ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO DE UN SISTEMA AUTÓNOMO PARA DETERMINAR EL PRECIO DE FRUTAS BASADO EN VISIÓN ARTIFICIAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

BRYAN PAUL PALATE SISALEMA

DIRECTOR: Ph.D. ROBIN GERARDO ÁLVAREZ RUEDA

Quito, junio 2021

AVAL

Certifico que el presente trabajo fue desarrollado por Palate Sisalema Bryan Paul, bajo mi supervisión.

Ph.D. ROBIN GERARDO ÁLVAREZ RUEDA
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Palate Sisalema Bryan Paul, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.
A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamento y Normas vigentes.

BRYAN PAUL PALATE SISALEMA

DEDICATORIA

Con infinita gratitud dedico este trabajo a mis abuelitos; quienes, en su gran sabiduría nos dieron el privilegio de estudiar.

AGRADECIMIENTO

A mis padres, por sacrificarse todos los días para brindarme la oportunidad de dedicarme únicamente a mis estudios, sin tener la necesidad de buscar un trabajo para subsistir durante mi época universitaria.

A mis dos hermanas porque, sin saberlo, me recargaron de energía en los momentos que tenía ganas de tirar la toalla.

A mis amigos por el tiempo, amanecidas, alimentos, posada y momentos compartidos.

Un profundo agradecimiento al Dr. Robin Álvarez por su dedicación y tiempo invertido para guiar este Trabajo de Titulación de la mejor manera.

A los muchachos de “La Santiago BOX” por inyectarle vida a los días de pandemia.

A las artes marciales y al deporte en general por enseñarme la disciplina y trabajo en equipo.

Paul Palate

ÍNDICE DE CONTENIDO

RESUMEN

En el presente trabajo se diseñó y construyó un prototipo de un sistema autónomo para determinar el precio de frutas basado en Visión Artificial. Se lo dividió en cuatro módulos para asegurar su correcto funcionamiento.

El primer módulo se encarga del pesaje de las frutas y posterior envío de los valores hacia el ordenador. Para esto se construyó una balanza digital, utilizando una celda de carga y un transmisor de peso. Se implementó un dispositivo Arduino para la transmisión de datos. El segundo módulo se encarga de la adquisición de imágenes en las mejores condiciones al controlar el entorno completo de la captura de imágenes. Para esto se agregó un soporte a la balanza digital en donde se ubica una lámpara circular y una cámara web en el centro de ésta. En el tercer módulo se realiza el reconocimiento de frutas basado en su color y características en su morfología tales como superficie, eje mayor y eje menor. En el cuarto módulo se desarrolla una interfaz gráfica que unifica las etapas anteriores. Se determina el precio a pagar por las frutas y le brinda al usuario el detalle de compra de manera autónoma, es decir, sin la necesidad de presencia humana para este fin. El prototipo es capaz de reconocer seis diferentes tipos de frutas entre los que constan manzanas verdes y rojas, peras verdes y amarillas y bananas amarillas y rojas. Después de una serie de pruebas se determinó que el sistema tiene una exactitud del 98% en cuanto al reconocimiento de frutas.

Palabras clave: Visión artificial, reconocimiento de frutas, sistema de pago autónomo.

ABSTRACT

In the present work, a prototype of an autonomous system to determine the price of fruits based on Artificial Vision was designed and built. It was divided into four modules to ensure its proper functioning.

The first module is responsible for the weighing of fruit, and subsequent sending of the values to the computer. For this, a digital scale was built using of a load cell and a weight transmitter. An Arduino device was implemented for data transmission. The second module takes care of the acquisition of images in the best conditions by controlling the environment of the image capture. For this, a support was added to the digital scale where a circular lamp and a web cam in the center of it are located. In the third module, fruit recognition is carried out based on their color and characteristics in their morphology such as area, major axis and minor axis. In the fourth module, a graphical interface that unifies the previous stages was developed. The price to pay for the fruit is determined, and the user is provided with the purchase details autonomously, that is, without the need for human presence for this purpose. The prototype is able to recognize six different types of fruit, including green and red apples, green and yellow pears and yellow and red bananas. After a series of tests, it was determined that the system has an accuracy of 98% regarding the recognition of fruits.

Keywords: Artificial vision, fruit recognition, autonomous payment system.

1. INTRODUCCIÓN

Actualmente, los modelos de negocios clásicos están siendo forzados a usar la tecnología digital para automatizar sus sistemas productivos y comerciales, siendo uno de ellos el mercado de venta de frutas, en donde los locales comerciales deben adaptarse a las nuevas exigencias del consumidor a fin de ofrecer el mejor servicio y paralelamente asegurar la rentabilidad del negocio [1]. El tema que se pretende abordar es el cobro automático de las frutas escogidas por el cliente, el cual, y al momento requiere necesariamente de un empleado que identifique visualmente la fruta en cuestión, la pese en una balanza y determine el valor a pagar de acuerdo a una tabla de precios. Por lo anterior, la calidad del servicio ofrecido al cliente se ve afectado por el tiempo que debe esperar hasta saber el valor a pagar por su producto.

Para tratar de resolver estos inconvenientes, el uso de etiquetas para la identificación inmediata de la fruta, así como su precio parece ser una solución útil; sin embargo, también presenta problemas de eficiencia ya que se los pega de manera individual y con el paso del tiempo se pueden desprender de la fruta o volverse ilegibles por deterioro, causando que el empleado deba acudir nuevamente a una tabla de códigos para poder cobrar por el producto [2].

La utilización de Visión Artificial representa una mejora significativa frente a algoritmos anteriores que intentan resolver la misma problemática. Por ejemplo, en el trabajo “Desarrollo de un Sistema Prototipo de Pago de Frutas a través del Entrenamiento de Redes Neuronales Artificiales Convolucionales” [3] , sus autores tratan de resolver el problema utilizando herramientas de Inteligencia Artificial (IA); aunque la atención que se da al funcionamiento de dichas herramientas hace que se descuide la comprensión del problema y no se realicen pruebas reales con distintas frutas y en tiempo real.

Creemos importante resumir dicho trabajo para que se vea la complejidad del sistema implementado. A continuación, se hace un resumen de las herramientas que se emplearon en [3]:

Módulo de Reconocimiento de Frutas: Este módulo está dirigido para la identificación de las frutas con la utilización de Redes Neuronales Convolucionales (CNN) mediante fotografías tomadas con una cámara externa conectada a la computadora mencionada en el prototipo electrónico.

Las librerías que permitirán el entrenamiento de la Red Neuronal Convolucional (CNN) son las siguientes:

- TensorFlow: Plataforma de código abierto para el aprendizaje automático.

• Keras: API de TensorFlow para construir y entrenar modelos de aprendizaje profundo.

• OpenCV: Es una biblioteca altamente optimizada con enfoque en aplicaciones en tiempo real. En este módulo se utilizarán para el entrenamiento y predicción de frutas las librerías de TensorFlow y Keras; mientras que OpenCV será usado para la captura de fotografías.

Estas y otras herramientas empleadas quedan resumidas en la Figura 1.1.

FASE DE IMPLEMENTACIÓN:

- Instalar paquete de lenguaje de programación Python.
- Instalar Mysql para creación de base de datos.
- Instalar entorno de desarrollo Anaconda Navigator.
- Crear entorno virtual en Anaconda para instalación de herramientas de IA.
- Instalar librerías: TensorFlow, Keras, OpenCV y Matplolib.
- Instalar el IDE Spyder.
- Elaborar balanza digital con dispositivos mencionados anteriormente.
- Instalar el IDE de Arduino.
- Codificar el programa para balanza digital.
- Calibrar la balanza elaborada.
- Configurar la comunicación de la balanza con la computadora.
- Crear base de datos de imágenes para el entrenamiento, validación y test.
- Programar la red neuronal de acuerdo a las características planteadas.
- Entrenar la red neuronal convolucional.
- Configuración de la cámara externa para comunicación con un computador.
- Crear función para predicción de la fruta con la red neuronal previamente entrenada.
- Realizar los formularios para módulo de detalle de pago.
- Crear la base de datos con sus respectivas tablas.

Figura 1.1. Herramientas empleadas en [3].

Para una mejor precisión en el entrenamiento de la Red Neuronal se creó una propia base de datos de imágenes de frutas; **la cual será en el orden de los miles por cada clase de fruta**. Esta base de datos contendrá imágenes de distintas frutas reales en diferentes posiciones y ángulos, para que el entrenamiento de la Red Neuronal Artificial Convolucional sea lo más exacta posible y a su vez, el reconocimiento obtenga un porcentaje de precisión considerable.

Técnica de **Transfer Learning** permite utilizar una red ya entrenada y adecuarla en función a las características que se requiera. De esta forma no es necesario entrenar una red desde cero, dado a que esto implica un gran procesamiento de datos y requiere de mucho tiempo de ejecución.

Esta técnica permite descargar un modelo entrenado previamente con una extensa base de datos y utilizar sus parámetros como inicio para un nuevo modelo con datos de entrenamiento mejores al original.

En la Tabla 1.1. se resume los parámetros de la red neuronal empleada.

Tabla 1.1. Parámetros de la red neuronal empleada en [3].

		Descripción
Número de Clases	6	Corresponderá a las seis frutas propuestas
Tipo de red neuronal	Red Neuronal Convolucional (CNN)	Red potente que permitirá trabajar con imágenes de frutas.
Tipo de aprendizaje	Supervisado	La red aprenderá con ejemplos de fotos de frutas (directorio de entrenamiento)
Tipo de problema a resolver	Clasificación de varias frutas (Categorical)	Se trabajará con más de dos frutas.
Técnicas para aumentar precisión	*Data Augmentation*	Aumentará imágenes con cambios tales como zoom y giros.
	Transfer Learning	Se utilizarán modelos bien estructurados para ser adaptados a nuestro problema de clasificación de frutas.
Funciones de activación	RELU	Se utilizará para pasar los valores necesarios entre capas de la red.
	Softmax	Establecerá a las clases (frutas) como salidas en la última capa de la red.
Optimizadores	Adam	Permitirán reducir el error al momento de entrenar la red.
	RMSprop	
Número de	Épocas	Se establecerán de acuerdo al resultado que nos devuelva la métrica durante el proceso de entrenamiento. Esto con el objetivo de evitar el Sobreajuste de la red.
	Pasos en cada época y pasos de validación	

Como se puede ver, a pesar de la complejidad del trabajo mencionado, los resultados finales que se logran son obtenidos en base a las imágenes obtenidas del video realizado de un ejemplar de cada fruta, es decir, el sistema jamás fue evaluado en base a diferentes frutas de cada tipo, lo cual es también un detalle que se debe mejorar.

Por lo anterior, el presente trabajo pretende desarrollar el mismo sistema automático, pero de manera mucho más simple y únicamente basada en la clasificación de la fruta por medio de su color y de su morfología ya que cada tipo de fruta tiene distinta longitud y ancho. Finalmente, este trabajo sí obtendrá sus resultados de clasificación a partir de distintas frutas de cada tipo.

1.1. OBJETIVOS

El objetivo general de este Proyecto Técnico es: desarrollar un Prototipo de un Sistema Autónomo para determinar el Precio de Frutas Basado en Visión Artificial.

En tanto que los objetivos específicos son los siguientes:

- Estudiar los algoritmos de visión artificial necesarios para la clasificación de la fruta basado en su color y morfología (dimensiones).
- Diseñar e implementar sistema autónomo de pesaje y cálculo de costo final.
- Implementar una interfaz de usuario que integre todos los módulos de pesaje, identificación del tipo de fruta mediante visión artificial y cálculo de su costo final.
- Realizar pruebas para analizar el funcionamiento del prototipo de sistema autónomo y los resultados.

1.2. ALCANCE

El sistema a desarrollar consistirá de una interfaz gráfica amigable con el consumidor y solicitará que el consumidor coloque la fruta en la balanza, después el sistema devolverá el precio del producto de manera totalmente automática.

El reconocimiento del tipo de la fruta realizado mediante visión artificial tiene dos subprocesos:

- Reconocimiento del color de la fruta (rojo, verde y amarillo).
- Determinación del centroide de la fruta y medición del largo y ancho de dicha fruta.

El proyecto se limitará al reconocimiento de tres especies de frutas, cada una con dos variedades de esta:

1) Manzana

    - Roja

    - Verde

2) Pera

    - Verde

    - Amarilla

3) Banana

    - Amarilla

    - Roja

El módulo de pesaje de las frutas será realizado mediante una balanza construida para este fin en base a un Arduino y un sensor especializado.

Se diseña una interfaz gráfica amigable con el usuario para presentar en detalle el pago por las frutas seleccionadas para la compra.

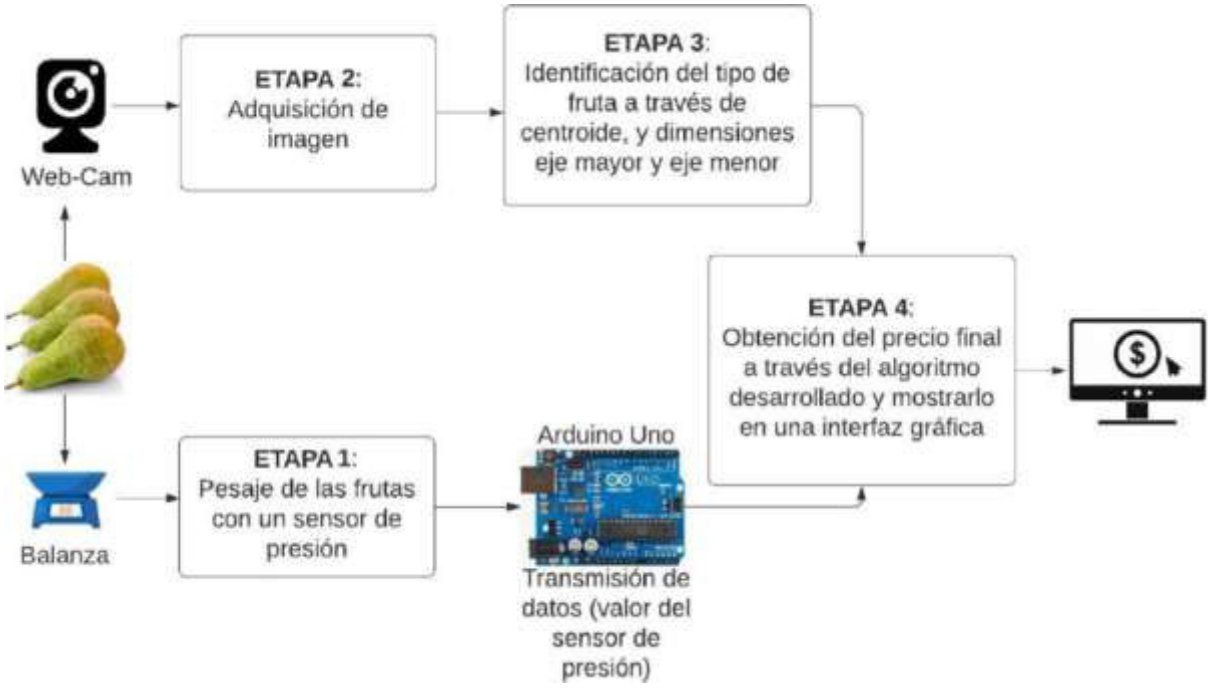


Figura 1.2. Diagrama de bloques para el desarrollo del sistema autónomo de cobro de frutas.

La Figura 1.2. resume de manera gráfica la forma en la que se implementa el sistema autónomo para determinar el costo de las frutas. En la ETAPA 1 se realiza el proceso de pesaje de la fruta y se envía el valor a través del dispositivo Arduino que sirve para la transmisión de datos. En la ETAPA 2 se adquiere la imagen a través de una webcam ubicada convenientemente en un soporte. En la ETAPA 3 se realiza el proceso de identificación de frutas al encontrar el centroide y midiendo las dimensiones del eje mayor y eje menor para determinar el tipo de fruta de acuerdo con la morfología de ésta. Finalmente, el cliente podrá observar el proceso de compra en la ETAPA 4 y saber el precio a pagar. Además, se hace un registro de las compras para llevar un inventario de la tienda necesario para llevar un control por parte del dueño.

Finalmente, para mostrar el detalle de pago al cliente, se desarrollará una interfaz gráfica para observar todo el proceso de compra. Para verificar el funcionamiento del sistema autónomo de cobro de frutas se realizará un prototipo de éste, y cumplirá con las siguientes características:

• Se trabajará únicamente con tres tipos de frutas, cada una con dos variedades como se explicó en párrafos anteriores.

• La base de la balanza será de color blanca mate para evitar reflejos por el uso de la lámpara circular en la parte superior del módulo.

• Web-Cam, cuya resolución no necesariamente debe ser alta para esta aplicación.

• Detalle de todo el proceso de compra y el pago final a través de una interfaz de usuario con App Designer de Matlab [4].

• El peso máximo será de cinco kilogramos ya que es el peso máximo que tolera el transmisor de peso HX711 [5].

1.3. MARCO TEÓRICO

1.3.1. PROCESAMIENTO DE IMÁGENES

1.3.1.1. Definiciones

Para entender de manera clara el procesamiento de imágenes, es necesario establecer ciertas definiciones que se utilizarán a lo largo de este proyecto.

- Visión Artificial: Consiste en adquisición, procesamiento, clasificación y reconocimiento de imágenes digitales.
- Píxel: Elemento más básico de una imagen.
- Imagen: Arreglo bidimensional de píxeles con diferente intensidad de escala de grises.

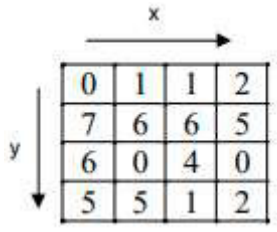


Figura 1.3. Imagen de 16 píxeles con diferentes intensidades de gris en cada pixel.

- Brillo: Indica si un área está más o menos iluminada.
- Tono: Indica si un área parece similar al rojo, amarillo, verde o azul o a una proporción de ellos.
- Luminosidad: Brillo de una zona respecto a otra zona blanca en la imagen.
- Croma: Indica la coloración de un área respecto al brillo de un blanco de referencia.

Para obtener una imagen a color primero se deben transformar los parámetros cromáticos en eléctricos y representar los colores, lo cual puede realizarse de diferentes maneras, dando lugar a diferentes espacios de colores o mapas de color [6].

1.3.1.2. Espacio RGB

Se basa en la combinación de tres señales de luminancia cromática distinta: rojo, verde, azul (RGB). La forma más sencilla de obtener un color específico es determinar la cantidad de color rojo, verde y azul que se requiere combinar para obtener el color deseado como se ve en la Figura 1.4. para lo cual se realiza la suma aritmética de las componentes: X = R + G + B, gráficamente representada por un cubo.

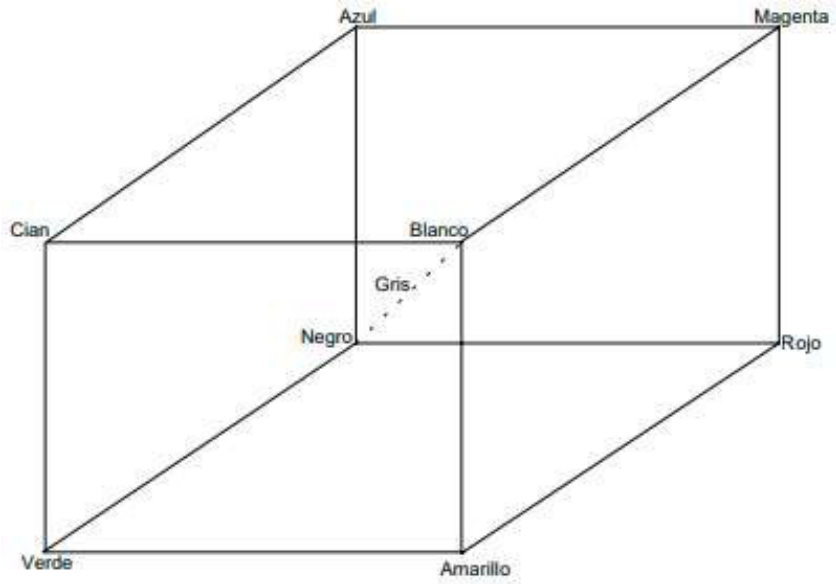


Figura 1.4. Espacio de colores RGB.

En la recta que une el origen con el valor máximo se encuentran ubicados los grises (escala de gris) debido a que sus tres componentes (RGB) son iguales. Cuando una cámara adquiere una imagen a color, para cada píxel en color se tienen en realidad tres valores, una para cada uno de los colores básicos (rojo, verde y azul); la ganancia máxima para cada componente corresponde a la longitud de onda de los tres colores básicos.

Un color puede definirse como la combinación de tres colores básicos: rojo, verde y azul, y expresarse mediante una tripleta de valores de 0 a 1 (RGB), donde R, G y B representan las intensidades de cada uno de los tres colores básicos rojo, verde y azul, respectivamente. En la Tabla 1.2. se presentan ejemplos de colores definidos mediante estas tripletas.

Tabla 1.2. Colores RGB [7].

COLOR	R	G	B
Blanco	1	1	1
Rojo	1	0	0
Amarillo	1	1	0
Verde	0	1	0
Turquesa	0	1	1
Gris	0.5	0.5	0.5
Rojo Oscuro	0.5	0	0
Azul	0	0	1
Aguamarina	0.5	1	0.83
Negro	0	0	0

Histograma de una imagen: Es una representación del número de píxeles de cierto nivel de gris en función de los niveles de gris.

Conectividad: Es un concepto importante, utilizado para establecer los límites de objetos en regiones dentro de una imagen. Para determinar si dos píxeles están conectados se determina si son adyacentes de alguna manera, y si sus niveles de gris satisfacen un criterio de similitud (por ejemplo, si son iguales). Por ejemplo, en una imagen binaria con valores de 1 y 0, dos píxeles pueden ser vecinos, pero se dice que están conectados solo cuando tienen el mismo valor.

Distancia: La distancia o transformada de distancia proporciona una medición de la separación existente entre dos puntos dentro de una imagen [8].

