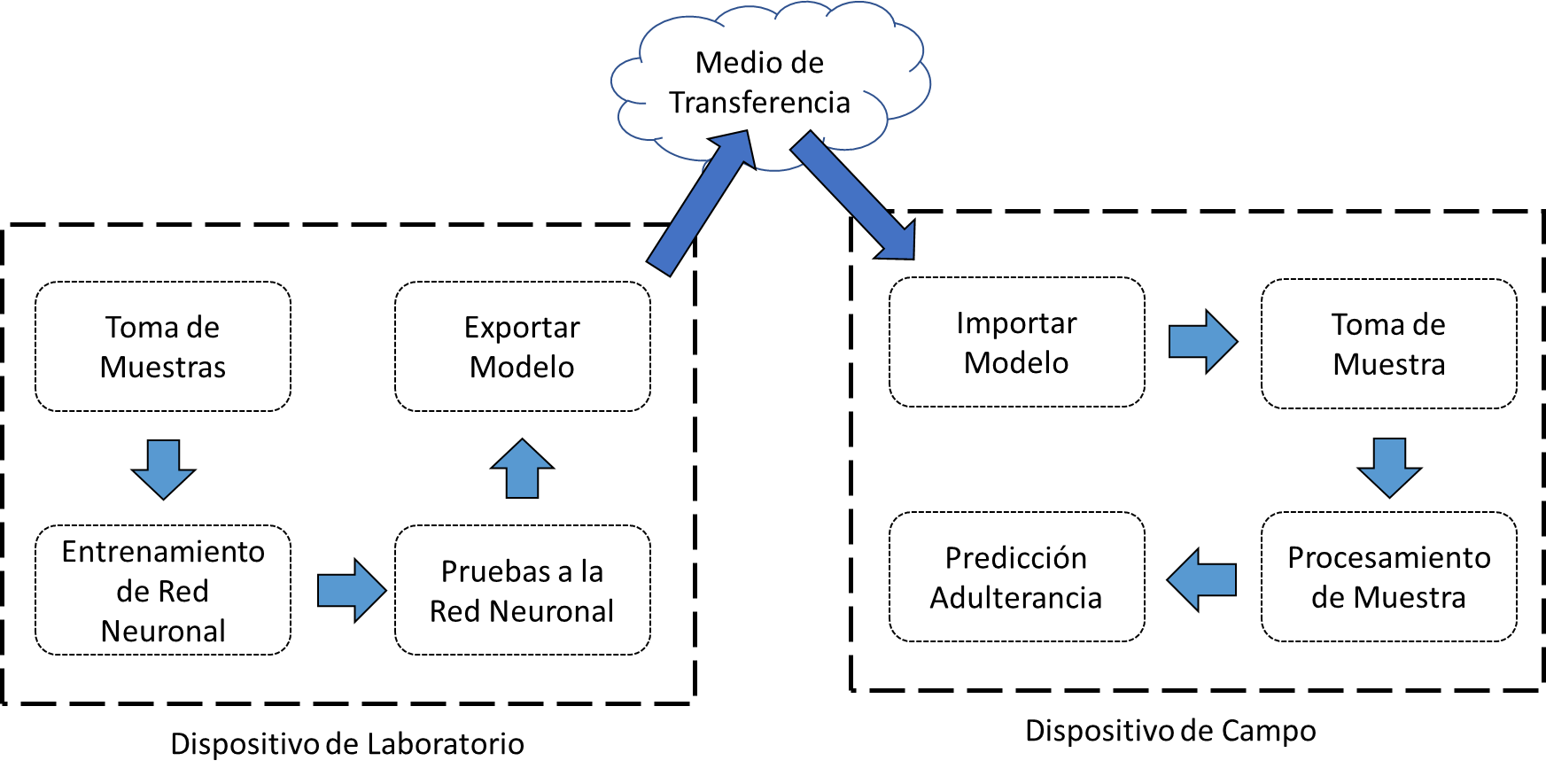
## 3.1 Diseño Sistema General

El sensor AS7265x permite detectar cambios en la composición de la leche. Para ello el sistema a desarrollar tiene que ser capaz de comunicarse con este sensor y tomar las posibles muestras para ser analizadas. Además, el sistema tiene que permitir encontrar patrones específicos para el reconocimiento de anomalías, para ello se utilizaran las grandes capacidades que brindan las RNA para clasificación de variables.

Para utilizar redes neuronales artificiales (RNA) de manera efectiva, es recomendable contar con un dispositivo con un alto nivel computacional. Esto se debe a que el entrenamiento de modelos de RNA puede ser computacionalmente intensivo y requerir recursos significativos, como capacidad de procesamiento y memoria.

Una vez que el modelo ha sido entrenado, es posible transferirlo a otras placas o dispositivos para su uso óptimo. Esto se debe a que la arquitectura y parámetros de la RNA pueden ser exportados y utilizados en otros dispositivos con recursos computacionales más limitados; por lo tanto, el diseño de dividirá en dos etapas o dispositivos:

* Dispositivo de laboratorio
* Dispositivo de campo



Esquema 1: Arquitectura General del Sistema.