1.3.1.3. Clases de almacenamiento usadas en Matlab

Por defecto, Matlab almacena la mayoría de los datos en clase ***double*** (doble). Los datos en estos arreglos se almacenan como datos de punto flotante de doble precisión (64 bits).
Para reducir el espacio en memoria, Matlab almacena los datos en arreglos de 8(***uint8***) o 16 bits(***uint16***) sin signo. Estos arreglos requieren cuando mucho la octava o cuarta parte de la memoria requerida por un arreglo tipo ***double.*** En la Tabla 1.3. se presentan los tipos de imágenes.

Tabla 1.3. Tipos de imágenes y clases numéricas.

Tipo de imagen	Clase de almacenamiento	Interpretación
Binaria	logical	Arreglo de 1s y 0s
Indexada	double	Arreglo de enteros en el rango [1, p]
	uint8 o uint16	Arreglo de enteros en el rango [1, p-1]
Intensidad	double	Arreglo de valores en punto flotante, su rango típico es [0, 1]
	uint8 o uint16	Arreglo de enteros, rango típico [0, 255] o [0, 65535], respectivamente
RGB (color verdadero)	double	Arreglo de valores en punto flotante de m x n x 3 en el rango [0, 1]
	uint8 o uint16	Arreglo de enteros de m x n x 3 en el rango [0, 255] o [0, 65535], respectivamente

El toolbox de Image Processing permite manejar imágenes de clase ***uint16***, pero antes de procesar estas imágenes, deben convertirse a clase ***double*** o ***uint8***. Para convertir a double, utilizar el comando ***im2double***.

Para desplegar una imagen se utiliza el comando "***imshow***", pero se especifica tanto la matriz-imagen como el mapa de color de la forma: ***imshow(X,mapa)***, donde ***imshow*** despliega para cada píxel de X el color almacenado en la correspondiente columna del mapa de color.

1.3.2. RECONOCIMIENTO EN BASE AL COLOR DE LA FRUTA

1.3.2.1. Extracción de capas RGB

Haciendo énfasis en el tema de la clasificación de frutas. Es evidente que su color es determinante para realizar un buen reconocimiento de estas. Los tres colores que se deben diferenciar son el rojo, verde, y amarillo. Al trabajar con imágenes RGB se debe separar cada una de las capas por separado. Esto se realiza como se muestra en la Figura 1.5. en donde se tiene en primer lugar una matriz tridimensional y luego se extraen cada una de sus capas [9].

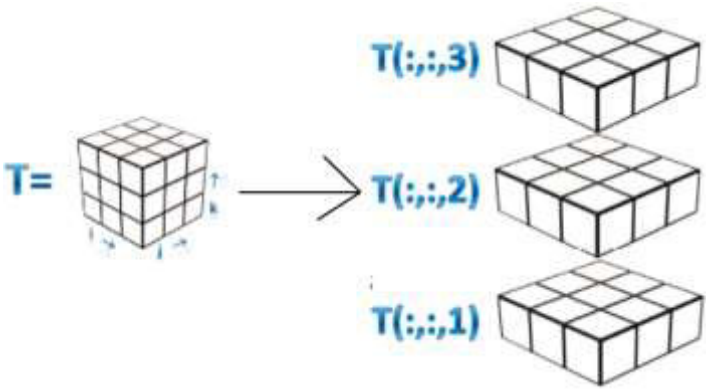


Figura 1.5. Extracción de capas de una imagen RGB.

1.3.2.2. Inversión de colores

Se utiliza para encontrar el complemento de un color. Es así que será necesario para el reconocimiento del color amarillo, el cual es el color complemento del azul. Para realizar este proceso se usa el comando ***imcomplement***.

1.3.2.3. Imágenes binarias

En una imagen binaria, cada píxel asume un valor discreto; esencialmente dichos valores corresponden a 1(uno) o 0(cero), encendido o apagado. Una imagen binaria se almacena en un arreglo de píxeles 1s o 0s. Para convertir una imagen a binaria se utiliza el comando ***imbinarize*** acompañada del valor umbral para decidir si el valor del pixel resultante será 1 si está por encima del umbral o 0 en caso de que sea un valor inferior. En la Figura 1.6. se muestra una imagen binaria.

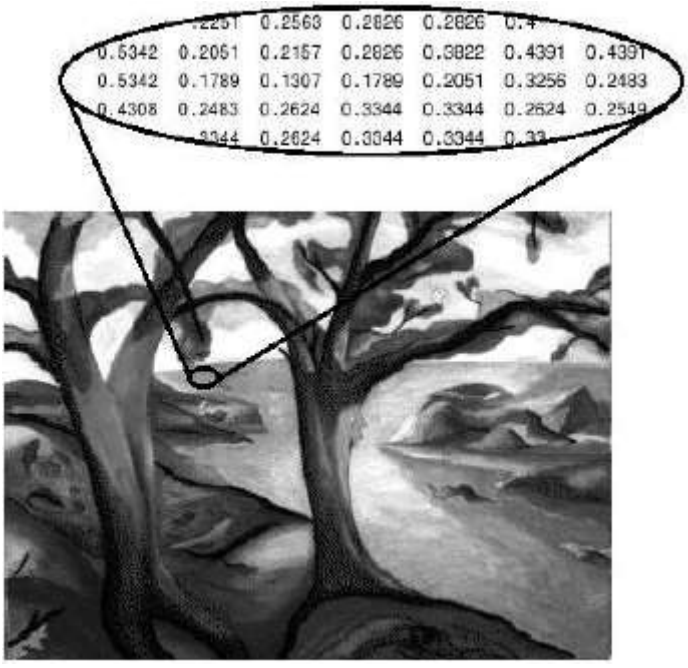


Figura 1.6. Estructura de una imagen binaria

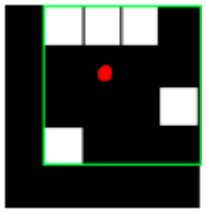
1.3.3. CLASIFICACIÓN EN BASE A LA MORFOLOGÍA DE LA FRUTA

1.3.3.1. Comando *regionprops*

Mide las propiedades de las regiones de imagen, especificados como una lista separada por comas de escalares de cadena o vectores de caracteres, una matriz de celdas de escalares de cadena o vectores de caracteres. Los nombres de propiedad no distinguen mayúsculas de minúsculas y se pueden abreviar.

En la Tabla 1.4. se enumeran todas las propiedades que proporcionan medidas de forma, las mismas que servirán para clasificar a las frutas en base a su morfología.

Tabla 1.4. Medidas de forma.

Propiedad	Descripción
'Area'	Número real de píxeles en la región, devueltos como escalares. Este valor puede diferir ligeramente del valor devuelto por el comando '***bwarea***'.
'BoundingBox'	Rectángulo más pequeño que contiene la región.
'Centroid'	Centro de masa de la región. El primer elemento es la coordenada horizontal "X" del centro de masa. El segundo elemento es la coordenada vertical "y". Esta figura ilustra el cuadro centroide y el cuadro delimitador de una región. La región consta de los píxeles blancos; el cuadro verde es el cuadro delimitador, y el punto rojo es el centroide.  **Figura 1.7.** Detección de Centroide
'MajorAxis'	Longitud (en píxeles) del eje principal de la elipse que tiene el mismo segundo momento central normalizado que la región, devuelto como escalar.

‘MinorAxis’	Longitud (en píxeles) del eje menor de la elipse que tiene el mismo segundo momento central normalizado que la región, devuelto como escalar.
‘Orientation’	El ángulo entre el eje y el eje principal de la elipse que tiene los mismos segundos momentos que la región, devuelto como escalar. El valor está en grados, que van desde -90 grados a 90 grados. La figura ilustra los ejes y la orientación de la elipse. El lado izquierdo de la figura muestra una región de imagen y su elipse correspondiente. El lado derecho muestra la misma elipse con las líneas azules sólidas que representan los ejes. Los puntos rojos son los focos. La orientación es el ángulo entre la línea de puntos horizontal y el eje principal.  **Figura 1.8.** Orientación de una figura a) Detección de objeto b) Ángulo calculado con la horizontal
‘Perimeter’	La distancia alrededor del límite de la región se devolvió como escalar. calcula el perímetro calculando la distancia entre cada par de píxeles adyacente alrededor del borde de la región.

1.3.4. PESAJE

En la Figura 1.9. Se encuentran los principales dispositivos que se utilizarán para este módulo y el software encargado de su funcionamiento. A continuación de describirán cada uno de ellos y sus características principales:

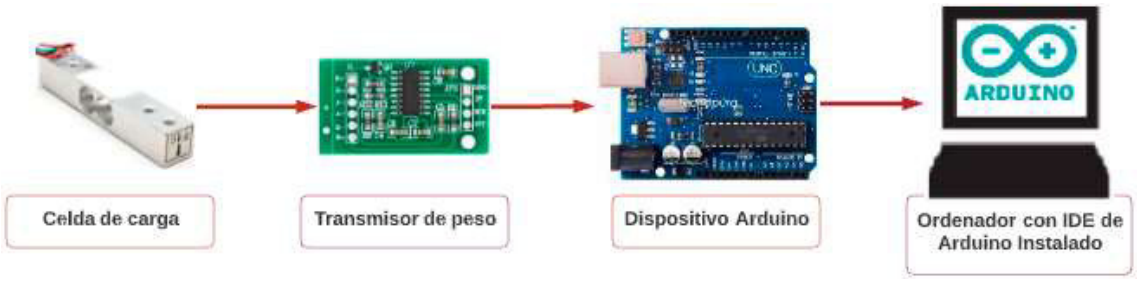


Figura 1.9. Dispositivos utilizados en el módulo de pesaje

1.3.4.1. Celda de carga

En 1856 Lord Kelvin descubrió que al aplicar una fuerza sobre un hilo conductor o un semiconductor se presenta una variación en su resistencia eléctrica. Este principio permite realizar mediciones de fuerzas muy tenues que provoquen pequeñas deformaciones en el conductor. Este principio sirve en la construcción de las galgas extensiométricas, que son dispositivos transductores pasivos que permiten medir la fuerza ejercida sobre él a partir de la deformación resultante. Así, fuerzas de compresión, tracción o torsión, aplicadas sobre materiales elásticos, generan deformaciones que son transmitidas a la galga, respondiendo ésta con una variación de su propia resistencia eléctrica [11].

Su principio de funcionamiento se basa en el efecto piezorresistivo de metales y semiconductores, según el cual, su resistividad varía en función de la deformación a la que están sometidos, el material de que está hecho y el diseño adoptado [12].

En la Tabla 1.5. Se muestran las características principales de las diferentes galgas extensiométricas con su rango de peso, aplicaciones, ventajas, y desventajas.

Tabla 1.5. Diferencias entre tipos de celdas de cargas [13].

Tipo de galga	Tolerancia	Ventajas	Desventajas	Aplicaciones
Viga de Flexión	5 – 2500 Kg	Bajo Costo. Construcción sencilla.	Requiere protección	Tanques
Cizallamiento	5 – 2500 Kg	Rechazo de altas cargas	Alto costo	Cargas descentradas
Depósitos	250 ton.	Movimientos de carga	No presenta protección de carga horizontal	Camiones. Tanques
Compresión	250 – 1500 Kg	Robusta. Fácil de instalar	Solo para cargas fijas en lugares planos	Mezcladores. Reactores

1.3.4.2. Módulo HX711 transmisor de celda de carga

El módulo HX711 es un transmisor entre las celdas de carga y un microcontrolador como Arduino, permitiendo leer el peso en la celda de manera sencilla. Es compatible con las celdas

de carga de 1kg, 5kg, 20kg y 50kg. Utilizado en sistemas de medición automatizada, procesos industriales, industria médica.

El chip HX711 posee internamente la electrónica para la lectura del puente de Wheatstone formado por la celda de carga y también un conversor ADC de 24 bits. Se comunica con el microcontrolador por medio de un protocolo de tipo serial mediante 2 pines (Clock y Data).

Las celdas de carga están formadas por galgas extensiométricas en configuración de puente Wheatstone. Para conectar la celda al módulo HX711 son necesarios 4 cables, los colores utilizados habitualmente son Rojo, Negro, Blanco y Verde. Cada color corresponde a una señal como se muestra a continuación:

- Rojo: Voltaje de excitación +, E+, VCC
- Negro: Voltaje de excitación -, E-, GND
- Verde: Amplificador -, Señal -, A-
- Blanco: Amplificador +, Señal +, A+

1.3.4.3. Placa Arduino UNO

Arduino Uno es una placa electrónica basada en el microcontrolador ATmega328. Cuenta con 14 entradas/salidas digitales, de las cuales 6 se pueden utilizar como salidas PWM (Modulación por ancho de pulsos) y otras 6 son entradas analógicas. Además, incluye un resonador cerámico de 16 MHz, un conector USB, un conector de alimentación, una cabecera ICSP y un botón de reseteado. La placa incluye todo lo necesario para que el microcontrolador haga su trabajo, basta conectarla a un ordenador con un cable USB o a la corriente eléctrica a través de un transformador [14].

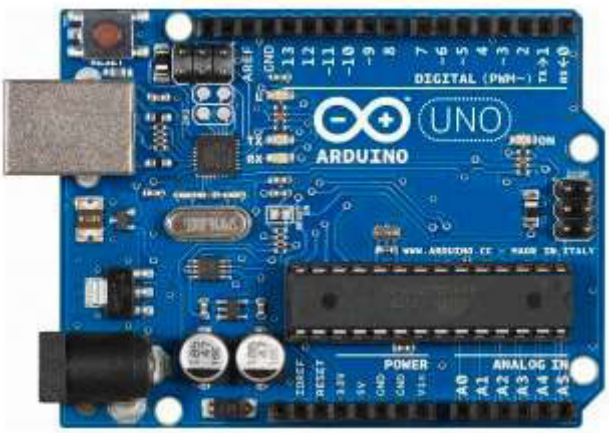


Figura 1.10. Placa Arduino UNO

Características técnicas de Arduino Uno:

- Voltaje: 5V
- Pines Digitales de Entrada/Salida: 14 (de los cuales 6 son salida PWM)
- Entradas Analógicas: 6
- Amperaje DC en Pines Entrada/Salida: 40 mA
- Memoria Flash: 32 KB de los cuales 0.5 KB son utilizados para el arranque
- Velocidad de Reloj: 16 MHz.

1.3.4.4. IDE de Arduino UNO

Es un software oficial presentado por Arduino.cc, que se utiliza principalmente para editar, compilar y cargar el código en el dispositivo Arduino. IDE son las siglas de "Integrated Development Environment". Casi todos los módulos Arduino son compatibles con este software. Es un programa de código abierto. Está disponible para sistemas operativos como MAC, Windows, Linux y se ejecuta en la plataforma Java. Viene incorporado con funciones y comandos para depurar, editar y compilar el código en el entorno [15].

El código principal, creado en la plataforma IDE generará un código hexadecimal. El archivo hexadecimal se transfiere y se carga en el controlador de la placa del Arduino.

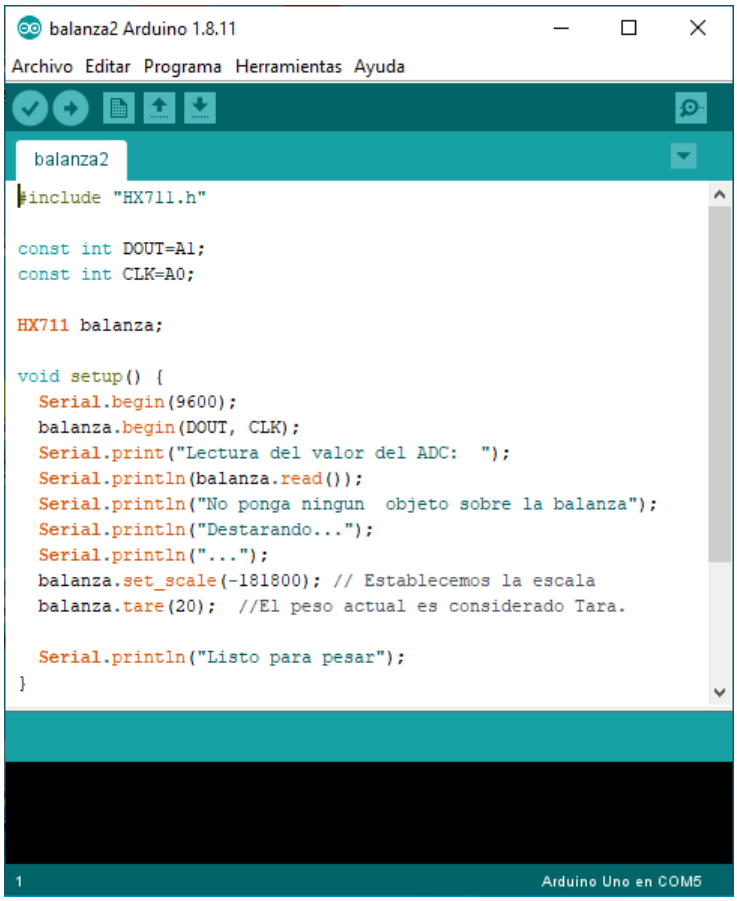


Figura 1.11. Entorno IDE de Arduino.

1.3.5. APP DESIGNER

Este programa es un entorno de desarrollo interactivo para diseñar una aplicación y programar su comportamiento. Proporciona una versión completamente integrada de MATLAB y un gran conjunto de componentes de interfaz de usuario interactivas. Permite distribuir aplicaciones, empaquetándolas en archivos de instalación directamente desde la barra de herramientas de App Designer, o creando una aplicación web o de escritorio independiente.

Para ingresar a la página de inicio de App Designer, se escribe en el Command Window de Matlab la palabra "***appdesigner***" y se despliega la página de inicio del programa.

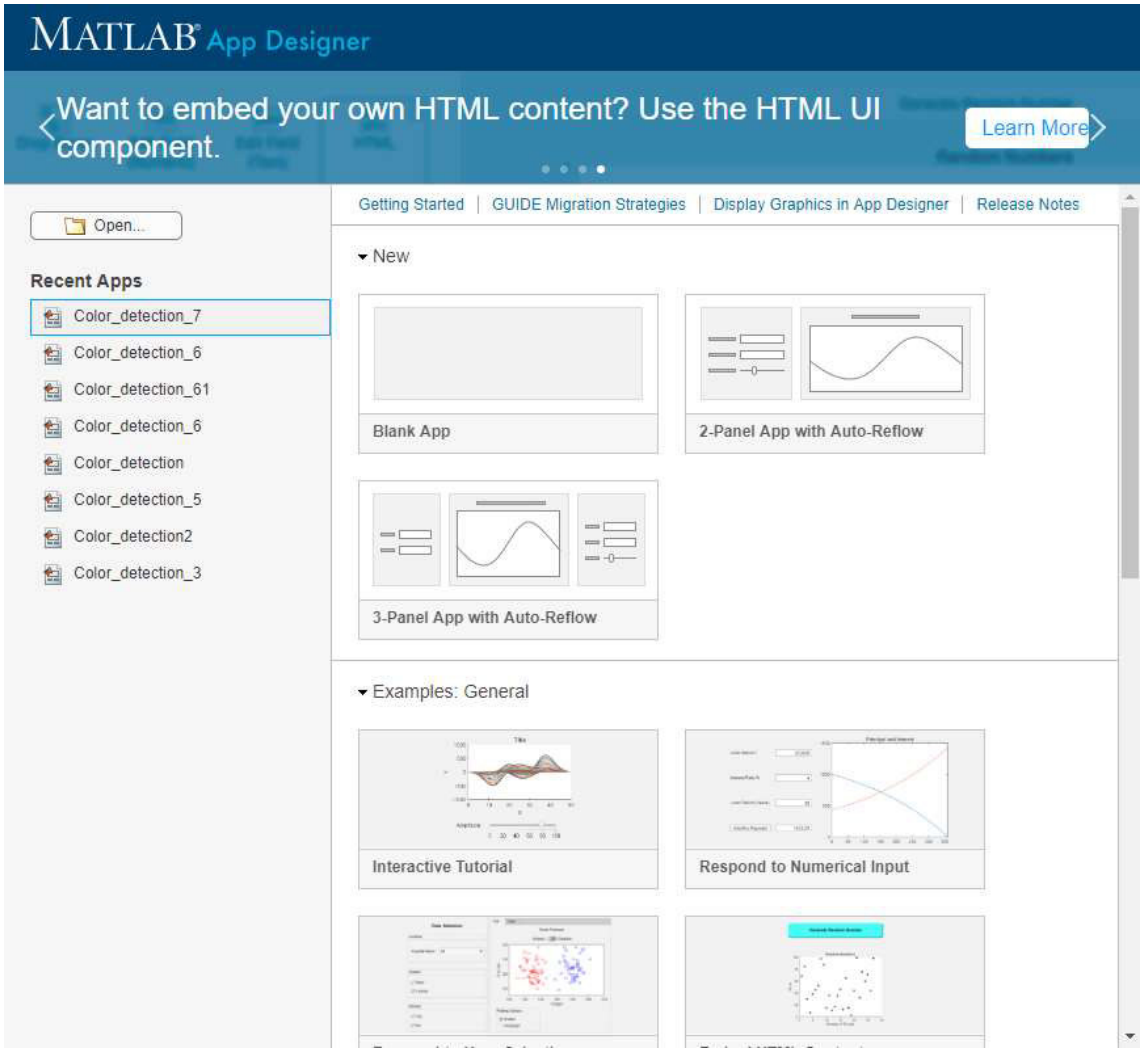


Figura 1.12. Página de inicio de App Designer.

Al momento de dar click en "Blank App" se despliega un lienzo en blanco en donde se diseñará la interfaz gráfica del sistema autónomo para determinar el precio de frutas.

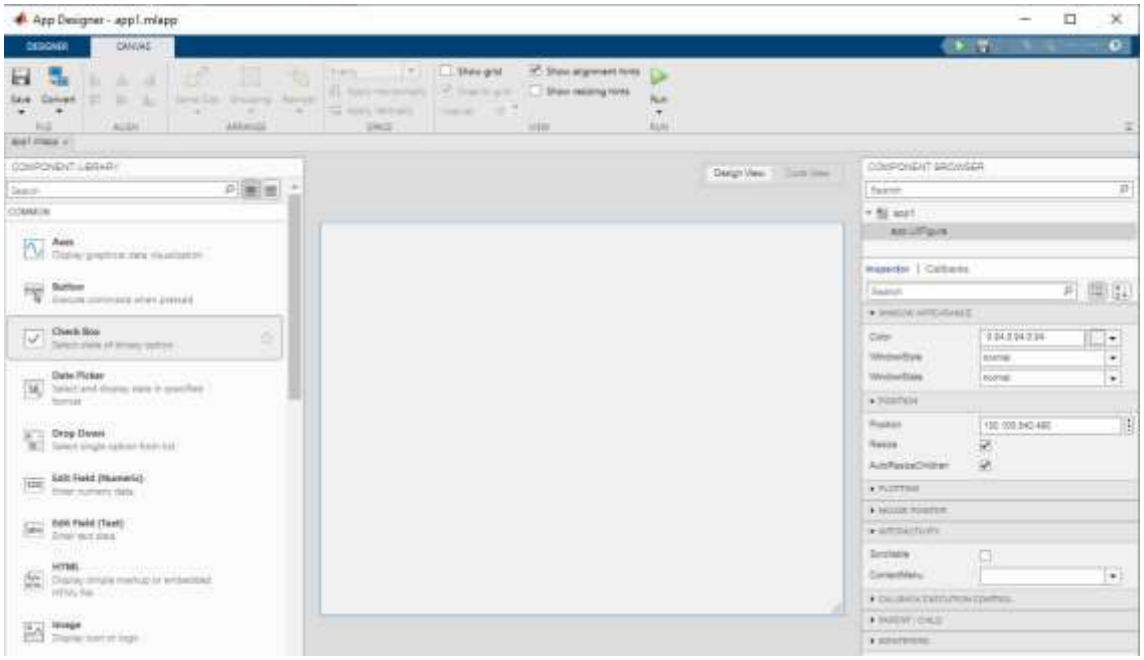


Figura 1.13. Entorno de App Designer para el diseño de la intefaz gráfica del sistema autónomo para determinar el precio de frutas basado en visión Artificial.

En la Figura 1.13. se muestra el lienzo en blanco para el diseño de la interfaz de usuario. Las herramientas que se pueden utilizar que observan al lado izquierdo, mientras que en la parte derecha se muestran todas las herramientas que se están utilizando dentro del lienzo. La manera de trabajar con las herramientas es a través de objetos que tienen un nombre asignado para su respectiva configuración.

2. METODOLOGÍA

En este capítulo se detallan los cuatro módulos que componen el sistema autónomo para determinar el precio de frutas basado en visión artificial tal como se observó en la Figura 1.1. Además, se detalla paso a paso la construcción del prototipo creado para este fin, tanto la parte destinada al módulo de pesaje como la que servirá para la adquisición de imágenes. También se mostrarán los programas, librerías, toolbox, y respectivas codificaciones que hacen que funcione la parte de recolección de datos e identificación de la fruta del prototipo diseñado. En la Figura 2.1. se observa un diagrama de flujo del funcionamiento del prototipo con sus respectivos módulos de la siguiente manera:

MODULO I → Pesaje de frutas.

MODULO II → Adquisición de imágenes.

MODULO III → Reconocimiento del tipo de fruta mediante Visión Artificial.

MODULO IV →Cálculo del valor de la fruta ya pesada.

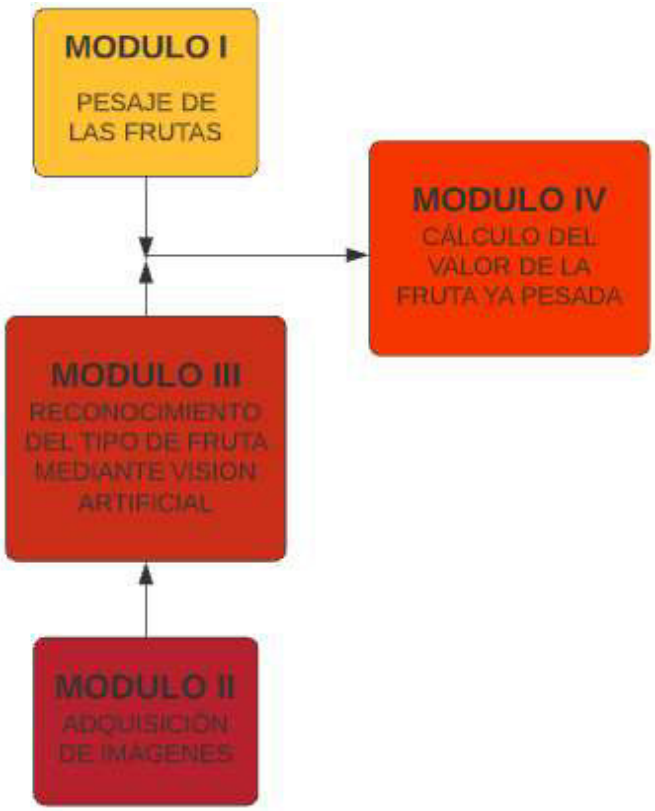


Figura 2.1. Diagrama de flujos del sistema autónomo para determinar el precio de una fruta.

La Figura 2.1. También puede ser vista de forma gráfica al representar cada módulo dentro del prototipo que se construirá para el sistema autónomo para determinar el precio de frutas

como se visualiza en la Figura 2.2. De esta manera se empezará a describir cada bloque por separado dentro de este capítulo.

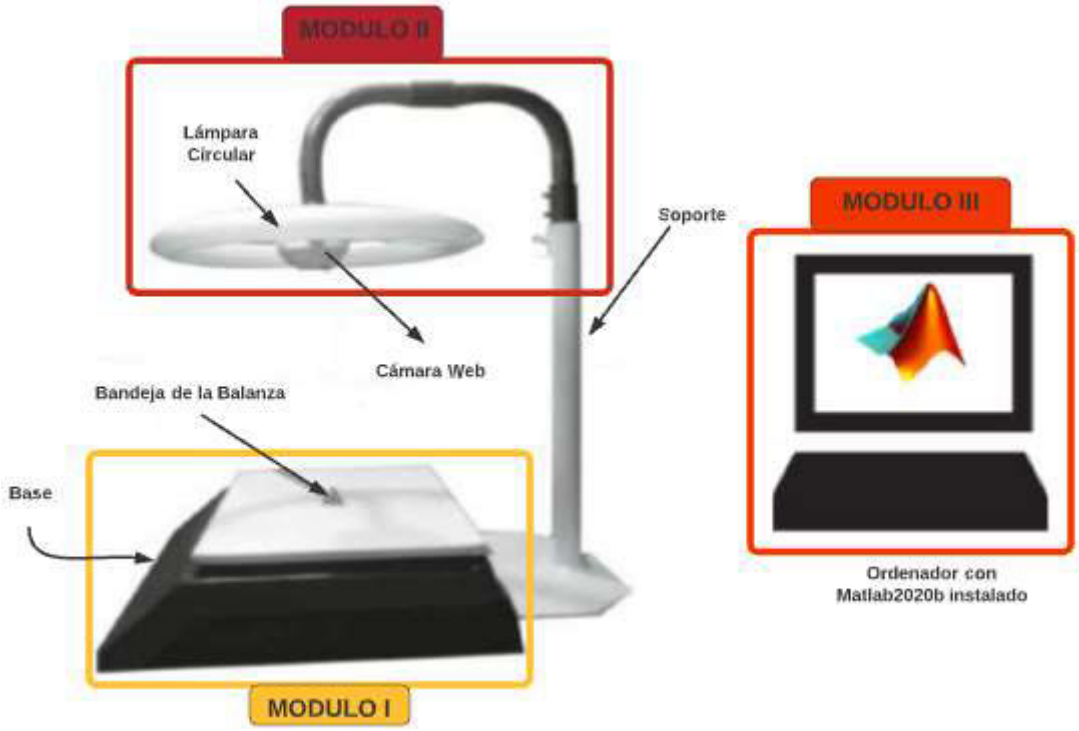


Figura 2.2. Representación gráfica de cada módulo que contiene el sistema autónomo para determinar el precio de frutas.

2.1. MÓDULO DE PESAJE DE LAS FRUTAS

En esta etapa se realiza el proceso del pesaje de la fruta. En primer lugar, se construyó una balanza cuyas piezas están hechas de metal. Posteriormente, a la balanza se le agregaron dispositivos que ayudarán a tomar el valor del peso de las frutas y transferirlas a la PC. Finalmente, se codificó un programa en IDE de Arduino para convertir la señal recibida de los dispositivos anteriores y obtener el peso de la fruta.

2.1.1. CONSTRUCCIÓN DE LA BALANZA DIGITAL

Aquí se detalla el proceso de construcción de la balanza metálica. Se eligió dicho material para lograr una balanza resistente y capaz de soportar un peso de frutas máximo de 5 kilogramos y una serie de pruebas que se realizarán posteriormente con este prototipo. La Figura 2.3. representa las partes que conforman la balanza construida.

Figura 2.3. Partes principales de la balanza: De arriba hacia abajo se aprecia la bandeja y la base de ésta.

De la Figura 2.3. se pueden apreciar las siguientes partes:

- Base: Superficie que da soporte a todos los elementos de la balanza, además de servir de base para el sistema de adquisición de imágenes, el cual se detallará posteriormente. Las dimensiones de la base son 40x40x7 centímetros, útiles para resguardar a la bandeja y servir como ancla al soporte del sistema de adquisición de imágenes.

Figura 2.4. a) Vista superior de la base de la balanza digital. b) Vista lateral de la base de la balanza digital.

- Bandeja: Ubicada en la parte superior y será donde se coloquen las frutas para su debido pesaje. Las dimensiones de la bandeja son 40x40 centímetros. El área que cubre la bandeja es menor al área la base para evitar golpes directos a la misma y

que en caso de accidentes, éstos afecten solamente a la base, dejando ilesa a la bandeja.

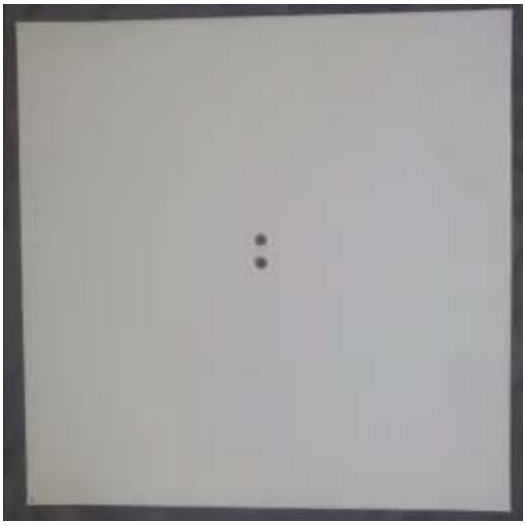


Figura 2.5. Vista superior de la bandeja de la balanza digital.

2.1.2. CONSTRUCCIÓN DEL SOPORTE PARA LA ADQUISICIÓN DE IMÁGENES

Tanto la cámara como la lámpara circular que forman parte del sistema de adquisición de imágenes, que se verán posteriormente con más detalle, se encuentran ubicadas en un soporte de metal lo cual permite posicionarlas en el centro de la parte superior de la bandeja tal como muestra la Figura 2.6.

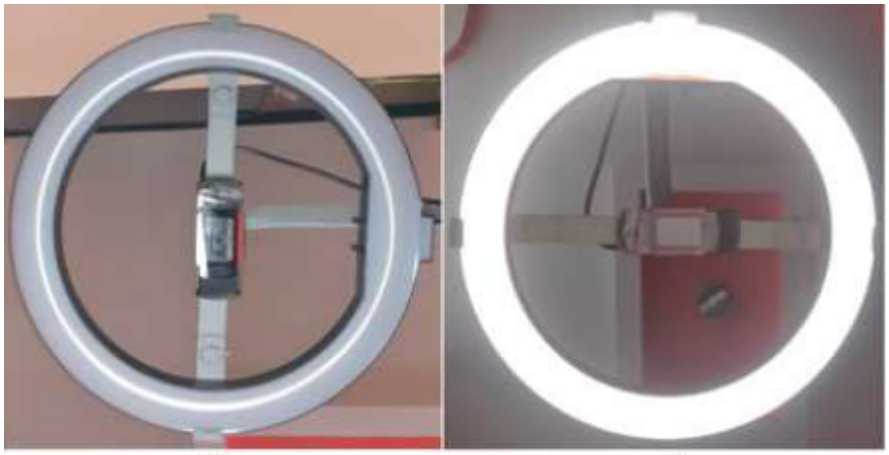


Figura 2.6. Cámara web y lámpara circular ubicadas en un soporte metálico para iluminar la bandeja. a) lámpara apagada b) lámpara encendida

El soporte de metal tiene una longitud de 60 centímetros desde su base hasta la parte más alta de ésta. Tiene una curvatura en la parte superior y consta de un sujetador en su extremo para colocar la lámpara circular y la cámara web. La Figura 2.7. es una visualización del doporte antes de ser fijada a la base de la balanza.

Figura 2.7. Soporte del módulo de adquisición de imágenes.

En la Figura 2.8. Se observan los orificios que se encuentran tanto en el soporte como en la base de la balanza para que se unan a través de tuercas. Finalmente, se muestra cómo es que quedan unidas ambas estructuras.

Figura 2.8. a) Orificios de la base del soporte del sistema de adquisición de imágenes. b) Orificios de la base de la balanza digital. c) Tuercas para unir el soporte del sistema de adquisición de imágenes con la balanza digital. d) Unión terminada entre el soporte del sistema de adquisición de imágenes con la balanza digital.

El soporte se une a la base de la balanza por medio de pernos y tuercas, conformando así la estructura del sistema de pago autónomo de frutas como se muestra en la Figura 2.9.

Figura 2.9. a) Vista lateral y b) vista frontal de la estructura del prototipo del sistema autónomo de pago de frutas a través de visión artificial.

2.1.3. PROTOTIPO DEL SISTEMA AUTÓNOMO DE PAGO DE FRUTAS A TRAVÉS DE VISIÓN ARTIFICIAL.

Con la unión de la lámpara circular y la cámara web a la estructura mostrada en las figuras anteriores queda completo el prototipo del sistema autónomo de pago de frutas a través de visión artificial como se muestra en la Figura 2.10.

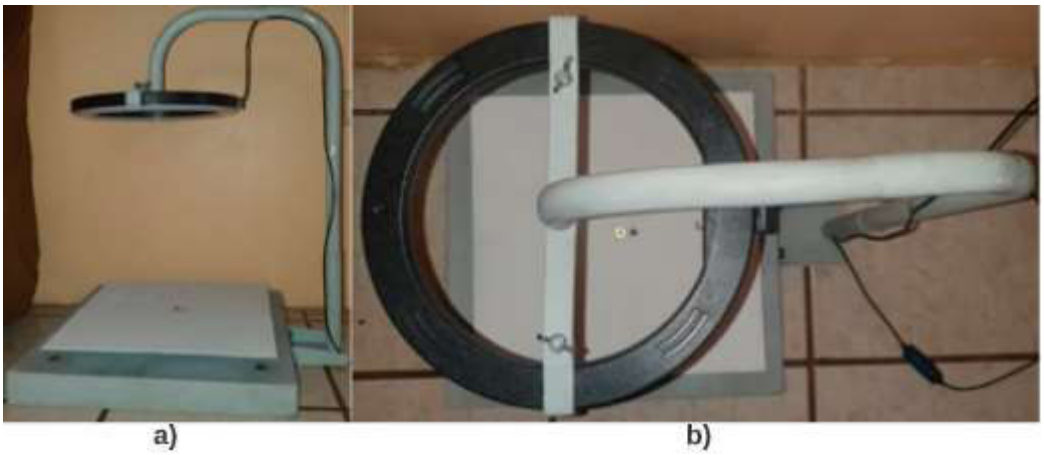


Figura 2.10. a) Vista lateral y b) Vista superior del sistema del prototipo del sistema autónomo de pago de frutas a través de visión artificial.

2.1.4. DISPOSITIVOS PARA TOMAR EL VALOR DEL PESO DE LAS FRUTAS

Además de las partes mostradas en las figuras anteriores, entre la base y la bandeja se encuentra una celda de carga. El cual es un dispositivo electrónico que convierte la tensión aplicada en una señal eléctrica [16].

Figura 2.11. Celda de carga de tolerancia máxima de 5 kilogramos.

Para sujetar la celda de carga a la base y a la bandeja de la balanza construida se utilizaron dos soportes. El primero y sujeto a la base de dimensiones 3x3x0.5 centímetros para que exista un espacio considerable entre la celda de carga y la superficie de la base y que ambos no se lleguen a topar directamente. Mientras que el segundo soporte, de dimensiones 5x5x0.5 centímetros, unirá a la superficie inferior de la bandeja con el otro extremo de la celda de carga. El área del segundo soporte es mayor para abarcar la mayor área posible de la bandeja y ganar estabilidad en esta.

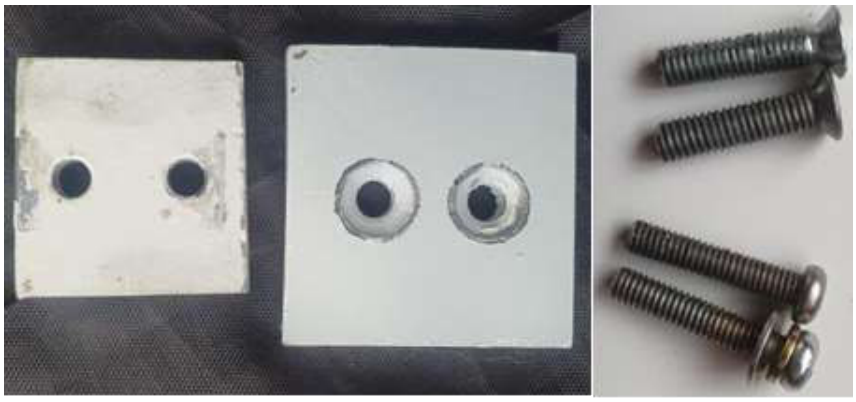


Figura 2.12. Soportes utilizados con sus respectivas tuercas para fijar la celda de carga entre la base y la bandeja de la balanza.

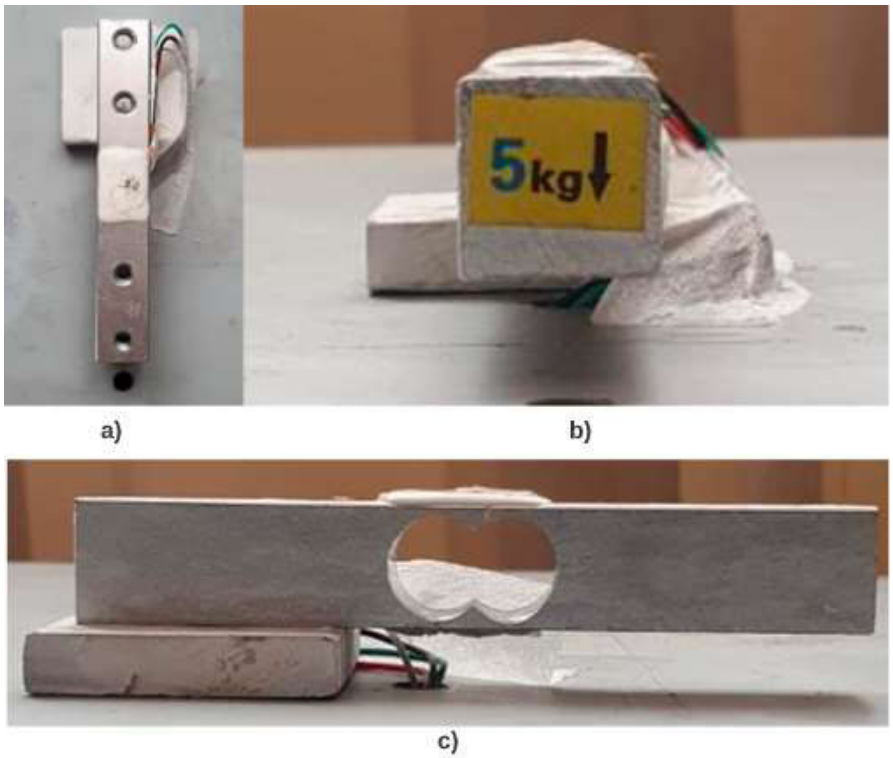


Figura 2.13. Celda de carga sujeta la base de la balanza, vista desde diferentes puntos antes de colocar la bandeja. a) Vista superior b) Vista frontal c) Vista lateral.

La Figura 2.13. muestra a la celda de carga cómo queda adherida a la base desde diferentes vistas, procurando que ambas superficies no tengan contacto con la ayuda del suporte de 0.5 centímetros de grosor. En la Figura 2.14. se muestra finalmente a la base y a la bandeja unidas por la celda de carga y el uso de soportes para evitar el contacto entre las superficies de estos tres elementos.

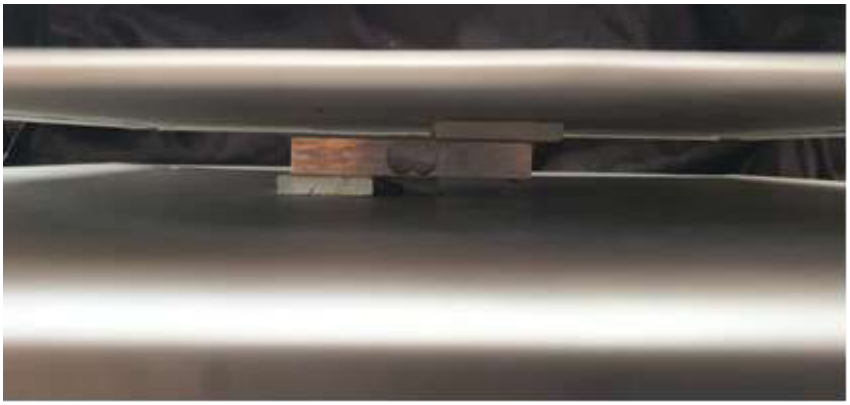


Figura 2.14. Vista lateral de la celda de carga sujetando a la base y a la bandeja a través de dos soportes que evitan el contacto de sus superficies.