## 3.2 Dispositivo de Laboratorio

El DL (dispositivo de laboratorio) en general, tendrá la tarea de tomar las muestras necesarias del sensor AS7265X, y entrenar un modelo óptimo que permita reconocer adulterancias en la leche. Este proceso puede requerir de mucho computo, por lo tanto, se necesita un dispositivo de altas capacidades de trabajo.

Este dispositivo de laboratorio tiene que ser capaz de:

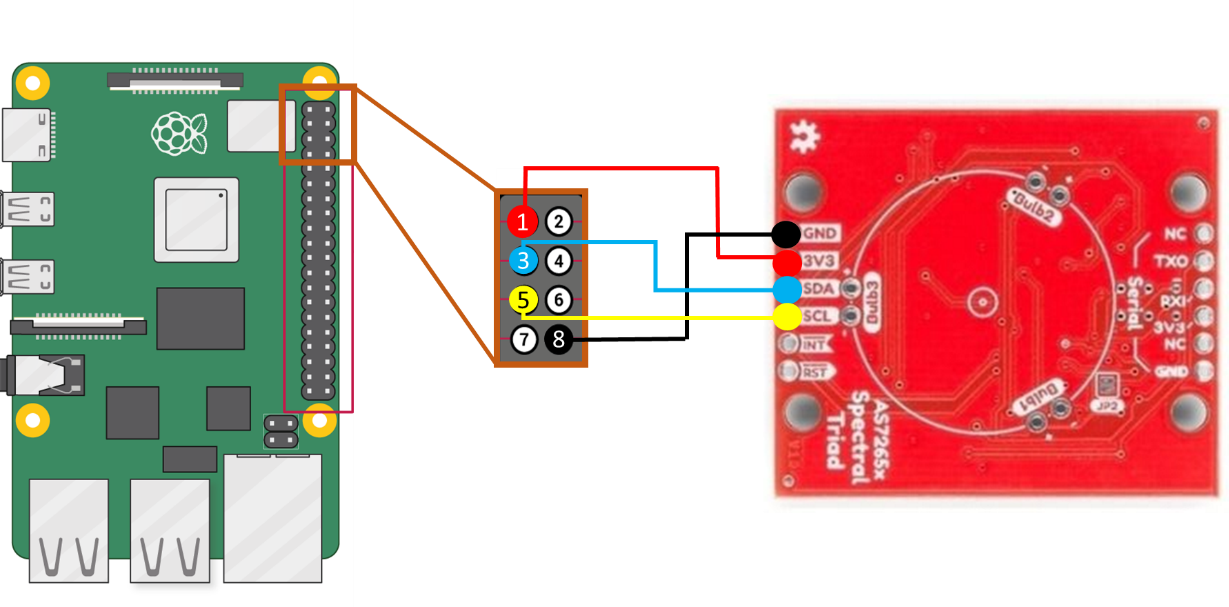
* Comunicación con el sensor y lectura de muestras (I2C o UART)
* Almacenar muestras de forma consistente (Base de datos)
* Entrenamiento del modelo
* Pruebas de predicción con modelo entrenado
* Interfaz visual para la interacción humano-maquina
* Alojar modelo hacia servidores para ser usado por los distintivos de campo

La placa de desarrollo Raspberry tiene el potencial necesario para ser seleccionado como el DL, ya que permite la conectividad de los protocolos establecidos por el sensor, permite lenguajes de bases de datos, además puede implementar Python junto a todas las librerías de redes neuronales (Tensorflow), gestión web (Django) y trabajo con servidores. Se utilizará más específicamente su versión 3 modelo B, debido a que presenta modulo de wifi integrado para una mejor comunicación con el exterior y una mayor capacidad computacional que las versiones anteriores.

### 3.2.1 Comunicación con el Sensor

Para le desarrollo del sistema de hardware, es necesario un bus de comunicación con el sensor, este soporta conexiones I2C, por lo tanto, este protocolo será utilizado en ambos dispositivos (DL y DC) para la recolección de información del espectro.

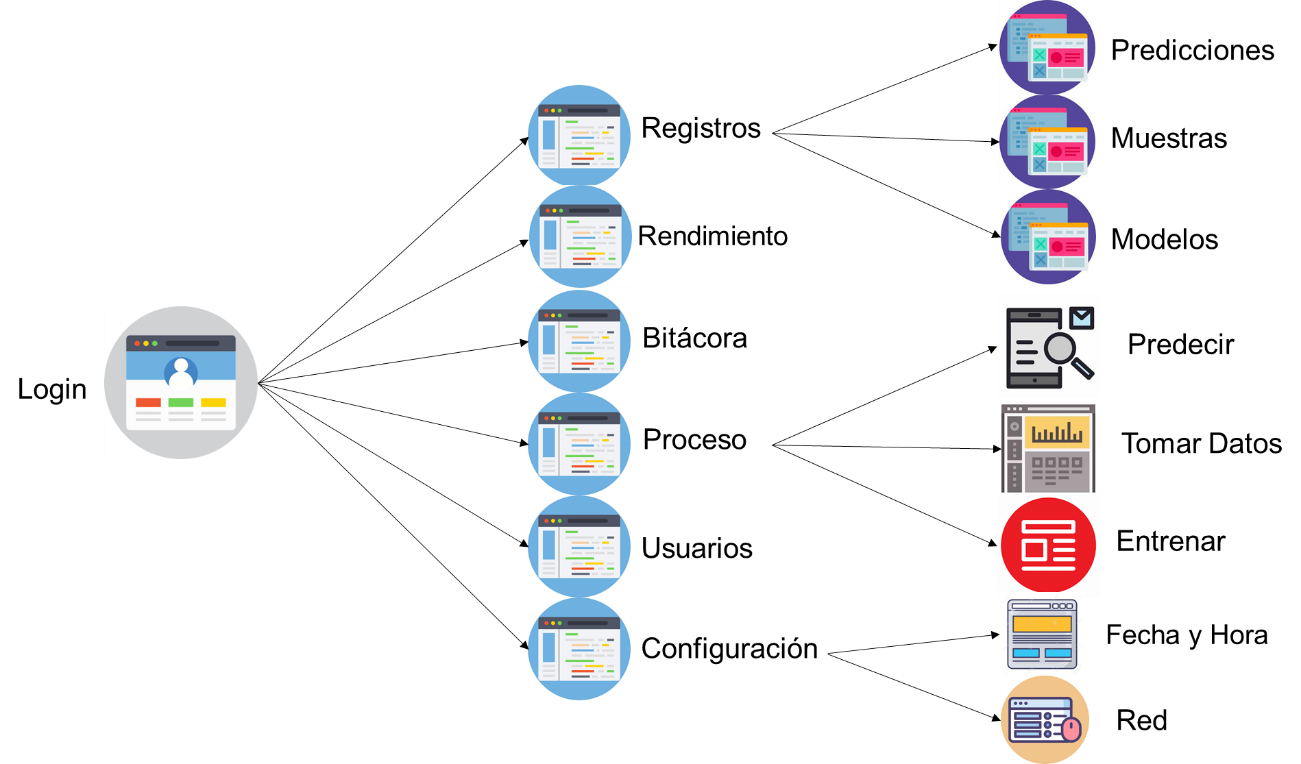
La Raspberry Pi 3B presenta hasta dos interfaces I2C (Ver Ilustración ???), se utilizarán los pines GPIO02(SDA1) y GPIO03(SCL1) para la conexión del bus. Se utilizara Python junto con la librería de smbus para la conexión entre los pines físicos y el lenguaje de alto nivel.



Esquema 2: Conexión DL-Sensor

### 3.2.2 Arquitectura de la Interfaz Visual

Para la interfaz visual se utilizará el framework Django, junto a la plantilla AdminLTE3, las cuales permitirán crear una interfaz simple, agradable y segura para el intercambio de información entre el usuario y el dispositivo.



Esquema : Interfaces Web del dispositivo de laboratorio

La programación del framework Django será orientada a “clase basada en vistas” donde se aprovechan las capacidades de este método, para brindar de una amplia seguridad al acceder a las páginas. La autenticación se hará al acceder al sitio siendo esta obligatoria en cualquier URL que se acceda. Para implementarlo se han desarrollado tres niveles de seguridad:

* Desarrollador
* Administrador
* Invitado

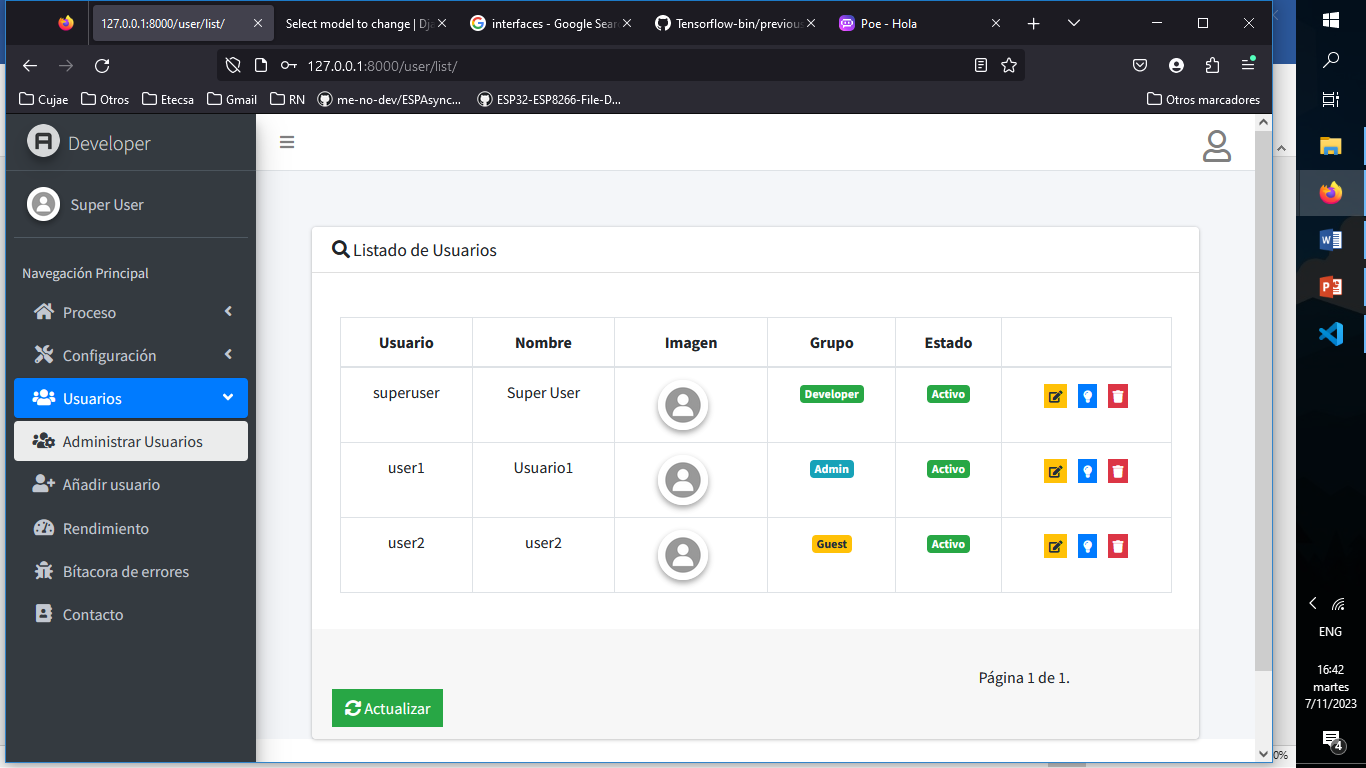


Ilustración 1: Vita de usuarios

Donde el desarrollador tendrá acceso total a todas las vistas de la página, el administrador no podrá a acceder a vistas como configuraciones internas mientras que el invitado solo podrá visualizar la información de las muestras y modelos, así como los resultados de entrenamiento.

La interfaz contara con una barra lateral de navegación para desplazarse por toda la aplicación, mostrando en todo momento la ubicación del usuario