De la celda de carga se observan cuatro cables de color rojo, negro, verde, y blanco. Los mismos que deberán ser soldados al transmisor de peso HX711. Un transmisor de peso permite la lectura del valor de la celda de carga al ser una interfaz entre ésta y el microcontrolador del Arduino UNO. Dado a que en su interior se procesa la lectura de la tensión que se produce en la celda de carga, convirtiendo la lectura analógica en digital con la ayuda de un conversor A/D interno.

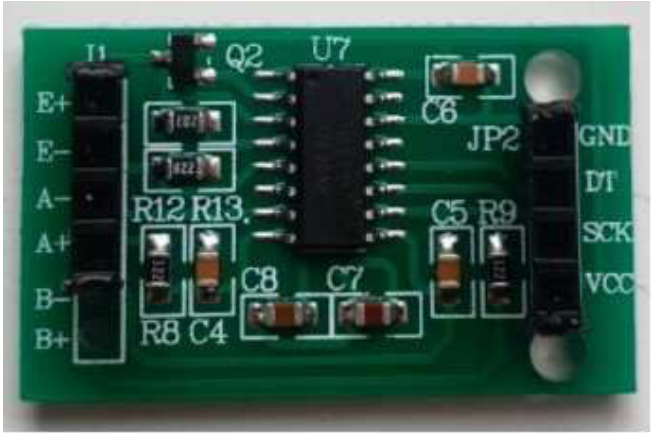


Figura 2.15. Transmisor de peso HX711

La señal que produce el transmisor de peso HX711 es recogida por el Arduino UNO, el cual se encarga de transmitir el valor a la PC y obtener el valor del peso de las frutas.

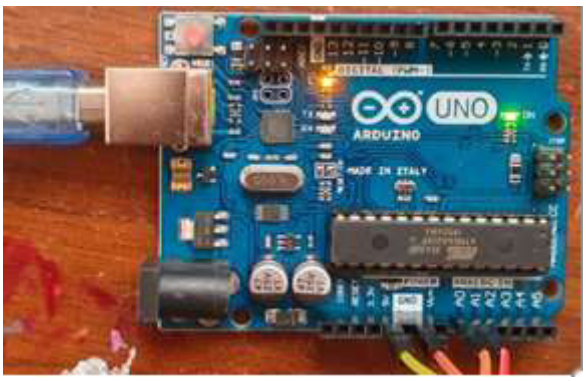


Figura 2.16. Arduino UNO utilizado para recoger la señal del transmisor HX711 y enviarlo a la PC

La conexión entre el Arduino UNO y el transmisor de peso HX711 se lo hace a través de cuatro cables de 18 centímetros. Mientras que la conexión entre el Arduino UNO con la PC se lo hace con un cable USB con salida tipo B para el Arduino y salida tipo A para la PC.

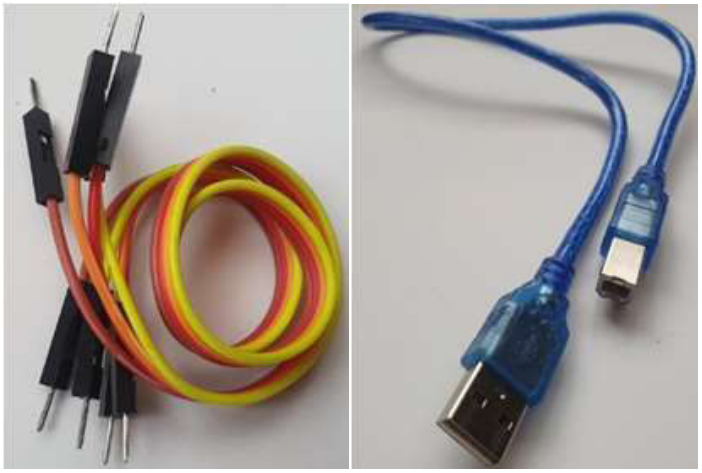


Figura 2.17. Cables de 18 centímetros y cable USB

2.1.4.1. Conexión de pines entre dispositivos

En este segmento se presentan las respectivas conexiones entre los dispositivos encargados de recoger la señal generada por el peso de las frutas y enviarlas a la PC para determinar el costo de estas. En primer lugar, se tiene la conexión entre la celda de carga (la cual cuenta con cuatro cables) con los pines del lado izquierdo E+, E-, A-, A+ del transmisor de peso HX711. Se tienen dos pines adicionales B- y B+, sin embargo, no serán utilizados para esta aplicación.

Tabla 2.1. Conexión de pines entre la celda de carga y el transmisor de peso HX711 [17].

Cables Celda de carga	Pines Transmisor de peso HX711
Rojo	E+
Negro	E-
Verde	A-
Blanco	A+

La siguiente conexión de pines se la hace entre los pines del lado derecho GND, DT, SCK, y VCC del transmisor de peso HX711 con los pines GND, A1, A0, y 5V del Arduino UNO. En el caso de los pines A1 y A0 es necesario mencionar que se pueden escoger cualquier otro pin de entrada con los que cuenta Arduino UNO, no necesariamente los utilizados en esta aplicación. Es así que, la conexión de pines queda como se muestra en la Tabla 2.2.

Tabla 2.2. Conexión de pines entre el transmisor de peso HX711 y el Arduino UNO [18].

Pines Transmisor de peso HX711	Arduino UNO
VCC	5V
DT	A1
SCK	A0
GND	GND

Las conexiones mencionadas en las tablas anteriores se llevaron a cabo soldando los cables directo al transmisor de eso y utilizando cables de 18 centímetros como se muestra en la **Figura 2.18.**

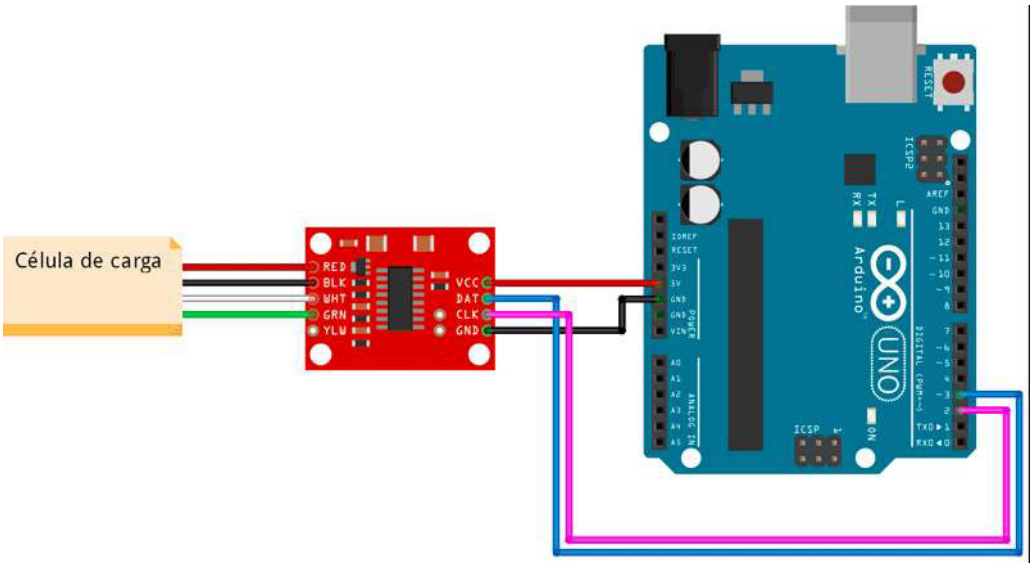


Figura 2.18. Esquema de conexión de pines con soldadura y cables [19].

2.1.5. PROCESO DE INSTALACIÓN DEL IDE DE ARDUINO

El software se puede descargar desde el sitio web principal de Arduino https://www.arduino.cc/ [20]. Se da click en el enlace SOFTWARE y se despliega la ventana vista en la Figura 2.19.

Figura 2.19. Venta de SOFTWARE de la oficial de Arduino.

Como se aprecia en la Figura 2.19., el software está disponible para sistemas operativos como Windos, Linux, o MACos. Se da click en la versión que se tenga en el ordenador y se abre una ventana en la que se puede donar dinero a la organización que desarrolla Arduino o, por el contrario, simplemente descargar el instalador e inicia el proceso de descarga.

Figura 2.20. Cuadro de diálogo para contribuir a los desarrolladores de IDE de Arduino

El instalador del IDE de Arduino tiene un tamaño aproximado de 112 Megabytes y al ejecutarlo aparece en primer lugar el acuerdo de licencia al que daremos aceptar.

Figura 2.21. Acuerdo de licencia de Arduino Setup.

Durante el proceso de instalación aparecen una serie de ventanas emergentes de Seguridad de Windows en la que preguntan al usuario si se desea instalar los drivers con los que se tendrá comunicación entre el dispositivo Arduino con la PC, al cual damos “Instalar” a todas.

Figura 2.22. Ventana emergente de Seguridad de Windows preguntando al usuario si desea instalar los drivers del IDE de Arduino.

Finalmente, El firewall de Windows emergerá, preguntando al usuario si permite el acceso del software a redes públicas, a los cual damos “Permitir Acceso”.

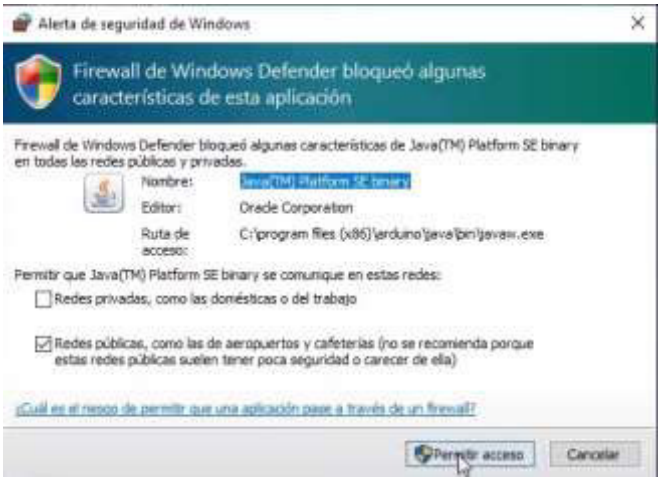


Figura 2.23. Firewall de Windows Defender preguntando al usuario si permite el acceso del software IDE de Arduino a redes públicas.

2.1.6. PROCESO DE INSTALACIÓN DE LA LIBRERÍA HX711

Para trabajar con celdas de carga en el IDE de Arduino es necesario la instalación de la librería para el módulo HX711. Para esto se debe ingresar a “Gestión de Librería” aplastando las teclas “CTRL+SHIFT+I” al mismo tiempo.

Después de aplastar dichas teclas al mismo tiempo, se abre una ventana llamada “Gestión de Librerías” como se muestra en la figura.

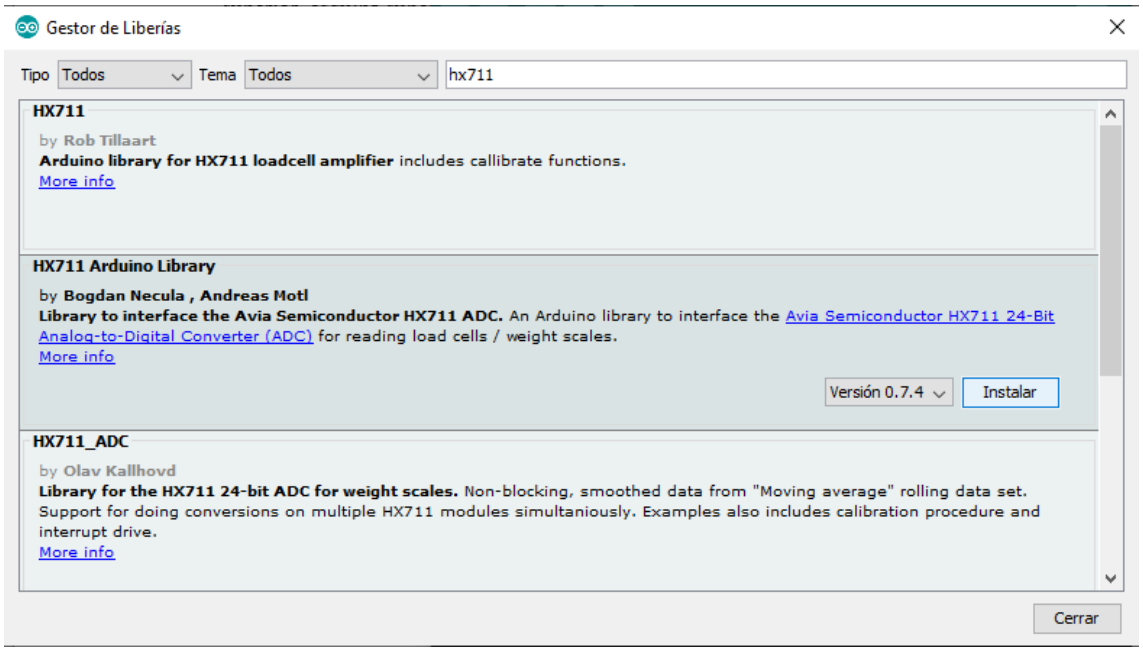


Figura 2.24. Entorno del Gestor de Librería

En la barra superior en donde se muestra el mensaje “Filtre su búsqueda” escribimos “HX711” y escogemos la primera opción “HX711 Arduino Library” de Bogdan Necula, Andreas Motl.

como se muestra en la figura. Finalmente presionamos instalar para obtener todos los comandos de dicha librería. Una vez terminada la instalación salimos de la ventana emergente haciendo click en “cerrar”.

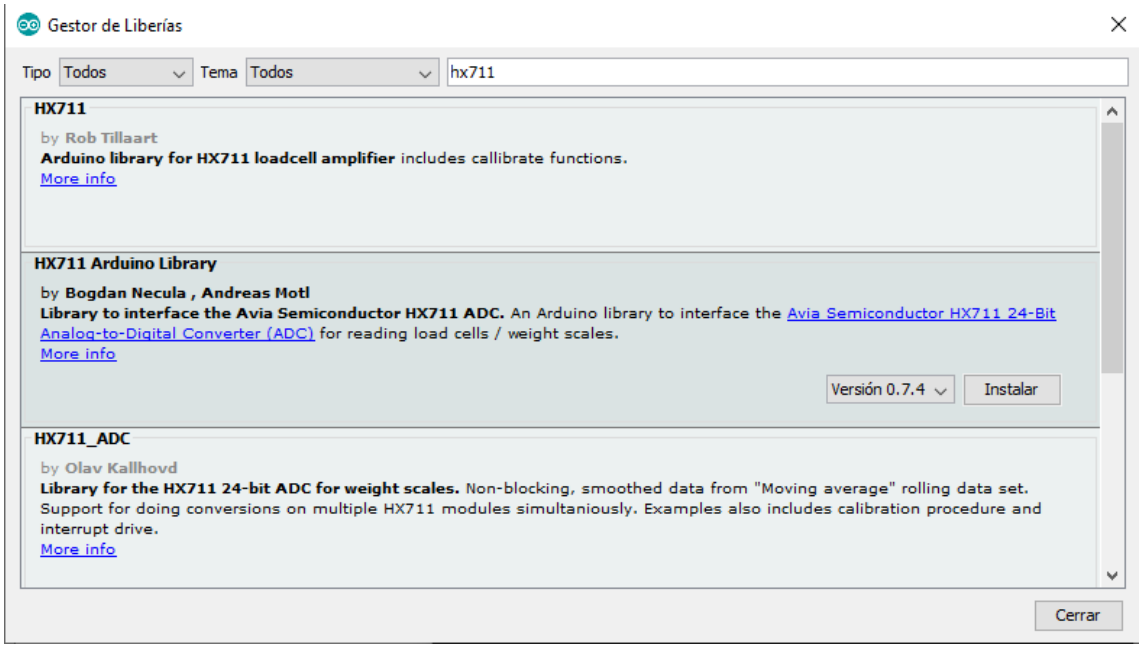


Figura 2.25. Selección de la Librería HX711 Arduino Library.

Al momento de conectar al dispositivo Arduino UNO con la PC a través del cable USB, el IDE de Arduino comunica al usuario el COM que se está utilizando para la comunicación entre ambos aparatos. El COM se muestra en la parte del Panel de Salida del entorno del IDE de Arduino. Como se aprecia en la Figura 2.26. el dispositivo Arduino Uno está conectado a la PC en el COM5.

Figura 2.26. Panel de Salida del entorno del IDE de Arduino donde se indica que el dispositivo Arduino está conectado a la PC a través del COM5

Una vez nos hemos asegurado que existe conectividad entre PC y dispositivo Arduino y de que está instalada la librería HX711 se consideraron dos procesos para asegurar el correcto funcionamiento de la balanza que son los de calibración y pesaje. Los códigos implementados, así como los comandos se detallarán en los siguientes sub-capítulos.

2.1.7. PROCESO DE CALIBRACIÓN

Para llamar a la librería que controla el módulo HX711 se utiliza el comando "***#include***" acompañado del nombre de ésta entre comillas:

```
#include "HX711.h"
```

Se definen los pines del Arduino que se utilizarán para la recepción de la señal del transmisor HX711. Es así que el pin A1 servirá la señal como tal, mientras que el pin A0 servirá como señal de reloj.

```
byte pinData = A1;
byte pinClk = A0;
```

Para llamar a todas las funciones que contiene la librería "HX711" se crea un objeto al que le llamaremos "bascula".

```
HX711 bascula;
```

Se inicia el segmento de código con el comando "***void setup ()***", acompañado de una llave, de la siguiente manera:

```
void setup() {
```

Se inicia el módulo HX711 definiendo los pines que se utilizarán en el Arduino, es decir ***pinData*** y ***pinClk***.

```
bascula.begin(pinData, pinClk);
```

Se inicia la comunicación con la PC a través del puerto serial a una velocidad de símbolo de 9600 baudios.

```
Serial.begin(9600);
```

Se imprime el siguiente texto “El valor del ADC es:” con el comando “***Serial.print***” para acompañar al valor de peso inicial, es decir, el peso de la bandeja junto con el soporte que lo une a la celda de carga.

```
Serial.print ("El valor del ADC: ");
```

A continuación, se imprime el valor medido de la lectura del módulo HX711 con el comando “***Serial.println***”.

```
Serial.println(bascula.read());
```

Para comenzar con el proceso de calibración propiamente dicho, se imprime un mensaje de alerta para evitar que se pongan frutas sobre la bandeja mientras se ejecuta este proceso.

```
Serial.println("No ponga ningún objeto sobre la balanza");
```

Para indicar que el proceso aún se encuentra en ejecución se imprime un mensaje con el texto “destarando…”.

```
Serial.println("destarando...");
```

En la siguiente línea se coloca el valor de escala que funcionará en la balanza. Por el momento se envió un valor de escala por defecto.

```
bascula.set_scale();
```

Para determinar el valor de escala se utilizará un peso conocido que pesa un libra exactamente. El valor que se muestre al momento de ejecutar el código será el valor de escala que se escribirá en la línea de debajo de acuerdo a Ecuación 2.1.

$$Valor\ de\ escala = \frac{Valor\ promedio\ de\ lectura}{Peso\ conocido} \tag{2.1}$$

Como el peso conocido es de 1lb reemplazamos dicho valor en la Ecuación 2.1 obteniendo la Ecuación 2.2:

$$Valor\ de\ escala = Valor\ promedio\ de\ lectura \tag{2.2}$$

El valor de 20 es el número de muestras que se realizarán para obtener el valor de la tara.

```
bascula.tare(20);
```

Se imprime un mensaje para indicar a la persona encargada de la calibración que coloque el peso conocido y así obtener el valor de calibración.

```
Serial.println("Listo para pesar");
```

Terminamos la parte de codificación de "***void setup***" con una llave.

```
}
```

Iniciamos un segmento de código redundante para obtener el valor de calibración

```
void loop() {
```

Se imprime un mensaje indicando el valor de calibración que se debe colocar en "***set scale***"

```
Serial.print("Valor de calibración: ");
```

Obtenemos el valor con el comando "***get value***" como se muestra en la siguiente línea. El valor de 20 indica que se tomarán 20 valores previos para imprimir el valor de calibración.

```
Serial.println(bascula.get_value(20));
```

Se añade un retardo de 100 milisegundos.

```
delay(100);
```

Se termina el segmento iterativo con una llave

```
}
```

Es así que el código completo de calibración queda de la siguiente forma:

```
HX711 bascula; // OBJETO HX711

void setup() {

bascula.begin(pinData, pinClk); // Iniciar modulo HX711 definiendo sus pines

Serial.begin(9600); //inicio de comunicación con PC por puerto serial a 9600 baudios

Serial.print ("El valor del ADC: ");

Serial.println(bascula.read()); //lectura ADC de HX711

Serial.println("No ponga ningún objeto sobre la balanza");

Serial.println("destarando...");

bascula.set_scale(); //establecer escala por defecto (1)

bascula.tare(20); //considerar cero al peso actual (tarar)

Serial.println("Coloque peso conocido: ");

}

void loop() {

  Serial.print("Valor de calibración: ");

  Serial.println(bascula.get_value(20)); // obtener valores para factor de conversion

  delay(100);

}
```

Finalmente, se tomaron 20 muestras del valor de lectura y se hizo un promedio para determinar el valor de calibración

```
Valor de lectura: -181809.00
Valor de lectura: -181825.00
Valor de lectura: -181821.00
Valor de lectura: -181809.00
Valor de lectura: -181836.00
Valor de lectura: -181781.00
Valor de lectura: -181789.00
Valor de lectura: -181804.00
Valor de lectura: -181784.00
Valor de lectura: -181772.00
Valor de lectura: -181768.00
Valor de lectura: -181769.00
Valor de lectura: -181759.00
Valor de lectura: -181771.00
Valor de lectura: -181777.00
Valor de lectura: -181765.00
Valor de lectura: -181778.00
Valor de lectura: -181792.00
Valor de lectura: -181795.00
Valor de lectura: -181798.00
```

Figura 2.27. Valores de lectura del peso de calibración

De los 20 valores se determinó que el valor de calibración es -187900.00 con la ayuda de la Ecuación 2.2.

$$Valor\ de\ escala = Promedio\ de\ los\ 20\ primeros\ Valores\ de\ lectura = -181790.00$$

2.1.8. PROCESO DE PESAJE

Para el proceso de pesaje se utiliza el mismo código del proceso de calibración con la diferencia de que ahora se escribirá el valor de escala en la línea "***set scale***".

```
balanza.set_scale(-181790);
```

Así mismo, en el segmento iterativo se imprime el mensaje de "Peso:" para acompañar al valor medido por la báscula.

```
Serial.print("Peso: ");
```

Se imprime el valor medido por la balanza con el comando "***get units***". El valor de 20 indica que se tomarán 20 muestras antes de imprimir un valor.

```
Serial.print(balanza.get_units(20));
```

Se acompaña al valor con el sufijo de libras “lb” ya que esta es la unidad de los valores que se obtengan del pesaje de frutas.

```
Serial.println(" lb");
```

El código completo del proceso de pesaje quedaría de la siguiente manera:

```
#include "HX711.h"

const int DOUT=A1;

const int CLK=A0;

HX711 balanza;

void setup() {

  Serial.begin(9600);

  balanza.begin(DOUT, CLK);

  Serial.print("Lectura del valor del ADC:  ");

  Serial.println(balanza.read());

  Serial.println("No ponga ningun  objeto sobre la balanza");

  Serial.println("Destarando.");

  Serial.println(".");

  balanza.set_scale(-181790); // Establecemos la escala

  balanza.tare(20);  //El peso actual es considerado Tara.

  Serial.println("Listo para pesar");

}

void loop() {

  Serial.print("Peso: ");

  Serial.print(balanza.get_units(20));

  Serial.println(" lb");

  delay(100);

}
```

Finalmente podemos apreciar el correcto funcionamiento del módulo de pesaje y obteniendo el peso de las frutas en libras tal como se muestra en la Figura 2.28.