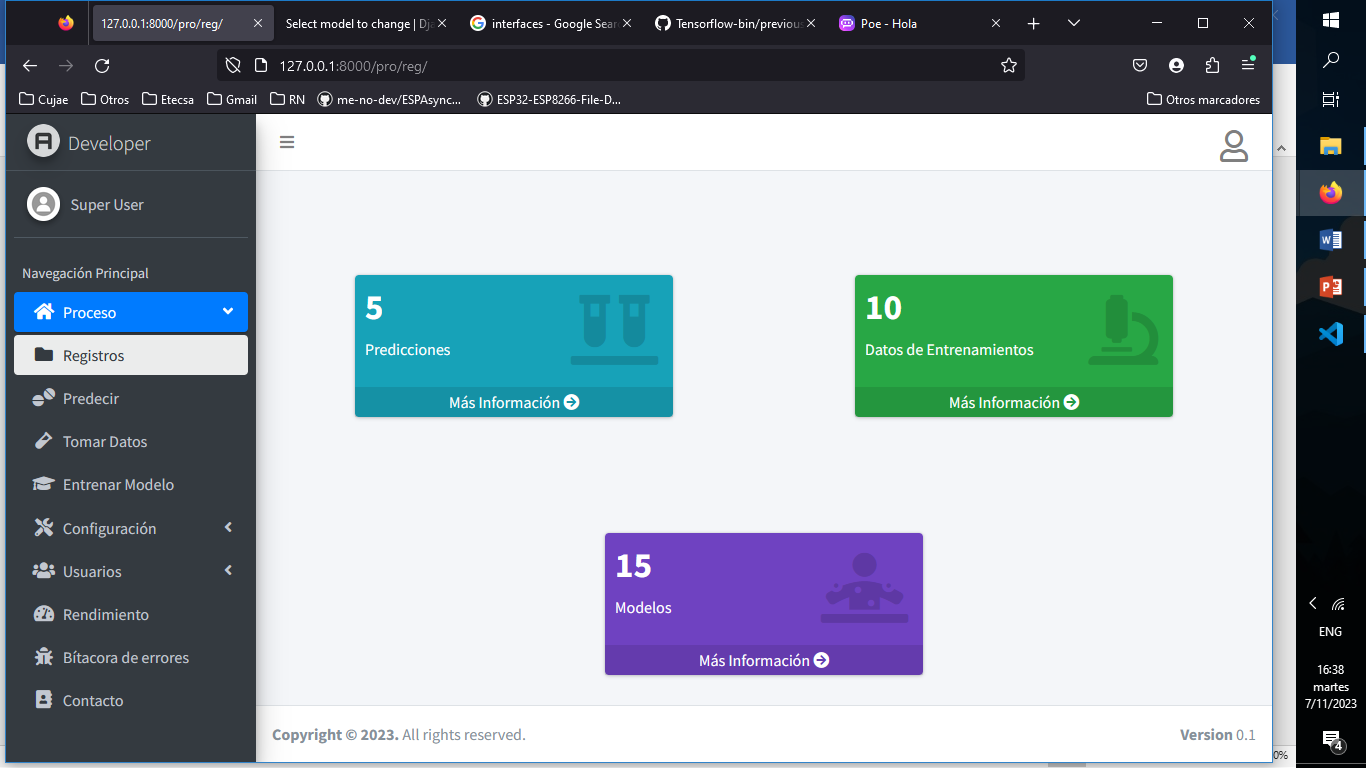


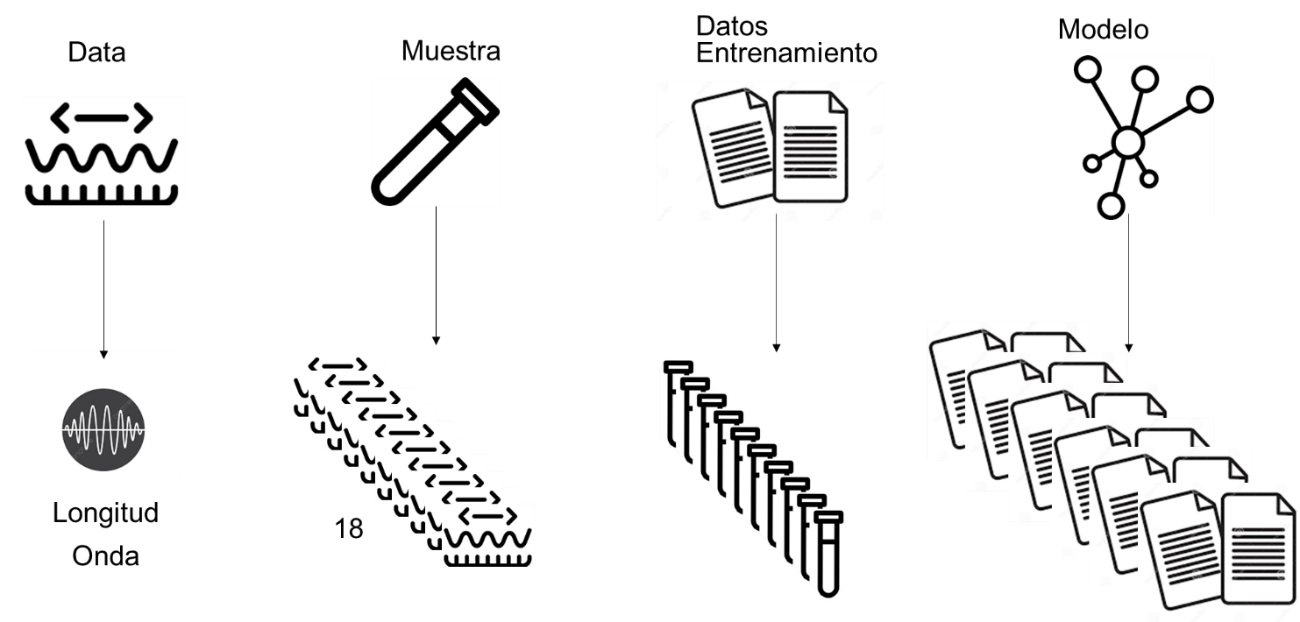
Ilustración 2: Interfaz visual con barral lateral

Para lograr un mayor rendimiento de la aplicación, se utilizará programación multihilo con la librería de Threading de Python, para así evitar bloqueos en la aplicación. Tanto en la interfaz de predicción, toma de datos, carga de ficheros y entrenamiento se iniciarán en hilos distintos, permitiendo solo uno de cada tipo a la vez, se podrá visualizar el estado de cada proceso en su respectiva interfaz de registro.

### 3.2.3 Arquitectura Base de Datos

Es necesario almacenar de manera consistente y organizada la información proveniente del sensor. Esto permite una búsqueda eficiente de los datos necesarios. Django viene implementado por defecto con la base de datos SQLite3, base de datos que permite alancear de manera organizada las muestras. Para el diseño de la base de datos relacional se establecieron los siguientes modelos:

* **Data**: Modelo base de la base de datos, contiene la información de una longitud de onda. Presenta campo FOREIGN KEY a Muestra
* **Muestra**: Modelo que contiene hasta 18 datas, campo FOREIGN KEY a Datos Entrenamiento.
* **Predicción**: Modelo Puntero a Muestra
* **Datos Entrenamiento:** Campo MANY TO MANY a Modelo
* **Modelo:** Contiene la información del modelo entrena



Esquema : Arquitectura de Base de Datos Relacional

### 3.2.4 Transferencia de datos

No todas las muestras tienen que ser tomadas por el mismo dispositivo , se pueden tener guardados registros con información del estado de la leche en bases de datos separadas, para ello es necesario algún prototipo de fichero para exportar e importar los datos. El formato CSV (Comma Separated Values) será el utilizado para cumplir este objetivo.

Se implementaron dos arquitecturas para formato de los ficheros CSV ( ver tabla ???) , donde el formato 2 es un formato que se utilizo en bases de datos que contenían información , mientras que el formato uno es el estándar que se usa entre la misma aplicación.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Formato | Data | Etiqueta | Adicional |
| Formato 1 | A, B, C, D, E, F, G, H, I, J, K, L, R, S, T, U, V, W, label | A~W | Label |  |
| Formato 2 | date,labels,410,435,460,485,510,535,560,585,610,645,680,705,730,760,810,860,900,940 | 410~940 | Labels | Date(Fecha) |

Tabla 1: Formato CSV

### ~~3.2.5 Despliegue de modelos en la nube~~

~~Para poder utilizar el modelo por en instrumento de campo será necesario subir a un servidor la información del modelo de keras, para ello se utilizará la propia plataforma de git para actualizar los modelos en la web, los modelos serán generados por la propia librería de tensorflow, pero en su versión lite para dispositivos de menores prestaciones, para que puedan ser utilizadas estas redes entrenadas; por otros lenguajes y plataformas.~~

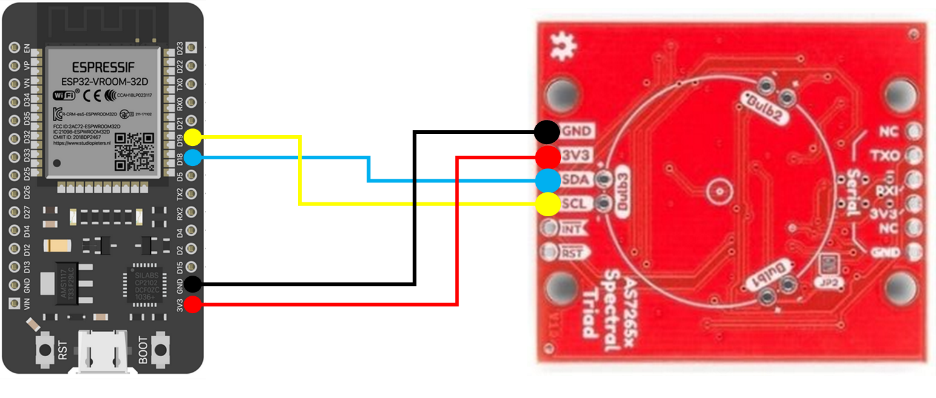
## 3.3 Dispositivo de Campo

Los DC (Dispositivos de Campo) tendrán el objetivo que cargar los modelos creados por los de laboratorio, y a partir de muestras nuevas muestras del sensor a través de la red neuronal conocer el estado de adulterancia de la leche.

Este dispositivo solo tendrá que utilizar los modelos ya entrenados, por lo tanto, no requerirá de alto procesamiento como el de laboratorio. Necesitará de una interfaz web para la comunicación con los usuarios, así como la posibilidad de comunicarse con los DL para la actualización de los modelos de redes neuronales; por lo tanto, un ESP32 es adecuado para este dispositivo, además de sus bajos costos, posee los módulos de Wifi los cuales serán necesarios para la comunicación con el dispositivo de laboratorio y el intercambio de modelos e información. Este se puede escribir en lenguaje C, lo que lo hace compatible con la librería de Tensorflow en su versión lite.

### 3.3.1 Interacción con el Sensor

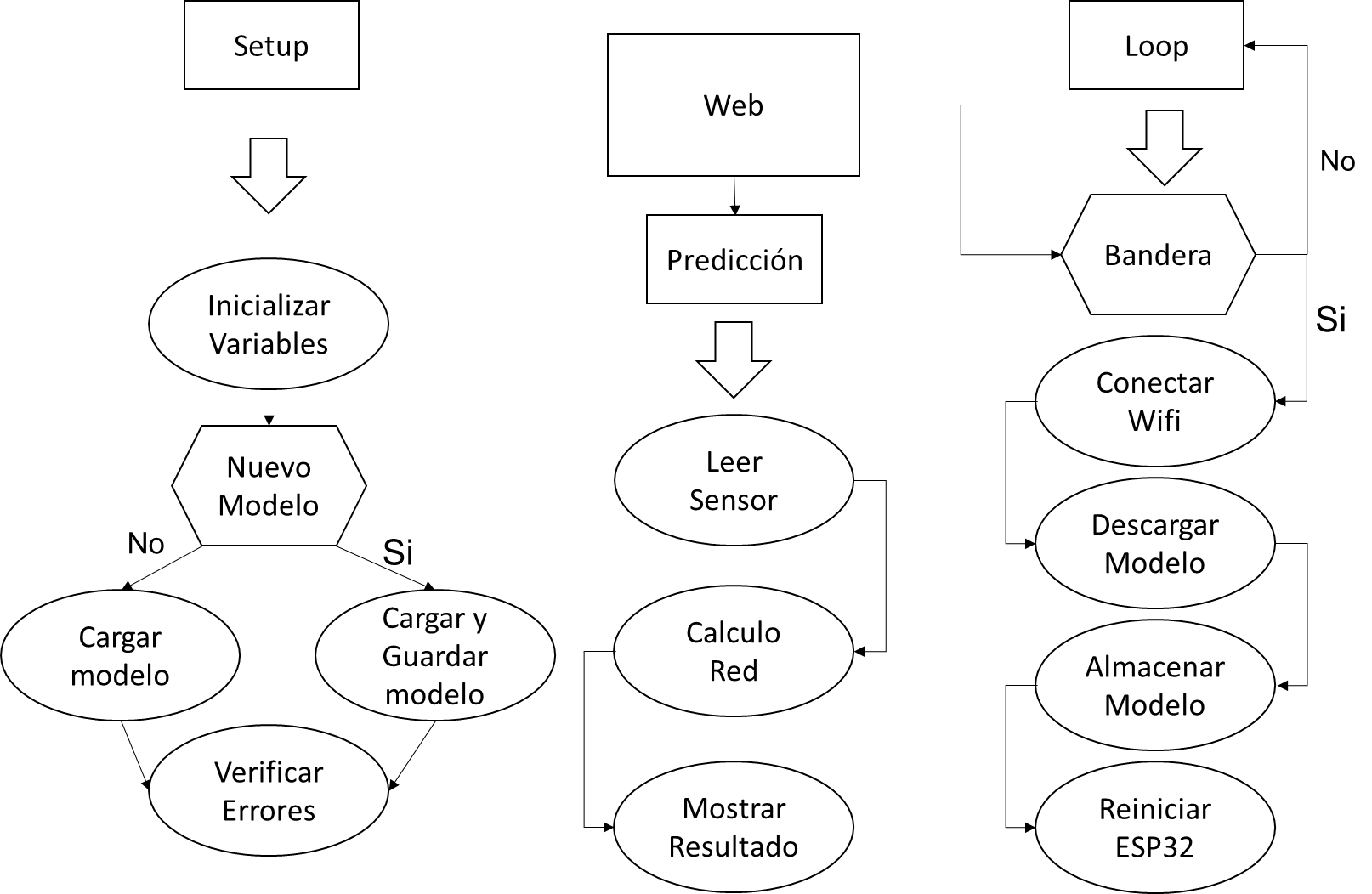
Para la toma de muestras en el ESP32 se necesita de un bus I2C para la comunicación con el AS7265x. El DC no trae por defecto pines asignados para I2C, pero puede asignar hasta 2 pares de pines para la utilización del protocolo. Se utilizaron los pines D18(SDA) y D1(SCL) para la comunicación con el sensor. Para la comunicación entre el lenguaje C con los pines físicos del dispositivo, se utilizara la librería Wire.h la cual implementa la comunicación I2C de manera optima