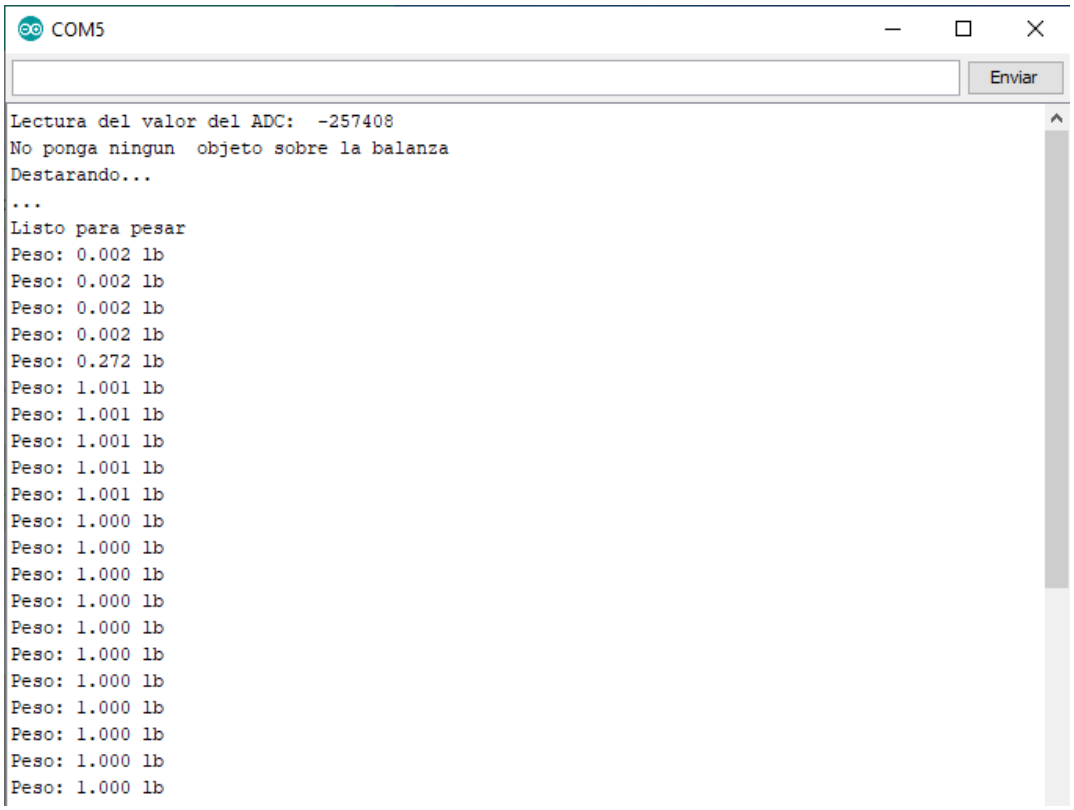


Figura 2.28. Valores que muestra el módulo de pesaje con el peso conocido de una libra.

2.1.9. CÁLCULO DE ERRORES EN EL VALOR DE CALIBRACIÓN DEL PESO DE LAS FRUTAS

En esta sección se pretende encontrar el rango de error de que se puede tener al momento de pesar las frutas con el valor de calibración encontrado. Para esto se aplicará la teoría de errores [21] que consiste en acotar el valor de las imprecisiones que se tengan al momento de pesar una fruta. Para esto te tomarán en cuenta los valores de calibración que se obtuvieron, en donde se tomó como valor verdadero el promedio (181790.00) de dichos valores.

En primer lugar, se determina el error absoluto con la Ecuación 2.3:

$$\Delta z = \frac{\sum_1^{20} |z_i - z^*|}{20} \tag{2.3}$$

En donde:

Δz *: Error Absoluto*

z_i *: Valor experimental*

z^* *: Valor Real*

Con la Ecuación 3 y los valores tomados en el proceso de calibración se determinó que el valor absoluto es

$$\Delta z = 340$$

El error absoluto nos ayudará a determinar el error relativo al dividir dicho valor por el valor real, es decir, en nuestro caso el valor promedio de los valores de calibración.

$$\varepsilon_r = \frac{\Delta z}{z^*} \tag{2.4}$$

En donde:

ε_r : Error relativo.

En base a la Ecuación 2.4 se determinó que el error relativo es igual a 0.00019

$$\varepsilon_r = 0.00019$$

Para determinar el error porcentual se multiplicará el error relativo por el 100% como se muestra en la Ecuación 2.5.

$$\varepsilon_{\%} = \varepsilon_r \times 100\% \tag{2.5}$$

Con la Ecuación 5 se determinó que el error porcentual es del 0.19% por lo que determinamos que el error es muy bajo y tenemos una lectura correcta del peso de las frutas.

2.2. MÓDULO DE ADQUISICIÓN DE IMÁGENES

En esta etapa se utilizará la Cámara web 480 HD USB para PC. Esta cámara proporcionará imágenes que usan el modelo RGB, donde cada color se representa en sus tres componentes, rojo (Red), verde (Green), y azul (Blue) [22]. Debido a esto se obtiene una matriz de tres dimensiones.

Figura 2.29. Cámara web 480 HD USB para PC que se utilizará en el módulo de adquisición de imágenes.

Además, la cámara se encuentra ubicada en el centro de una lámpara circular que iluminará a las frutas que se encuentran sobre la balanza para obtener una imagen libre de sombras y en un ambiente totalmente controlado.

Figura 2.30. Lámpara Circular que se utilizará para iluminar las frutas ubicadas sobre la bandeja de la balanza. [23]

Una característica importante de esta lámpara es que emite tres tipos de luces:

- Luz fría (4200-4500K).
- Luz cálida (3000-3500K).
- Luz diurna (6000-6500K).

La unidad que acompaña a cada valor del tipo de luz son los grados Kelvin, el cual es una unidad de medida utilizada para describir la tonalidad de una fuente específica de luz. Esta medida no está necesariamente relacionada con el calor proporcionado por la fuente de luz, sino con el color de la luz. Mientras más alto el valor Kelvin de la fuente de luz, más cerca el color a la luz del sol [24].

2.2.1. PROCESO DE CAPTURA Y ADQUISICIÓN DE IMÁGENES

Para que la cámara web pueda ser utilizada por el programa Matlab, primero es necesario tener instalado el Toolbox de adquisición de imágenes. En la Figura 2.31., se observa el nombre del toolbox disponible en la página de Matlab.

Figura 2.31. Toolbox de adquisición de imágenes de Matlab.

Para saber cuáles son los adaptadores de cámara disponibles y que los reconoce el programa de Matlab se utiliza el comando "***imaqhinfo***".

```
>> imaqhwinfo

ans =

  struct with fields:

    InstalledAdaptors: {'winvideo'}

        MATLABVersion: '9.9 (R2020b)'

          ToolboxName: 'Image Acquisition Toolbox'

       ToolboxVersion: '6.3 (R2020b)'
```

Se observa que el adaptador instalado es '***winvideo***'. Para obtener información del mismo se ingresa el comando "***imaqhwinfo('winvideo')***".

```
>> imaqhwinfo('winvideo')

ans =

  struct with fields:

     AdaptorDllName:
'C:\ProgramData\MATLAB\SupportPackages\R2020b\toolbox\imaq\supportpackages\gener
icvideo\adaptor\win64\mwwinvideoimaq.dll'

    AdaptorDllVersion: '6.3 (R2020b)'

          AdaptorName: 'winvideo'

            DeviceIDs: {[1]}

           DeviceInfo: [1×1 struct]
```

En la salida ***DevideIDs*** se observa que el programa reconoce un único dispositivo de captura de videos, el cual es el de la cámara web de la propia computadora. Se coloca el número 1 para acceder a su información.

```
>> imaqhwinfo('winvideo',1)

ans =

  struct with fields:

            DefaultFormat: 'MJPG_1280x720'

       DeviceFileSupported: 0

                DeviceName: 'Integrated Camera'

                  DeviceID: 1

     VideoInputConstructor: 'videoinput('winvideo', 1)'

    VideoDeviceConstructor: 'imaq.VideoDevice('winvideo', 1)'

          SupportedFormats: {1×18 cell}
```

La información recopilada acerca de la cámara web conectada mediante un puerto USB es de suma importancia, ya que a partir de estos datos se creará el objeto "***videoinput***" dentro de la variable llamada "video". En la misma línea se introduce el comando "***cell2mat***" para trabajar únicamente con matrices ordinarias de dimensiones 640x480x3.

```
video = videoinput(cell2mat(vinf.InstalledAdaptors), 2,'YUY2_640x480');
```

De la variable "video" se guardan las capturas que se necesiten de manera ilimitada en cada disparo, es por esto por lo que se escribe la característica "***Inf***".

```
set(video, 'FramesPerTrigger', Inf)
```

Además de especificar el número de capturas por disparo, también se especifica el espacio de color que desea utilizar cuando Matlab adquiere los datos de la imagen. En este caso se trabajará con imágenes del tipo RGB.

```
set(video, 'ReturnedColorspace', 'rgb')
```

De todas las capturas que se recogen se toma una de cada cinco para reducir la cantidad de procesamiento y optimizar los recursos de la PC.

```
video.FrameGrabInterval = 5;
```

Para iniciar con la captura de videos se utiliza el comando "***start***" acompañado de la variable "***video***" entre paréntesis.

```
start(video)
```

Con el comando "***getsnapshot***" se toma una fotografía de la imagen de la cámara. La imagen se guardará en la variable "fotograma".

```
fotograma=getsnapshot(video);
```

El código completo del módulo de adquisición de imágenes quedaría de la siguiente manera:

```
vinf=imaqhwinfo;

video = videoinput(cell2mat(vinf.InstalledAdaptors), 2, 'YUY2_640x480');

set(video, 'FramesPerTrigger', Inf)

set(video, 'ReturnedColorspace', 'rgb'

video.FrameGrabInterval = 5;

start(video)

while(app.func == 1)

Fotograma=getsnapshot(video);
```

La imagen que se obtiene es la bandeja de la balanza de color blanco para controlar el ambiente en el que se trabaje con las frutas como se muestra en la Figura 2.32.

Figura 2.32. Captura de la webcam de la bandeja de la balanza en la cual se colocarán las frutas.

Adicionalmente, las tres tonalidades de luz que ofrece la lámpara circular pueden ser utilizadas para realizar una serie de pruebas y así, determinar qué tipo de luz brinda mejores resultados al momento de reconocer colores y formas (Figura 2.33.). Además de los tres tipos de luz que brinda la lámpara circular también se puede ir jugando con la intensidad de radiación de estas. Es decir, la lámpara circular ofrece un control completo del entorno en el cual se adquirirán las imágenes para el posterior procesamiento de éstas. Este proceso mejorará la calidad de las imágenes obtenidas.

Figura 2.33. Tres tipos de luz que emite la lámpara circular. De izquierda a derecha se tiene: Luz fría, luz diurna, y luz cálida.

2.3. RECONOCIMIENTO DE COLORES

A partir de la captura de la imagen se trabaja con ella como cualquier matriz ordinaria de tres dimensiones. El siguiente paso es el de adquirir cada capa de la imagen por separado. Es decir, la capa de rojos (R), verdes (G), y azules (B). A cada capa se las ha guardado en las variables “Capa_Rojo”, “Capa_Verde”,” Capa_Amarillo” de los colores rojo, verde, y azul respectivamente. Se aprovecha la misma línea de código para convertir la imagen a formato

"***double***". Dicho cambio se lo realiza para facilitar el reconocimiento de colores, ya que todos los valores negativos significan la ausencia del color, mientras que los valores positivos la presencia del color que se está diferenciando del resto.

```
Capa_Rojo=double(Fotograma(:,:,1));

Capa_Verde=double(Fotograma(:,:,2));

Capa_Amarillo=double(Fotograma(:,:,3));
```

2.3.1. RECONOCIMIENTO DE COLOR ROJO

El siguiente procedimiento se lo realiza con las tres capas de la imagen, salvo en la imagen de azul donde se agrega un procedimiento extra. Para extraer la tonalidad roja del fotograma se restan las matrices "Capa_Verde" y "Capa_Amarillo" de la matriz "Capa_Rojo". El resultado es una imagen en blanco y negro, en donde el color blanco representa los lugares en los que se detecta el color rojo y el color negro representa los lugares donde se encuentra cualquier otro color.

```
Imagen_Rojo= Capa_Rojo – Capa_Verde – Capa_Amarillo;
```

Para controlar el correcto funcionamiento del comando, se muestra la captura de la imagen junto con el reconocimiento de tonos rojos únicamente.

Figura 2.34. a) Imagen original, b) Imagen en donde se reconocen únicamente tonos de color rojo

2.3.2. RECONOCIMIENTO DE COLOR VERDE

El mismo proceso se utiliza para el color verde, con las diferencias de que ahora se restan las matrices "Capa_Rojo", y "Capa_Amarillo" de "Capa_Verde".

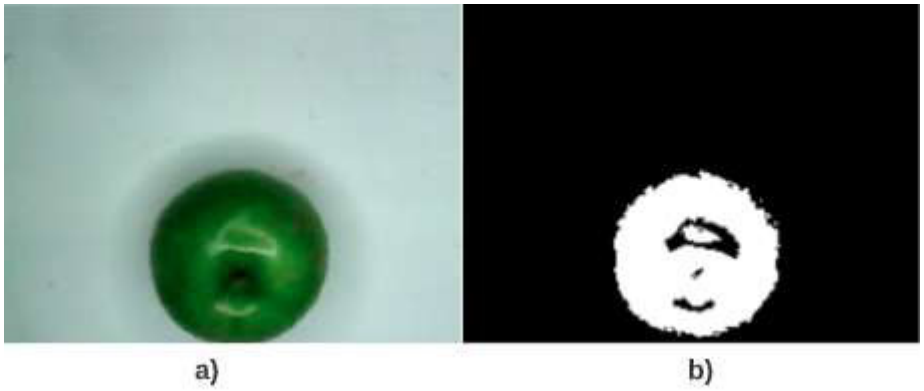


Figura 2.35. a) Imagen original b) Imagen en tonos de verde

2.3.3. RECONOCIMIENTO DEL COLOR AMARILLO

Para el caso del color amarillo se agrega unos pasos más al código. Se invierten los colores de la imagen original con el comando "***imcomplement***".

```
Fotograma_Invertido = imcomplement(Fotograma);
```

Después, se extrae cada capa en variables diferentes a las usadas con los colores rojo y verde:

```
Capa_Rojo_Invertido=double(Fotograma_Invertido(:,:,1));

Capa_Verde_Invertido=double(Fotograma_Invertido(:,:,2));

Capa_Amarillo_Invertido=double(Fotograma_Invertido(:,:,3));
```

Finalmente, para obtener el color amarillo en colores de blanco y negro se hace la resta de capas:

```
Imagen_Amarillo= Capa_Rojo_Invertido - Capa_Verde_Invertido -
Capa_Amarillo_Invertido;
```

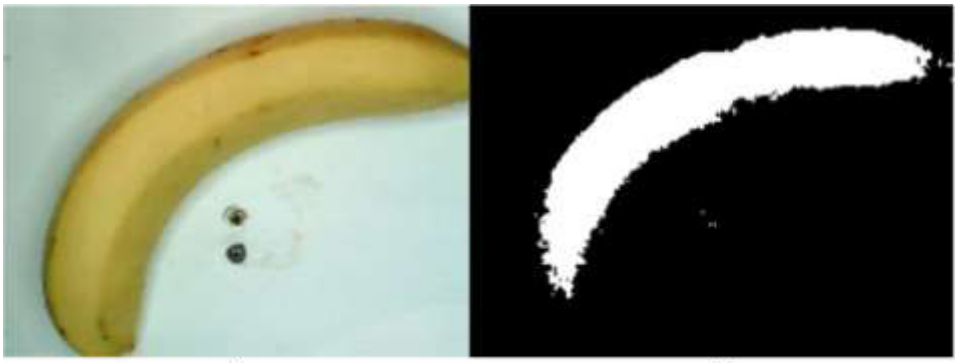


Figura 2.36. a) Imagen original b) Imagen en tonos amarillos

Es así que el código completo del reconocimiento de colores es el siguiente:

```
%% Extracción de capas

Capa_Rojo=double(Fotograma(:,:,1));

Capa_Verde=double(Fotograma(:,:,2));

Capa_Amarillo=double(Fotograma(:,:,3));

%% Tonos de rojo

Imagen_Rojo= Capa_Rojo - Capa_Verde - Capa_Amarillo;

%% Tonos de verde

Imagen_Verde= Capa_Verde - Capa_Rojo - Capa_Amarillo;

%% Tonos de amarillo

% Inversión de Fotograma

Fotograma_Invertido = imcomplement(Fotograma);

% Extracción de capas invertidas

Capa_Rojo_Invertido=double(Fotograma_Invertido(:,:,1));

Capa_Verde_Invertido=double(Fotograma_Invertido(:,:,2));

Capa_Amarillo_Invertido=double(Fotograma_Invertido(:,:,3));

% Tonos amarillos

Imagen_Amarillo= Capa_Rojo_Invertido - Capa_Verde_Invertido -
Capa_Amarillo_Invertido;
```

2.3.4. UMBRALES PARA DIFERENCIACIÓN ENTRE COLORES

Una vez que se observaron los resultados de la diferenciación de colores se descubrieron ciertos inconvenientes:

- Reconocimiento de tonos rojos en imágenes de frutas amarillas y viceversa.
- Tonos detectados como rojo o amarillo a veces sobrepasaban las dimensiones originales de la fruta.
- Estado de madurez de la fruta provoca que el reconocimiento de tonos verdes cuando la fruta está tierna y tonos amarillos cuando está muy madura.
- El brillo de las frutas debido a la intensidad de la luz provoca hoyos en los lugares en donde se refleja dicho brillo.
- Ciertas áreas de la fruta no eran detectadas.

Los resultados de las primeras pruebas, así como los inconvenientes detectados se detallan en el Anexo A.

Para tener un panorama más amplio de las posibles soluciones se graficaron los histogramas de las capas de rojo, verde, y amarillo de todas las frutas como se muestra en el ANEXO B y analizar sus resultados. A continuación, se muestran ejemplos de los histogramas obtenidos en cada capa de frutos rojos, verdes y amarillos:

Ejemplo de Histograma de un Fruto Rojo

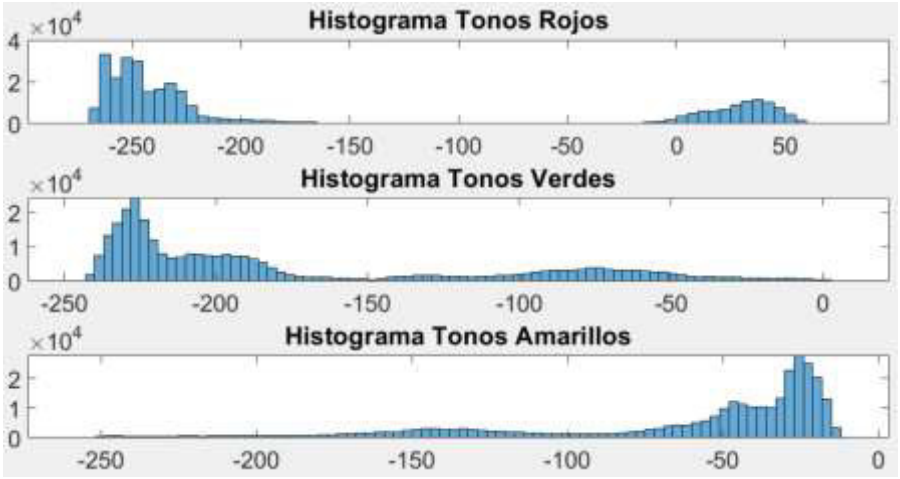


Figura 2.37. Histogramas de las tres capas de un fruto roja.