Esquema 5: Conexión DC-Sensor

### 3.3.2 Arquitectura de Clases

Como ya se mencionó, es ESP32 puede ser programado en lenguaje C, este será programado con el IDE VSC con PlatafformIO. En la función “Loop” del programa se implantará solamente una bandera de descarga que se activa desde la interfaz visual, debido a que este proceso de actualizar modelo es sin tiempo definido y hace necesario realizarlo en el programa principal. Además es claro mencionar que se utilizara la memoria flash del ESP32 de 4Mb para el almacenamiento de las páginas web junto los ficheros estáticos de estilos y JavaScript , así como los modelos de redes neuronales



Esquema : Flujo de Código del ESP32

Diseño de clases

* **As7265x**: Clase encargada de comunicarse con sensor, configurar y leer datos
* **Spiffs**: Clase encargada de manejar la memoria flash del esp32, verifica, carga y guarda información en la memoria
* **Wifi:** Clase encargada de manejar el controlador wifi del esp32, habilitar modos AP, con ssid y password específicos, así como conectarse a wifi para acceder al servidor. Contiene un puntero a la clase Spiffs para acceder a los datos almacenados de ssids.
* **Model**: Maneja el modelo descargado del servidor y almacenado en memoria. Actúa sobre la clase EloquentTinyML, librería que trabaja con elementos estáticos por lo tanto para modificar el modelo será necesario un reinicio de los parámetros de esta librería
* **Download**: Librería encargada de descargar el modelo del servidor. Utiliza HTTPClient para realizar las peticiones. Implementa la clase Spiffs para almacenar modelo provisional
* **Object**: Clase base para casi todas las operaciones que carga información en arreglos dinámicos, con fácil acceso, incremento y borrado.
* **\_18float:** Clase encargada de almacenar y manejar los 18 valores float provenientes del sensor
* **Web:** Clase que contiene el mayor peso ya que presenta un puntero a todas las librerías anteriores para poder utilizar sus funcionalidades de una manera ordenada. Contiene toda la implementación del servidor web, montado cobre la librería AsynWebServer

### 3.3.3 Pagina Web

Para la utilización de un servidor web se utilizado la librería AsynWebServer junto a WIFI del esp32, esta librería permite servir ficheros estáticos de una forma sencilla. Se ha utilizado Bootstrap y JQuery como librerías para la implementación de la página.

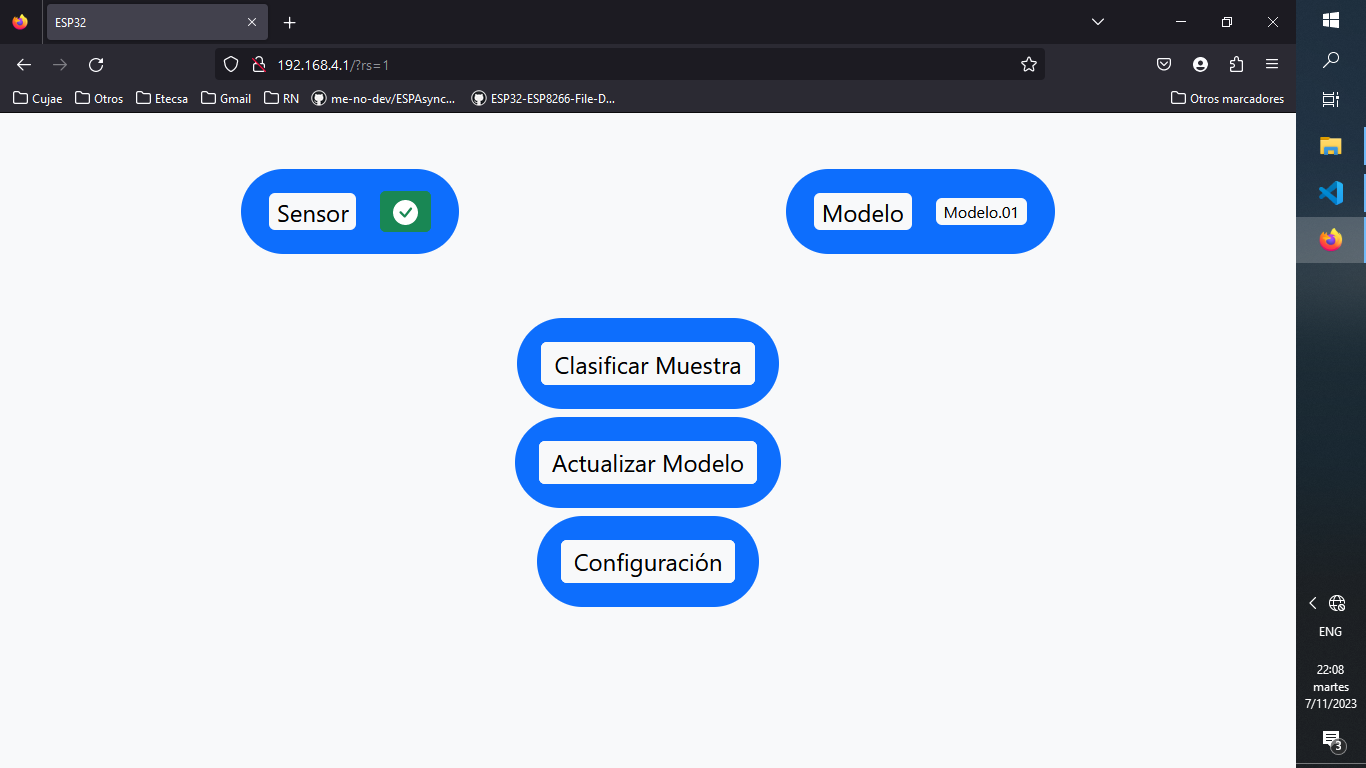


Ilustración 3: Página Web Desplegada por ESP32

Inicialmente el ESP32 arrancara con la Wifi en modo AP, con un usuario y contraseña establecido por el usuario o por defecto. Al conectarse y acceder a la página se podrá configurar la URL del servidor, el modelo a actualizar, el SSID y PASSWORD a conectarse para acceder al servidor.

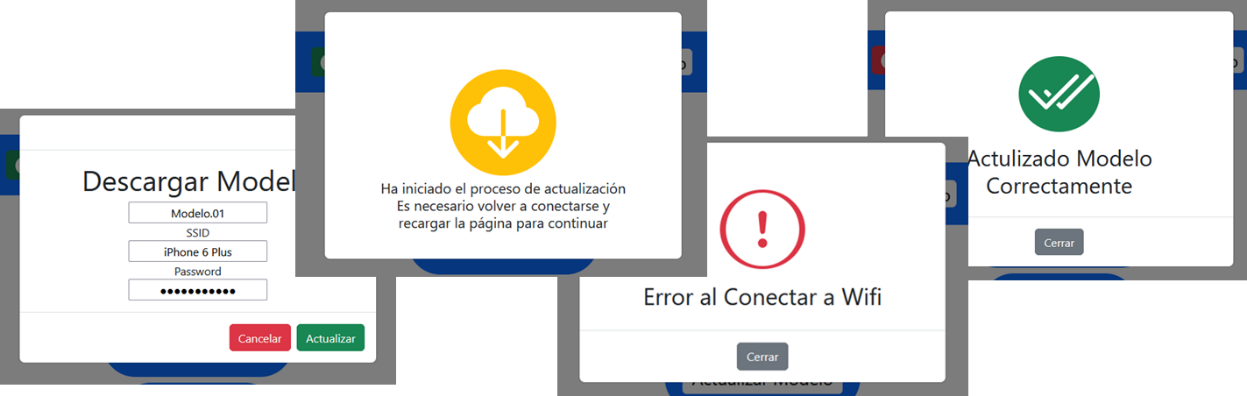


Ilustración 4: Interfaces de actualización de modelo en ESP32

### ~~3.3.4 Almacenamiento permanente~~

### ~~3.3.5 Actualización de modelos~~

## 3.4 Utilización de Redes Neuronales Artificiales

### 3.4.1 Arquitectura de Red Neuronal

### 3.4.2 Pruebas en Laboratorio

### 3.4.3 Pruebas en Campo

# Referencias

# Anexos