Ejemplo de Histograma de Fruto Verde

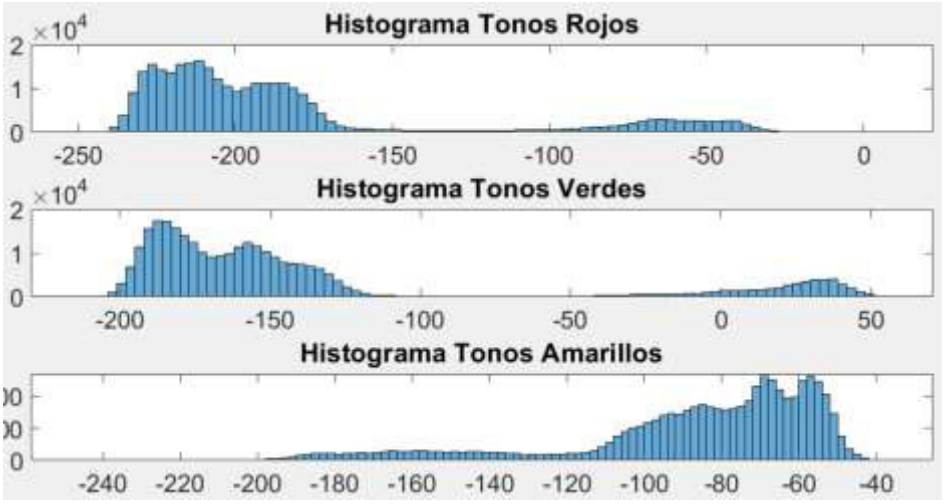


Figura 2.38. Histogramas de las tres capas de un fruto verde

Ejemplo de Histograma de Fruto Amarillo

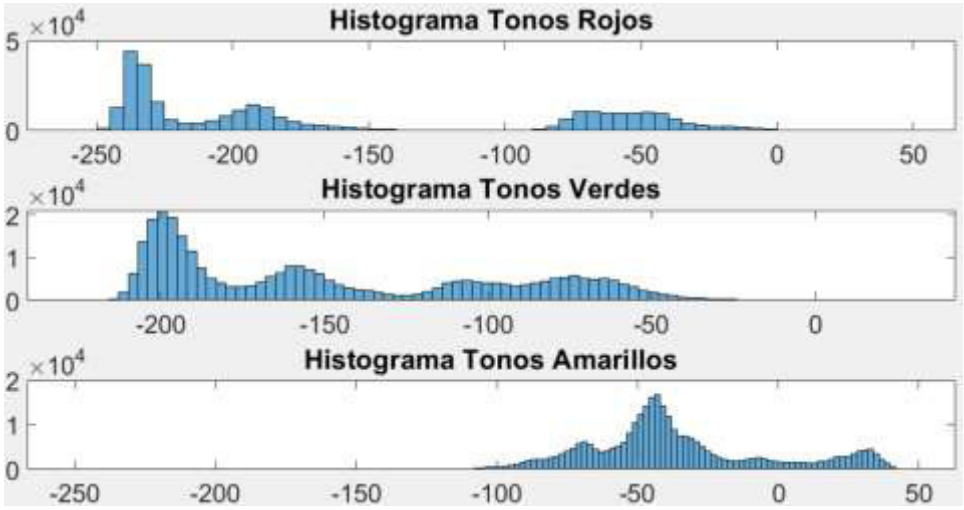


Figura 2.39. Histogramas de las tres capas de un fruto amarillo.

Para el estudio de histogramas se escribió un pequeño script en Matlab y con el código que se muestra a continuación:

```
%INICIALIZACIÓN DEL PROGRAMA
clc;
clear;
%Lectura de la imagen
I=imread('D:\Politecnica\TESIS_PASANTÍAS\TESIS\PESAJE DE
FRUTAS\scripts\BA1.jpg');
figure(1)
title('Imagen Original');
ImR=double(I(:,:,1));
```

```
ImV=double(I(:,:,2));
ImA=double(I(:,:,3));
%%%RECONOCIMIENTO DE COLORES
 %ROJO
 imagen_rojo= ImR - ImV - ImA;
 %VERDE
 imagen_verde= ImV - ImR - ImA;
 %AMARILLO
 Iim= imcomplement(I);
 ImR_im=double(Iim(:,:,1));
 ImV_im=double(Iim(:,:,2));
 ImA_im=double(Iim(:,:,3));
 imagen_amarillo= ImA_im - ImV_im - ImR_im;
%%%%% HISTOGRAMAS
%%%HISTOGRAMA ROJO
subplot(3,1,1);
histR=histogram(imagen_rojo);
title('Histograma Tonos Rojos');
xlim([-50 50])
ax = gca;
ax.FontSize = 20;
%%%HISTOGRAMA VERDE
subplot(3,1,2);
histV=histogram(imagen_verde);
title('Histograma Tonos Verdes');
xlim([-50 50])
ax = gca;
ax.FontSize = 20;
%%%HISTOGRAMA AMARILLO
subplot(3,1,3);
histA=histogram(imagen_amarillo);
title('Histograma Tonos Amarillos');
xlim([-50 50])
ax = gca;
ax.FontSize = 20;
```

De los histogramas estudiados en el ANEXO B se pueden sacar ciertas conclusiones:

- Los valores inferiores a cero corresponden a aquellos pixeles que no corresponden al color de la fruta de la cual se está encontrando su tono de color.

- Los histogramas que tienen valores que sobrepasan el valor de cero corresponden a pixeles en donde se detectó el color del tono de su capa.
- Existen ciertos valores que sobrepasan el cero y su color no es el de la fruta estudiada.
- Ciertos pixeles que pertenecen al color que se desea identificar están debajo del cero y, por lo tanto, se desechan.

Para solucionar el inconveniente de la última conclusión se optó por establecer un umbral capaz de reconocer pixeles que no están siendo reconocidos y se desechan. Los valores de los umbrales para cada color se los escogió mediante el análisis de los histogramas que se muestran en el ANEXO B. En la Tabla 2.3. se muestra un resumen de los límites y valores más significativos de cada histograma dentro de cada color de fruta.

Tabla 2.3. Límite de Pixeles

TABLA DE VALORES DE LÍMITE DE PIXELES					
ROJO		**VERDE**		**AMARILLO**	
FRUTOS ROJOS					
MÍNIMO	**CENTRAL**	**MÍNIMO**	**CENTRAL**	**MÍNIMO**	**CENTRAL**
-10	35	2		-12	
-10	38	0		-12	
-40	18	9		-18	
-20	45	3		-15	
-30	35	12		-21	
-50	10	6		-15	
-45	25	6		-21	
-15	40	2		-15	
-10	40	6		-9	
-10	45	6		-12	
-30	50	-11		-9	
-20	50	-45		-9	
-10	50	-19		-28	
0	50	-22		-10	
5	50	-15		-11	
-25	40	-24		-8	
-25	35	-8		-6	
-5	30	-15		-9	

-10	35	-5		-45	
-10	35	-15		-28	
FRUTOS VERDES					
-28		-41	38	-42	
-21		-42	38	-38	
-12		-50	38	-38	
-21		-50	32	-40	
-18		-50	38	-42	
-15		-50	40	-32	
-5		-40	40	-35	
-15		-40	35	-50	
-30		-30	40	-50	
-18		-35	40	-40	
0		-11	35	-15	
-12		-50	40	-9	
-10		-20	40	-18	
-20		-38	25	-15	
-12		-15	30	-9	
-12		-20	30	-18	
-10		-15	26	-15	
-20		-15	26	-22	
-20		-18	30	-25	
-30		-5	35	-25	
FRUTOS AMARILLOS					
-10		-15		36	-50
0		-25		42	34
10		6		50	2
-10		-22		16	8
-6		-25		28	6
-12		-18		38	-40
-12		-20		28	16
-20		-20		30	14
-10		-20		24	14
-10		-15		40	35
-15		-30		40	0

-40		-40		45	0
-30		-40		20	10
-10		-20		50	20
-10		-30		40	20
-10		-25		40	20
8		-30		40	30
10		-25		40	20
-40		-40		30	30
-40		-40		30	20

De acuerdo a la Tabla 2.3., se escogió un umbral capaz de reconocer tonos que antes se tomaban como ajenos al color original sin perjudicar el correcto reconocimiento del color verdadero de la fruta. A continuación, se muestran ejemplos de la recuperación de colores de frutas rojas, verdes, y amarillas.

Ejemplo de recuperación de colores de fruto rojo:

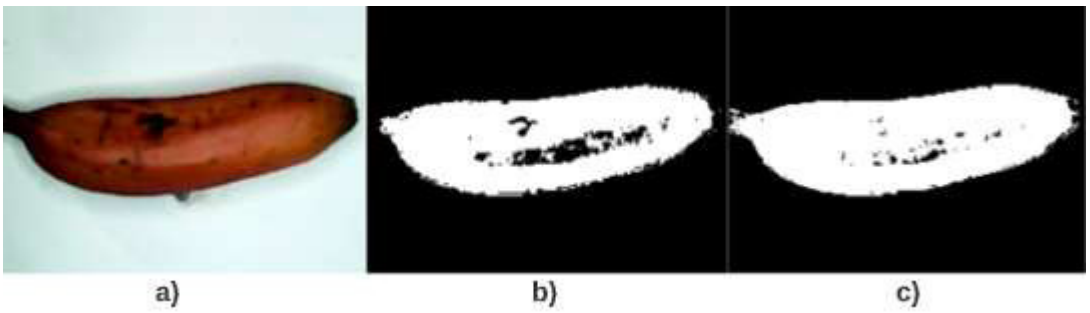


Figura 2.40. a) Imagen Original b) Reconocimiento de Tonos Rojos c) Imagen con Umbral.

Ejemplo de recuperación de colores de fruto verde:

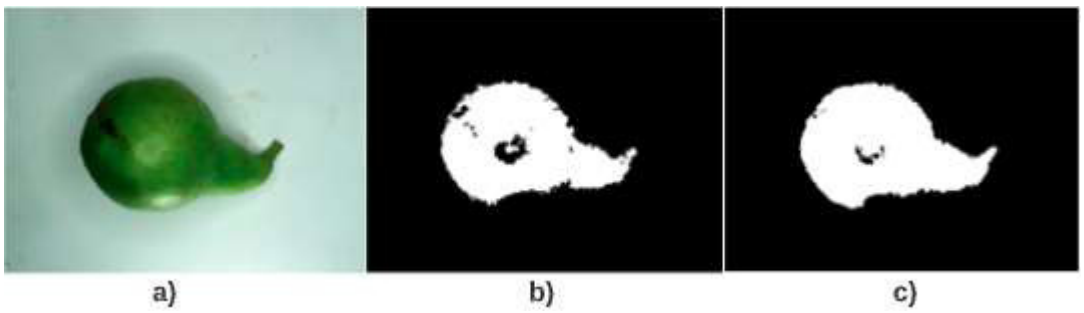


Figura 2.41. a) Imagen Original b) Reconocimiento de Tonos Verdes c) Imagen con Umbral.

Ejemplo de recuperación de colores de fruto amarillo:

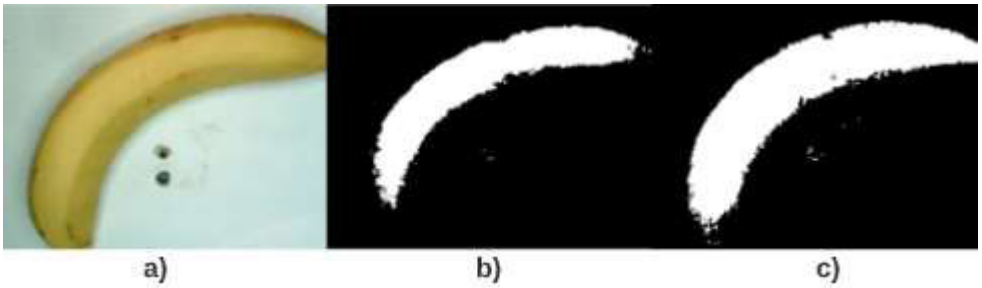


Figura 2.42. a) Imagen Original b) Reconocimiento de Tonos Amarillos c) Imagen con Umbral.

Para implementar el umbral en el código se utilizó el comando "***imbinarize***" y a continuación el valor del umbral escogido como se muestra en la siguiente línea:

```
Imagen_Rojo_Umbral=imbinarize(Imagen_Rojo, -5);

Imagen_Verde_Umbral=imbinarize(Imagen_Verde, -20);

Imagen_Amarillo_Umbral=imbinarize(Imagen_Amarillo, -10);
```

2.4. RECONOCIMIENTO DE FRUTAS

Después de analizar los histogramas y de colocar un umbral para cada capa, el formato de las matrices cambió de "***double***" a "***logical***". Esto debido a que ahora se tienen únicamente valores de 1 cuando se tiene un pixel del color de fruta esperado y 0 cuando se tiene un pixel de cualquier otro color. Este cambio de formato nos ayudará a diferenciar a un grupo de frutas basándonos únicamente en el área que cubren cada una. Es así que, si detectamos la presencia de una fruta de color rojo, las dos únicas opciones que se tienen es que sea o bien una manzana roja o en su defecto, una banana roja. En el caso de detectar la presencia de un tono amarillo, las dos únicas opciones que se tienen es que sea una pera amarilla o, una banana amarilla.

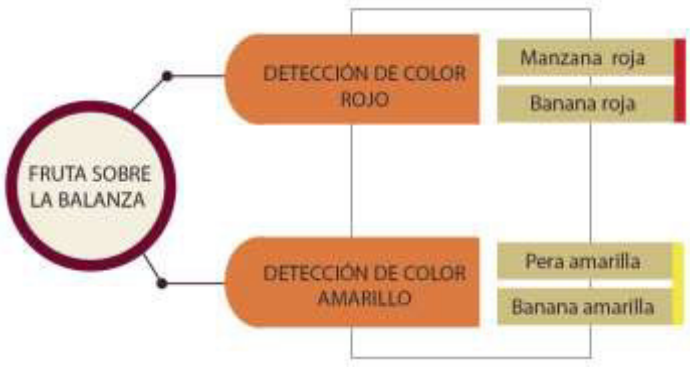


Figura 2.43. Clasificación de frutas a partir del reconocimiento de colores rojo y amarillo.

Para clasificar las frutas basta con hacer un conteo de los pixeles “1” y determinar una cantidad para clasificar a la fruta entre las opciones que se muestran en la Figura 2.43. Pero antes de realizar esta operación, es necesario resolver dos inconvenientes que aparecieron con la implementación de los umbrales en el subtema anterior.

2.4.1. CORRECCIÓN DE ÁREAS PEQUEÑAS DETECTADAS POR LOS UMBRALES

La idea principal de colocar umbrales fue la de recuperar ciertas áreas de la fruta que no se estaban reconociendo y causaban que no se tenga morfología completa de la fruta. Al momento de resolver este inconveniente surgió otro contratiempo. Esta vez, ciertas áreas de tonos diferentes al de la fruta aparecieron. Debido al estado de madurez de la fruta surgieron bordes de color verde cuando la fruta estaba tierna y manchas amarillas cuando la fruta estaba demasiado madura. También aparecieron tonos amarillos debido al reflejo de la bandeja con la luz. Las nuevas áreas detectadas se muestran a continuación:

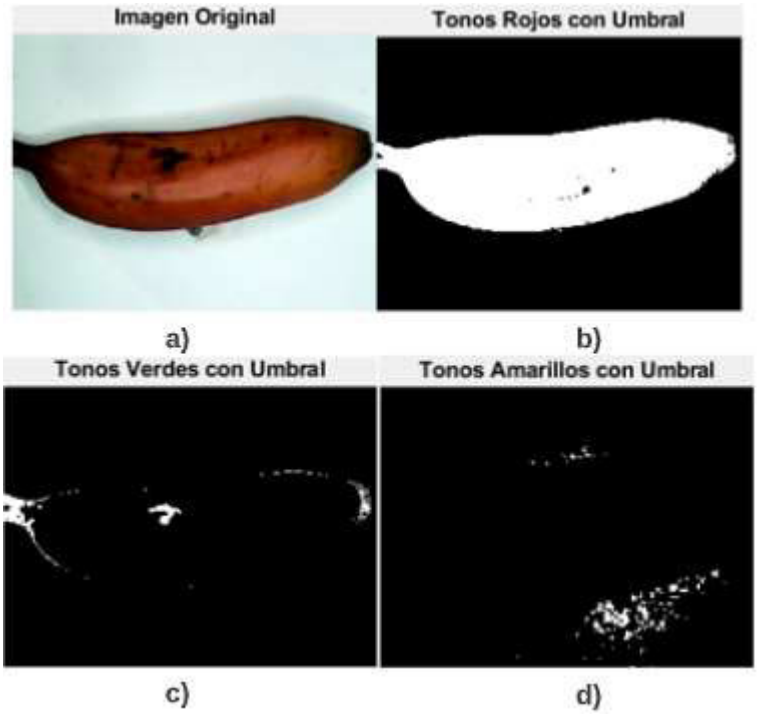
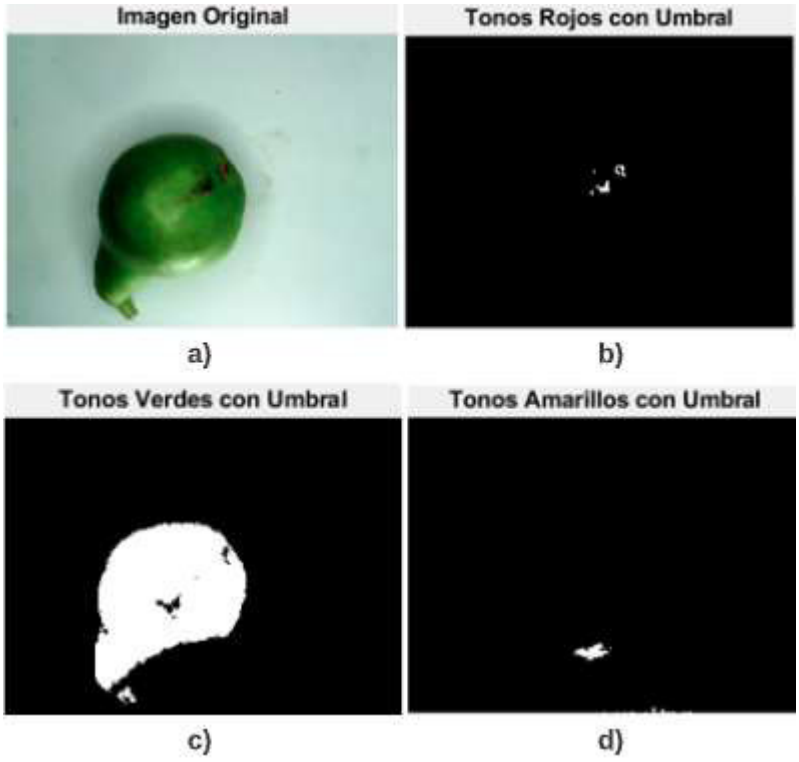
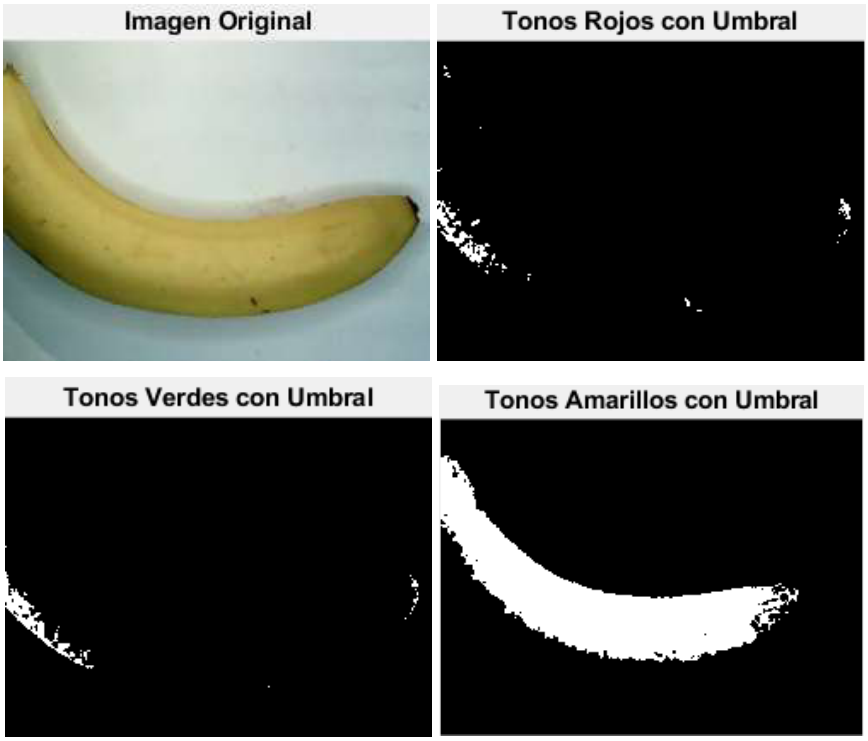


Figura 2.44. a) Imagen Original b) Imagen de Tonos Rojos con Umbral c) Imagen de Tonos Verdes con Umbral d) Imagen de Tonos Amarillos con Umbral de una fruta roja



Figuras 2.45. a) Imagen Original b) Imagen de Tonos Rojos con Umbral c) Imagen de Tonos Verdes con Umbral d) Imagen de Tonos Amarillos con Umbral de una fruta verde.



Figuras 2.46. a) Imagen Original b) Imagen de Tonos Rojos con Umbral c) Imagen de Tonos Verdes con Umbral d) Imagen de Tonos Amarillos con Umbral de una fruta amarilla.

Para resolver este contratiempo se optó por hacer un conteo del número de pixeles que eran detectados erróneamente en las capas que no correspondían a la fruta de la imagen original. Para esto se utilizaron los comandos "***numel***" y "***find***" para encontrar el valor escalar del número de pixeles detectados de un color específico en cada fruta. Se lo implementó en el código tal como se muestra a continuación:

```
Pixel_Rojo = numel(find(Imagen_Rojo_Umbral ==1));

Pixel_Amarillo = numel(find(Imagen_Amarillo_Umbral ==1));

Pixel_Verde = numel(find(Imagen_Verde_Umbral ==1));
```

Se recopilaron los valores de la cantidad de pixeles de 12 ejemplares de cada fruta obteniendo los siguientes valores:

Tabla 2.4. Conteo de pixeles de cada capa de colores.

CONTEO DE PIXELES DE CADA CAPA DE COLORES			
FRUTA	PIXELES		
	ROJO	VERDE	AMARILLO
MANZANA ROJA	49665	341	10
	47046	51	71
	38431	49	17
	49657	270	88
	51459	751	0
	39358	248	0
	40496	2615	87
	38468	366	36
	36364	0	0
	37577	1	0
	37596	8	0
	37038	4	19
MINIMO	36364	0	0
MAXIMO	51459	2615	88
BANANA ROJA	84794	3631	3471
	84845	3160	5461
	77546	7977	0

	75284	3757	209
	80193	7522	0
	63054	4589	121
	69932	6071	720
	84842	3537	2736
	85097	5546	5337
	85156	5447	4962
	83571	5338	7728
	83597	5602	8076
MINIMO	63054	3160	0
MAXIMO	85156	7977	8076
PERA VERDE	863	48173	61
	420	43815	854
	591	48332	81
	684	47187	577
	325	49300	16
	314	48131	2
	408	48650	7
	598	45637	53
	625	46404	87
	678	44699	228
	640	47679	0
	67	48631	13
MINIMO	67	43815	0
MAXIMO	863	49300	854
MANZANA VERDE	88	42321	0
	151	41402	0
	165	44133	2
	199	41029	59
	210	42509	52
	163	45604	69
	124	45219	23
	110	42417	0
	278	46671	0
	137	43794	9

	351	48755	39
	214	43745	50
MINIMO	88	41029	0
MAXIMO	351	48755	69
PERA AMARILLA	1523	660	14463
	1252	127	18706
	1155	79	17543
	1157	106	18652
	1231	90	15561
	1657	57	24263
	2072	278	23351
	7394	409	24337
	9002	248	20689
	1146	73	20112
	1930	61	19865
	1781	65	20892
MINIMO	1146	57	14463
MAXIMO	9002	660	24337
BANANA AMARILLA	1961	142	65186
	387	96	54550
	6748	11839	95631
	686	162	59233
	729	137	57949
	1010	908	52535
	1333	652	51790
	1099	655	51807
	1108	631	59361
	2596	2931	49211
	111	122	45490
	2198	1173	55097
MINIMO	111	96	45490
MAXIMO	6748	11839	95631

A partir de la Tabla 2.4. se extraen los valores máximos y mínimos para visualizar de manera clara las acciones a realizar a partir de estos datos. Es así que se tienen dichos valores en la Tabla 2.5.

Tabla 2.5. Valores mínimos y máximos de los pixeles de frutas

VALORES MINIMOS Y MAXIMOS DE LAS FRUTAS				
		ROJO	VERDE	AMARILLO
MANZANA ROJA	MINIMO	36364	0	0
	MAXIMO	51459	2615	88
BANANA ROJA	MINIMO	63054	3160	0
	MAXIMO	85156	7977	8076
PERA VERDE	MINIMO	67	43815	0
	MAXIMO	863	49300	854
MANZANA VERDE	MINIMO	88	41029	0
	MAXIMO	351	48755	69
PERA AMARILLA	MINIMO	1146	57	14463
	MAXIMO	9002	660	24337
BANANA AMARILLA	MINIMO	111	96	45490
	MAXIMO	6748	11839	95631

Con estos valores se pueden sacar las siguientes conclusiones:

- Existen pixeles que son reconocidos como un tono de un color diferente al de la fruta estudiada.
- Existen rangos de áreas bastante diferenciados para cada fruta, independientemente que sean del mismo color. Esto servirá posteriormente para la clasificación y reconocimiento de frutas a partir del área que poseen.

Para solucionar la primera conclusión se utiliza el comando "***bwareaopen***" para eliminar todas las áreas que tengan un área menor a cierto valor umbral. Los umbrales escogidos se obtuvieron de la Tabla 2.3. para cada color y se implementó en el código de la siguiente manera:

```
Imagen_Rojo_Filtrado=bwareaopen(Imagen_Rojo_Umbral, 20000);
Imagen_Verde_Filtrado=bwareaopen(Imagen_Verde_Umbral, 25000);
Imagen_Amarillo_Filtrado=bwareaopen(Imagen_Amarillo_Umbral, 11000);
```

2.4.2. RELLENO DE AGUJEROS

Uno de los problemas presentados al principio del reconocimiento de colores fueron los agujeros que se mostraban en las frutas causado por el brillo o por ciertas áreas maltratadas de la fruta. Al momento del reconocimiento de colores, saltaron agujeros como los que se muestran en la Figura 2.47.

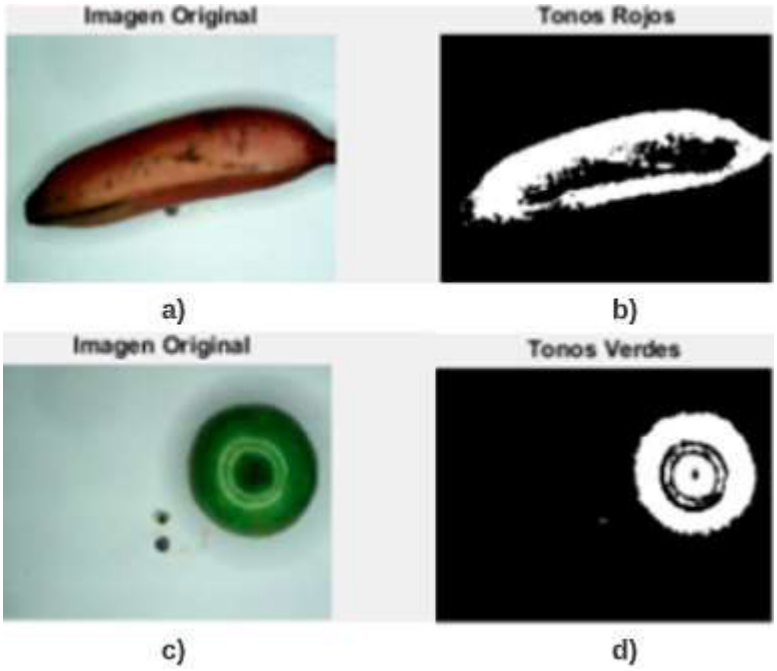


Figura 2.47. Ejemplos de la presencia de agujeros.

Para solucionar este inconveniente se utilizó el comando "***imfill***", seguido del argumento '***holes***' para rellenar estas partes que no son reconocidas por las razones mostradas anteriormente. Se implementó en el código de la siguiente manera:

```
Imagen_Rojo_Rellenado=imfill(Imagen_Rojo_Filtrado,'holes');
Imagen_Verde_Rellenado=imfill(Imagen_Verde_Filtrado,'holes');
Imagen_Amarillo_Rellenado=imfill(Imagen_Amarillo_Filtrado,'holes');
```

2.4.3. IDENTIFICACIÓN DE OBJETOS

Se identifican todos los ejemplares de la fruta presentes sobre la balanza y que son capturados por la cámara web con el comando "***bwlabel***". Se da un nombre a cada objeto identificado. Se acompaña al comando con el argumento "4" que es la manera en la que se emparejarán los pixeles. Si hay zonas cercanas con el mismo tono de color se toma a cada una de manera individual, en cambio, con "8" las zonas cercanas del mismo tono de color se las toma como uno solo. Este comando se implementó al código de la siguiente manera:

```
Sector_Rojo=bwlabel(Imagen_Rojo_Rellenado, 4);
Sector_Verde=bwlabel(Imagen_Verde_Rellenado, 4);
Sector_Amarillo=bwlabel(Imagen_Amarillo_Rellenado, 4);
```

2.4.4. CLASIFICACIÓN DE FRUTOS ROJOS Y AMARILLOS

Después de haber identificado los objetos, es posible hacer una clasificación de los frutos basándose en el área que presenta cada objeto. De acuerdo a la Tabla 2.3. Los rangos para diferenciar entre frutos del mismo color quedarían de la siguiente manera:

- Banana Roja → Áreas mayores a 50.000
- Manzana Roja → Áreas menores a 50.000
- Banana Amarilla → Áreas mayores a 35.000
- Pera Amarilla → Áreas menores a 35.000

Para la separación de acuerdo al área de cada fruta, bastaría con establecer un condicional con los valores mostrados anteriormente y se lo implementaría de la siguiente manera:

2.4.4.1. Frutos rojos

```
if Pixel_Rojo > 50000
app.EditField.Value = 'BANANA ROJA';
else
app.EditField.Value = 'MANZANA ROJA';
end
```

2.4.4.2. Frutos amarillos

```
if Pixel_Amarillo > 35000
app.EditField.Value = 'BANANA AMARILLA';
else
app.EditField.Value = 'PERA AMARILLA';
end
```

2.4.5. CLASIFICACIÓN DE FRUTOS VERDES

La clasificación de frutos verdes por medio del área que ocupa cada uno ya no es suficiente en este caso, ya que no existe una diferencia significativa entre el área de una manzana verde y una pera verde. Es por esto que, se debe elegir un algoritmo diferente para la clasificación de frutos verdes. Si bien el área no es un tema que diferencie a una especie de fruta de otra,

sí lo es su morfología. Se aprecia que la manzana es una fruta redonda y la pera generalmente es alargada. Es aquí donde se encuentra la manera de clasificación de ambas frutas a través de la relación existente entre el eje mayor y menor de las dimensiones de la fruta. La relación de eje mayor y eje menor de la pera siempre será mayor que la relación que tiene una manzana. Para implementar este algoritmo es necesario obtener el valor del eje mayor y eje menor con el comando "***regionprops***", acompañada de los argumentos '***MajorAxisLength***' y '***MinorAxisLength***'.

```
Region_Verde =
regionprops(logical(Sector_Verde),'BoundingBox','Centroid','MajorAxisLeng
th','MinorAxisLength');
```

Además de los argumentos '***MajorAxisLength***' y '***MinorAxisLength***' se extrajeron los argumentos '***BoundingBox***' y '***Centroid***' que servirán posteriormente para animar el reconocimiento de frutas.

El siguiente paso es extraer los valores de '***MajorAxisLength***' y '***MinorAxisLength***' para luego encontrar su relación de la siguiente manera:

```
Imagen_Verde_Eje_Mayor = Region_Verde.MajorAxisLength;
Imagen_Verde_Eje_Menor = Region_Verde.MinorAxisLength;
```

Después se encuentra la relación de ambos ejes dividiendo ambos valores:

```
Relacion_Verde = Imagen_Verde_Eje_Mayor / Imagen_Verde_Eje_Menor;
```

Con el valor de esta relación se crea un condicional para clasificar a la fruta, basado en un umbral para determinar si es una pera verde o una manzana verde. El valor de umbral escogido fue 1,2.

```
if Relacion_Verde > 1.2
 app.txtFruta.Value = 'PERA VERDE';
else
 app.txtFruta.Value = 'MANZANA VERDE';
end
```

2.5. DISEÑO DE LA INTERFAZ GRÁFICA DEL PROTOTIPO DE FRUTAS

Después de haber resuelto los módulos de pesaje, reconocimiento, y clasificación de frutas en los capítulos anteriores, en esta parte se cohesionarán todos los módulos para presentarlos de manera ordenada y así crear el sistema autónomo para determinar el precio de frutas basado en visión artificial. En la Figura 2.48. se observa la interfaz y el entorno final del sistema para determinar el precio de las frutas. A continuación, se detallarán todos los pasos para llegar al producto final.

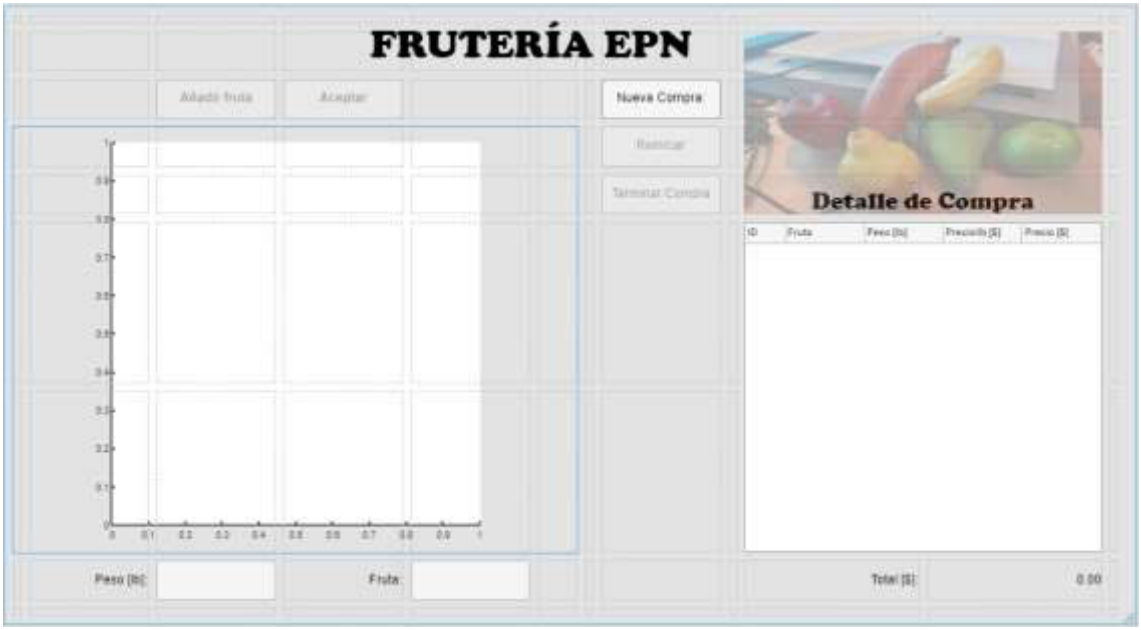


Figura 2.48. Interfaz del sistema autónomo para determinar el precio de frutas basado en visión artificial.

2.5.1. ESQUEMA DE LA INTERFAZ GRÁFICA UTILIZANDO APP DESIGNER

El diseño de la interfaz gráfica mostrada en la Figura 2.48. se armó a través de la herramienta para crear aplicaciones de Matlab llamada App Designer. El objetivo principal de esta herramienta es unificar las tres etapas en una interfaz gráfica que determine el precio de las frutas y muestre el detalle de compra al usuario de forma autónoma. En el apartado de Code Block se observa el lienzo de la interfaz gráfica expresada en lenguaje de programación. Características como largo, ancho, color, comportamiento de la herramienta, entre otros, se observan en este apartado y es aquí donde se ingresará la codificación detallada en los capítulos anteriores para la adquisición de imágenes y reconocimiento de frutas.

```
classdef app1 < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure  matlab.ui.Figure
    end
```

```
    % Component initialization
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create UIFigure and hide until all components are
created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';
            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end
    % App creation and deletion
    methods (Access = public)
        % Construct app
        function app = app1
            % Create UIFigure and components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

La codificación mostrada se genera por defecto y cada vez que se agregue una herramienta, la misma constará también dentro de este código con su nombre de objeto.

2.5.2. BOTONES UTILIZADOS EN LA INTERFAZ GRÁFICA

La estructura del funcionamiento de la interfaz gráfica funciona prácticamente por los botones con los que cuenta. Para utilizar dichos botones, se arrastra desde la barra de herramientas la opción “Button” y se lo ubica en el lugar correspondiente. Para la interfaz gráfica se

utilizarán un total de cinco botones, los mismos que controlarán el comportamiento de la interfaz, y dará inicio a las operaciones correspondientes. La manera en la que regulan los procesos se aprecia en la Figura 2.51. Los botones están agrupados en dos bloques.

En la Tabla 2.6. se observan los nombres de los botones utilizados en la interfaz gráfica, así como el nombre de objeto para configurar su comportamiento y procesos que deben realizar una vez que sean presionados.

Tabla 2.6. Botones utilizados en la interfaz gráfica del sistema autónomo para determinar el precio de las frutas.

Nombre del Botón	Nombre del Objeto
Nueva Compra	app.btnNuevaCompra
Añadir Fruta	app.btnAñadirFruta
Aceptar	app.btnAceptar
Reiniciar	app.btnReiniciar
Terminar Compra	app.btnTerminarCompra

2.5.2.1. Guía para utilización de la interfaz a través de audios.

Figura 2.49. Diseño final de la interfaz de usuario con cada uno de sus apartados.

En la Figura 2.49. se observa el diseño final de la interfaz de usuario. Debido a que el sistema para determinar el pago de frutas basado en Visión Artificial es autónomo. Se diseñó un sistema de audios para guiar al consumidor en el proceso de compra sin la necesidad de un

empleado. Al principio se observa que el único botón habilitado es el de “Nueva Compra” por lo que el comprador deberá hacer click aquí para empezar su compra guiada.

Después de hacer click en “Nueva Compra” se reproduce un audio que le da la bienvenida al usuario e indica los pasos que debe seguir el usuario para adquirir una fruta y dice lo siguiente: “Bienvenido a Frutería EPN. Para agregar frutas a su compra, haga click en ‘Añadir Fruta’. Cuando esté de acuerdo con las frutas que desear adquirir, haga click en ‘Aceptar’”.

Luego de dar click en “Añadir Fruta” se reproduce un audio indicando lo que debe hacer el usuario para comprar y dónde encontrará la información de la misma: “Coloque las frutas sobre la bandeja de la balanza. La información del peso y el tipo de fruta que desea comprar, se encuentra en la parte inferior de su pantalla. Cuando esté de acuerdo con la información, haga click en ‘Aceptar’”

Cuando usuario esté de acuerdo con la información de las frutas dará click en “Aceptar”. Se escucha un audio que confirma haber agregado dicha fruta a la lista de compras y cuáles son los pasos a seguir según la intención del usuario. El audio dice lo siguiente: “El detalle de compra de la fruta que ha adquirido se encuentra en la tabla de la parte derecha de su pantalla. Si desea terminar la compra, haga click en “Terminar Compra”. Si desea añadir más frutas a su compra, haga click en “Añadir Fruta”. Si desea cancelar la compra, haga click en “Reiniciar” y posteriormente en “Terminar Compra.”

Si la decisión del cliente es de cancelar la compra, sabe que debe hacer click en “Reiniciar” para posteriormente dar click en “Terminar Compra”. Si desea empezar nuevamente su compra, el usuario deberá hacer click en “Añadir Fruta”. El audio que se reproduce dice esto “Si desea cancelar su compra, haga click en ‘Terminar Compra’. Si desea empezar nuevamente su compra, haga click en ‘Añadir Fruta’”.

Si el usuario desea finalizar su compra y saber el precio que debe pagar dará click en “Terminar Fruta”. Se reproduce un audio que le indica dónde se encuentra el valor que debe pagar y agradece al cliente por su compra. El audio dice lo siguiente “El precio que debe pagar por su compra se encuentra impreso en la parte inferior derecha de su pantalla. ¡Gracias por preferirnos! Para iniciar su compra, haga click en ‘Nueva Compra’”.

A continuación, se detallan los procesos que se ejecutan al presionar cada botón:

2.5.2.2. Botón “Nueva Compra”

El botón “Nueva Compra” es el encargado de iniciar todo el sistema. El programa se encuentra en espera y no realiza ninguna acción hasta que este botón no sea presionado. La configuración respectiva se encuentra a continuación:

```
  % Button pushed function: btnNuevaCompra
        function btnNuevaCompraButtonPushed(app, event)
            app.id = 1;
            app.compra = {};
            app.tbDetalles.Data = app.compra;
            app.lblTotal.Text = "0.00";
            app.btnAnadirFruta.Enable = 1;
            app.btnReiniciar.Enable = 1;
            app.btnTerminarCompra.Enable = 1;
            app.btnNuevaCompra.Enable = 0;
        end
```

El código mostrado encima muestra todos los procesos que se inician al momento de presionar el botón. Entre los procesos que destacan están la habilitación de los botones "Añadir Fruta", "Reiniciar", y "Terminar Compra" con el argumento 'Enable = 1'. El botón "Nueva Compra" se deshabilita después de haber sido presionado con el argumento 'Enable = 0' y permanece en esta condición hasta que otro proceso lo vuelva a habilitar.

2.5.2.3. Botón "Añadir Fruta"

El botón "Añadir Fruta" hace correr el programa que se estructuró en los capítulos anteriores tales como Adquisición de Imágenes y Reconocimiento de frutas. Dentro del segmento de código que involucra este botón se agregará el código de los Módulos 2 y 3. Además, se habilita el botón "Aceptar" que cuando es presionado almacena el peso de la fruta reconocida para posteriormente determinar el precio de la compra. Además, se deshabilita el mismo botón hasta que vuelva a estar disponible por acción de otros procesos.

```
% Button pushed function: btnAnadirFruta
  function btnAnadirFrutaPushed(app, event)
  app.btnAnadirFruta.Enable = 0; % desactivo el boton de iniciar
para que no se presione de nuevo
  app.btnAceptar.Enable = 1; % se activa el boton de Aceptar
%%% CODIFICACIÓN DE LOS MÓDULOS DE ADQUISICION DE IMÁGENES %%%
   end
```

2.5.2.4. Botón "Aceptar"

Este botón se utiliza cuando el comprador está de acuerdo con el reconocimiento de la fruta que desea comprar y el peso que marca la balanza. Una vez presionada, la lista que se encuentra a la derecha de la interfaz gráfica indicará al usuario el detalle de su compra.

También se habilita el botón “Añadir Fruta” en caso de querer comprar otro tipo de frutas dentro del catálogo y se deshabilita el botón “Aceptar” para que no se vuelva a presionar.

```
function btnAceptarPushed(app, event)
   app.btnAnadirFruta.Enable = 1; % habilida nueva captura de video
   app.btnAceptar.Enable = 0; % deshabilita el boton de Aceptar
end
```

2.5.2.5. Botón “Terminar Compra”

La principal función que ejecuta este botón es el de devolverle al usuario el precio que debe pagar por su compra. El valor como tal se lo devuelve a través de una herramienta llamada “***app.lblTotal.Text***” que hace la suma de todas las compras que se han hecho previamente a presionar este botón.

También habilita el botón “Nueva Compra” para empezar de nuevo con el proceso de compra. A la vez, se deshabilita el resto de botones esperando a iniciar de nuevo la venta de frutas.

```
% Button pushed function: btnTerminarCompra
function btnTerminarCompraButtonPushed(app, event)
   app.btnNuevaCompra.Enable = 1;
   app.btnReiniciar.Enable = 0;
   app.btnAnadirFruta.Enable = 0;
   app.btnTerminarCompra.Enable = 0;
   if app.id > 1
      app.lblTotal.Text = num2str(sum(cell2mat(app.compra(:,5))));
   end
end
```

2.5.2.6. Botón “Reiniciar”

Este botón sirve para el caso en el que, una vez iniciada la compra, el usuario decide cancelarla. También sirve para el caso en el que se decida eliminar la compra de determinada fruta. Al momento de presionar el botón “Reiniciar”, el programa vuelve a condiciones iniciales.

```
% Button pushed function: btnReiniciar
        function btnReiniciarButtonPushed(app, event)
            app.id = 1;
            app.compra = {};
```

```
            app.tbDetalles.Data = app.compra;
        end
```

2.5.2.7. Botón “Eliminar Última Compra”

Se agregó un nuevo botón llamado “Eliminar último” que le permite al comprador borrar únicamente la última compra que se encuentre dentro del listado del “detalle de compra” en vez de eliminarlas todas con el botón “Reiniciar”.

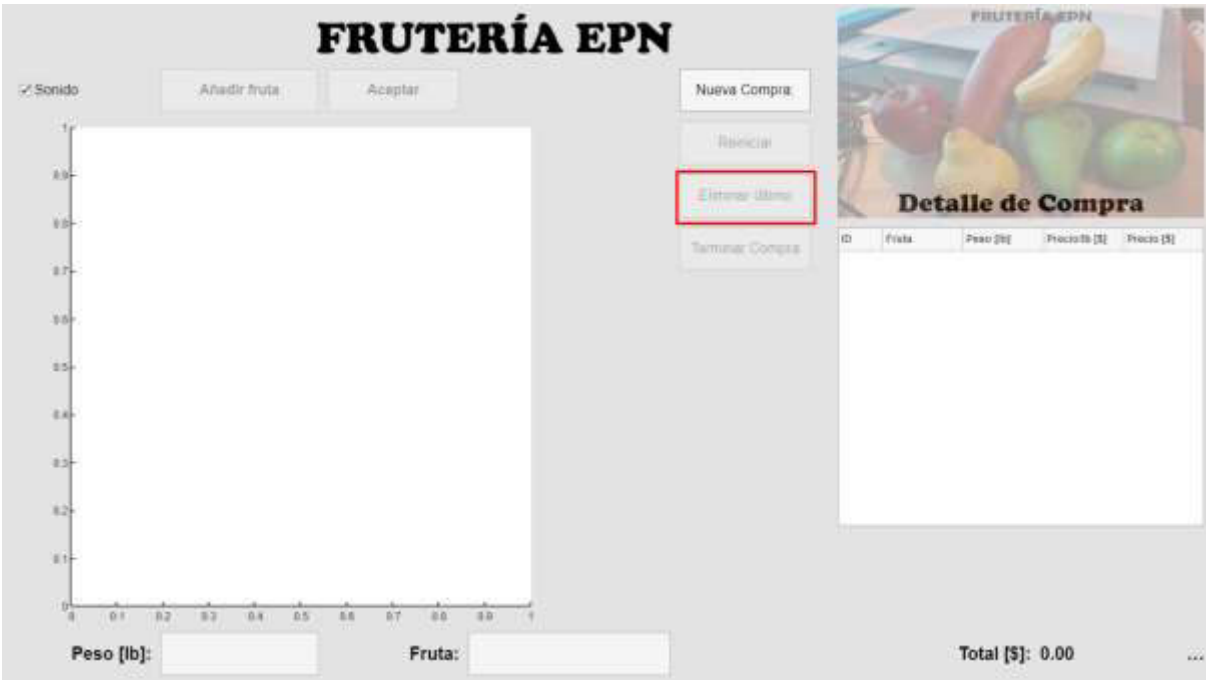


Figura 2.50. Botón “Eliminar último” que le permite al usuario borrar la última compra que se encuentre dentro del listado de “Detalle de compra”

2.5.2.8. Check Box “Sonido”

Cuando el cliente ya sabe cómo funciona el sistema, el hecho volver a escuchar los audios se vuelve incómodo. Es por esto que se agregó una herramienta “***Check Box***” con el nombre de ‘Sonido’ para activar o desactivar las audioguías cuando el usuario lo desee. La herramienta se encuentra en la parte superior izquierda de la interfaz gráfica y basta en dar click sobre el visto para desactivar el audio.

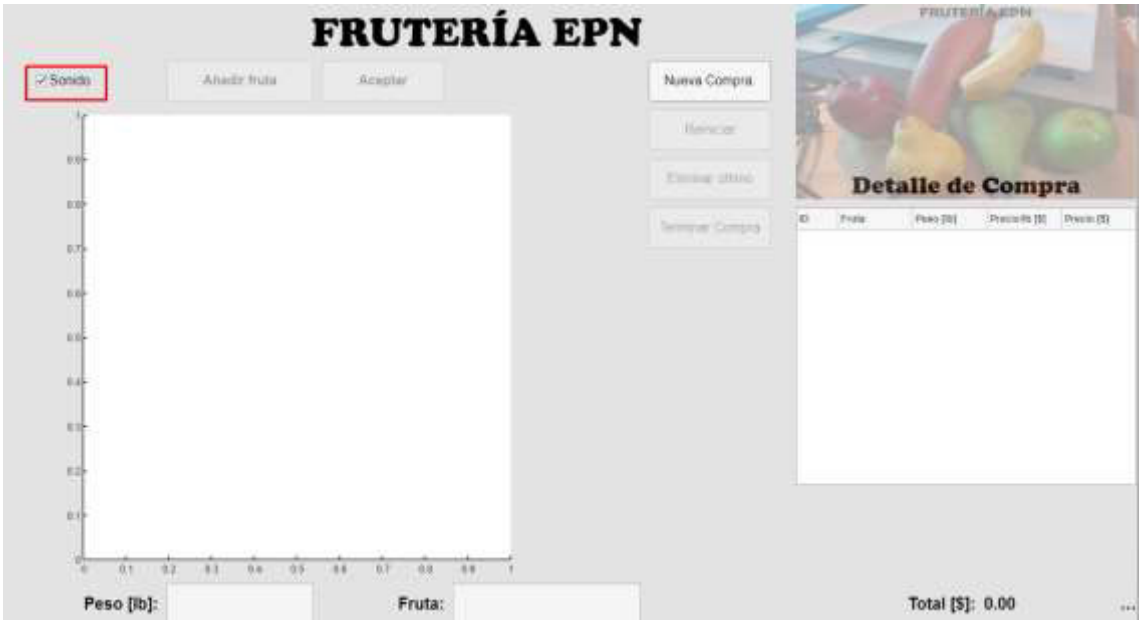


Figura 2.51. Implementación de un "***Check Box***" para activar o desactivar las audioguías del sistema autónomo para determinar el precio de frutas basado en Visión Artificial.

2.5.2. MOSTRAR LA CAPTURA DE IMÁGENES DE LA CÁMARA WEB A TRAVÉS DE UIAXES

La manera más sencilla de mostrar gráficos en App Designer es a través de la creación de un objeto llamado "***UIAxes***". Para insertarlo se arrastra este objeto desde la barra de herramientas hasta el lienzo en blanco de la interfaz gráfica.

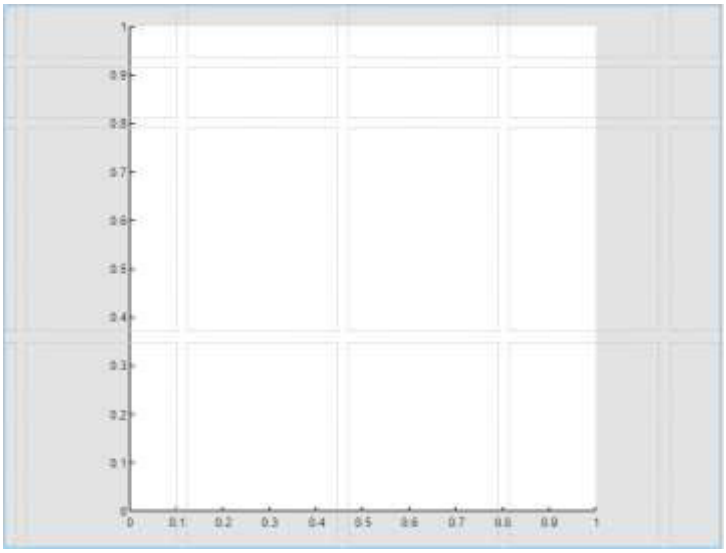


Figura 2.52. Objeto para mostrar la captura de imágenes tomadas por la webcam llamada "app.axes"

Para hacer que la imagen captada por la cámara web se muestre en el objeto se utiliza el comando "***imshow***" y entre paréntesis el nombre de la variable que se desea mostrar (Recordemos que "Fotograma es el nombre de la captura de imágenes"). El segundo argumento sirve para otorgar el permiso de mostrar la captura de imágenes las veces que sean necesarias y el tercer argumento es el nombre del objeto para mostrar imágenes, llamado "***app.axes***".

```
imshow(Fotograma,'Parent',app.axes);
```

2.5.2.1. Animación de colores

Lo primero será darle una animación de colores a la adquisición de imágenes y presentarlas en la interfaz gráfica de manera dinámica. Para esto se extraen ciertas propiedades de los objetos que ya fueron etiquetados en las variables "Sector_Rojo", "Sector_Verde", y "Sector_Amarillo" con el comando "regionprops". Las propiedades que se extrajeron son 'BoundingBox' y 'Centroid' que se detallaron más a profundidad en el Marco Teórico.

```
Region_Rojo = regionprops(logical(Sector_Rojo),
'BoundingBox','Centroid');


Region_Verde = regionprops(logical(Sector_Verde),
'BoundingBox','Centroid');


Region_Amarillo = regionprops(logical(Sector_Amarillo),
'BoundingBox','Centroid');
```

Lo siguiente es crear bucles basados en el reconocimiento de objetos que pueden ser rojos, verdes, o amarillos y si existe al menos uno de ellos en la clasificación de frutas. Si no existen objetos de determinado color no entran al bucle. La cantidad de objetos detectados de cada color se almacenaron en las variables "Region_Rojo", "Region_Verde", y "Region_Amarillo" las cuales se utilizaron en el capítulo de clasificación de frutas.

A continuación, se muestra el desarrollo del bucle para el color rojo. El procedimiento será el mismo para los otros dos colores. Para empezar el bucle se utiliza el comando "***for***", en donde el lazo empieza desde 1 hasta la cantidad de objetos que determinado color con el comando "***length***".

```
for Objecto_Rojo=1:length(Region_Rojo)
```

Se obtienen las coordenadas de las zonas en las cuales se encuentran los objetos encontrados y se los almacena en la variable “Caja_Rojo”

```
Caja_Rojo = Region_Rojo(Objecto_Rojo).BoundingBox;
```

Se dibuja un rectángulo en las coordenadas de las zonas en rojo. Con el comando ***rectangle***. La posición son las coordenadas de “Caja_Rojo”. El color del borde del rectángulo es rojo. Tiene un grosor de 2 y se lo muestra en el Axes creado en la interfaz gráfica.

```
rectangle("Position",Caja_Rojo,"EdgeColor",'r',"LineWidth",
2,'Parent',app.axes);
```

Se finaliza el bucle con “***end***”.

```
end
```

El código completo quedaría de la siguiente manera:

```
for Objecto_Rojo=1:length(Region_Rojo)
Caja_Rojo =
Region_Rojo(Objecto_Rojo).BoundingBox;rectangle("Position",Caja_Rojo,"Edg
eColor",'r',"LineWidth",2,'Parent',app.axes);
end
```

Para los colores verde y amarillo se tienen los siguientes códigos:

Color Verde

```
for Objecto_Verde=1:length(Region_Verde)
Caja_Verde = Region_Verde(Objecto_Verde).BoundingBox
rectangle("Position",Caja_Verde,"EdgeColor",'g',"LineWidth",2,'Parent',ap
p.axes);
end
```

Color Amarillo

```
for Objecto_Amarillo=1:length(Region_Amarillo)
```

```
Caja_Amarillo = Region_Amarillo(Objecto_Amarillo).BoundingBox;
rectangle("Position",Caja_Amarillo,"EdgeColor",'y',"LineWidth",2,'Parent'
,app.axes);
end
```

2.5.2.2. Adquisición de imágenes en tiempo real

Una de las características más importantes del sistema autónomo para determinar el precio de frutas basado en visión artificial es que realiza dicha tarea en tiempo real. Para lograr este objetivo, se crea una variable global que determina si el bucle donde se captura el fotograma del vídeo sigue en funcionamiento. Es global para que pueda ser editada por el botón "Añadir compra" para iniciar el bucle y por el botón "Aceptar" para finalizarla. A continuación, se muestran todas las variables globales que se crearon para la interfaz, así como su objetivo.

Tabla 2.7. Variables globales creadas y su objetivo

Nombre de la Variable global	Objetivo
func	Crear el bucle para la captura de imágenes.
id	Guardar el número de objeto a comprar.
compra	Es un arreglo de las compras existentes.
precio	Guarda el precio individual de la fruta detectada.

Estas variables globales se encuentran dentro del código de la siguiente manera:

```
properties (Access = public)
        func % variable global que determina si el bucle donde
        % se captura video sigue en funcionamiento, es global para
        % que pueda ser editada por ambos botones
        id %guarda el numero de objeto a comprar
        compra %es un arreglo de las compras existentes;
        precio %guarda el precio individual de la fruta detectada
    end
```

La variable global "func" se la utiliza dentro de la configuración del botón "Añadir Fruta" y el bucle arranca una vez que este botón ha sido presionado. Además, se deshabilita el botón "Añadir Fruta" para no vuelva a ser presionada y se habilita el botón "Aceptar" para ir agregando las frutas que se desea adquirir en el detalle de compra.

```
%% ADQUISICION DE IMÁGENES%%
```

```
app.func = 1; % hace que funcione la cámara en el bucle de abajo
app.btnAnadirFruta.Enable = 0; % desactivo el botón de iniciar para que
no se presione de nuevo
app.btnAceptar.Enable = 1; % activo el botón de terminar
```

Después de la inicialización del bucle, se crea un segmento de código para detectar errores en cuanto a la adquisición de imágenes. Si se presentan errores en la inicialización de la cámara, el programa devolverá una alerta, especificando el lugar donde se produjo el error. Para esto, se utilizan los comandos "***try***" y "***catch***" en conjunto. Este segmento de código se agrega al código de inicialización de video que se detalló en el capítulo de adquisición de imágenes y el resultado se lo muestra a continuación:

```
%% Inicializacion de video
vinf = imaqhwinfo; % adquiero la informacion de la camara disponible
camara = 0;%me indica si la camara se inicio correctamente
try
% Creo el objeto videoinput con la camara disponible, y recoge
% datos en formato rgb de 24 bits por pixel, dimension 640x480
if (exist('video') == 0)
video = videoinput(cell2mat(vinf.InstalledAdaptors), 2, 'YUY2_640x480');
end
% Se guarda la cantidad de frames que se necesite desde que se de
% el Iniciar hasta que se finaliza, por eso es inf
set(video, 'FramesPerTrigger', Inf)
set(video, 'ReturnedColorspace', 'rgb')% la imagen retornada sera en rgb
% se guarda un frame despues de cada 5 frames captados en camara
video.FrameGrabInterval = 5;
start(video) % se inicia la captura de video
   camara = 1;
catch ME
   app.func = 0;
   msgbox(ME.message);
end
while(app.func == 1) % funciona mientras no aplaste el boton aceptar
```

Después de la codificación de adquisición de imágenes y reconocimiento de frutas, el programa hace una pausa de 200 milisegundos. Después de la pausa, se paraliza la captura de fotogramas y se borra la imagen adquirida para procesar la siguiente. Se limpia la captura

que se está mostrando en el Axes y se realizan procesos adicionales para sumar la fruta adquirida al detalle de compra, mostrado en la parte derecha de la interfaz gráfica.

```
pause(0.2) % se da una pausa de 200 ms
end
if camara == 1
   stop(video); % para la captura de video
   delete(video);% elimina el objeto de captura de video
   cla(app.axes);
   app.compra(app.id, :) = {num2str(app.id), app.txtFruta.Value,
app.txtPeso.Value, (app.precio),
(app.precio*str2double(app.txtPeso.Value))};
   app.tbDetalles.Data = app.compra;
   app.id = app.id + 1;
   app.txtFruta.Value = "";
   app.txtPeso.Value = "";
end
```

2.5.3. ADQUISICIÓN DEL VALOR DEL PESO DE LA FRUTA EN MATLAB

El valor del peso que recoge el Arduino debe ser adquirido por el código implementado en Matlab. Para esto, en primer lugar, se debe instalar el Toolbox que haga posible la comunicación serial entre ambos programas.

2.5.3.1. Instalación de Matlab Support Package for Arduino Hardware

Para la instalación de este Toolbox se debe abrir la página de “Add-ons” de Matlab y buscar el nombre de la herramienta que deseamos descargar. En la Figura 2.53. se observa el Toolbox que se necesita instalar y su nombre es “MATLAB Support Package for Arduino Hardware”.

Figura 2.53. MATLAB Support Package for Arduino Hardware.

Damos click en Install y aparecerá una ventana emergente en donde solicita un permiso para la instalación del driver USB de Arduino como se muestra en la Figura 2.54.

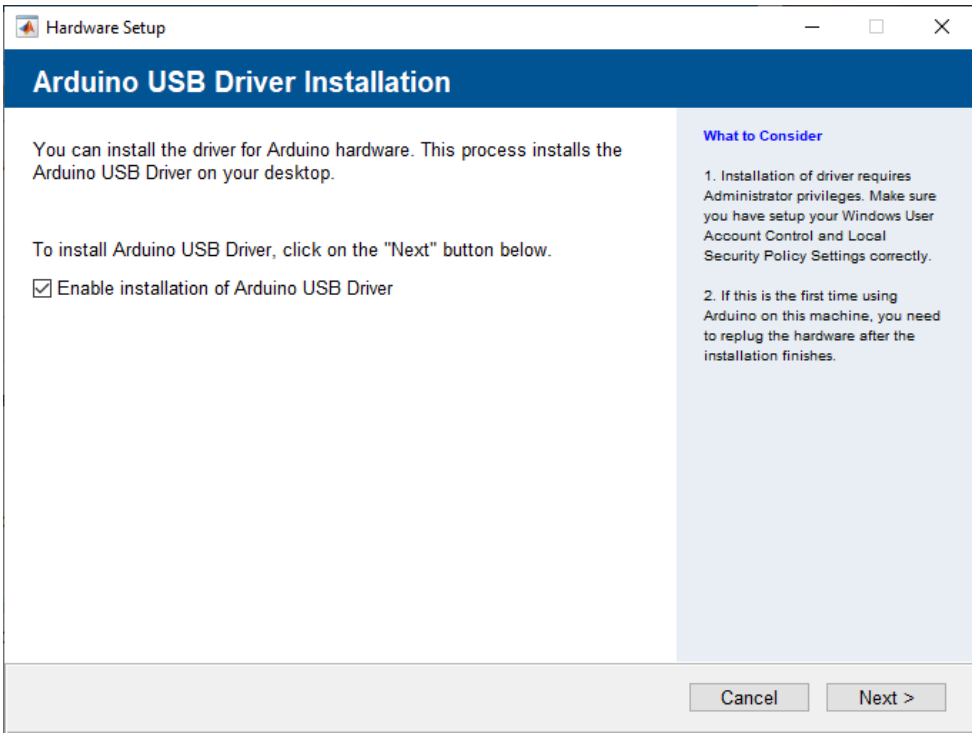


Figura 2.54. Ventana emergente para solicitud de la instalación del driver.

Después, se deberá elegir el modo de conexión en el cual se muestran tres opciones. Para este caso, el modo de conexión será vía USB.

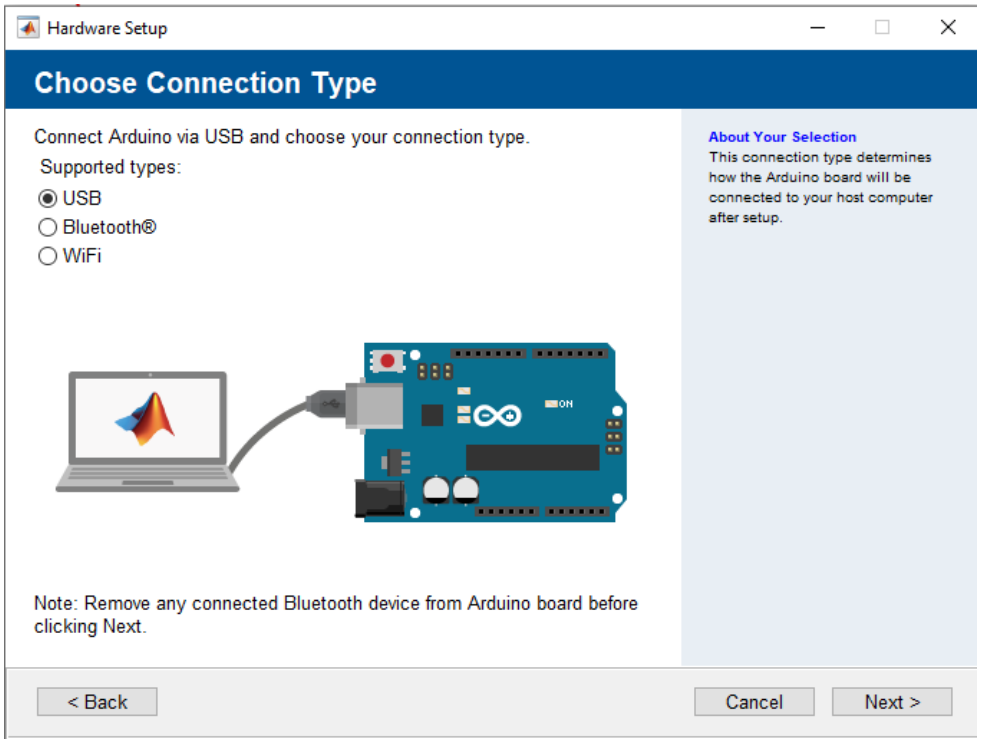


Figura 2.55. Ventana emergente para escoger el modo de conexión entre Matlab y el dispositivo Arduino UNO.

Posteriormente, en el proceso de instalación se muestran las diferentes librerías que posee el toolbox de acuerdo al modelo del dispositivo Arduino. Para este caso es el modelo UNO. También pregunta por el puerto serial que se utilizará para la comunicación y cuáles librerías desea instalar. Para esta instalación, únicamente se descargarán las seleccionadas por defecto. Para determinar el puerto de comunicación, es necesario abrir el código desarrollado en el IDE de Arduino y visualizar el número del puerto serial.

Figura 2.56. Puerto serial de comunicación de Arduino UNO.

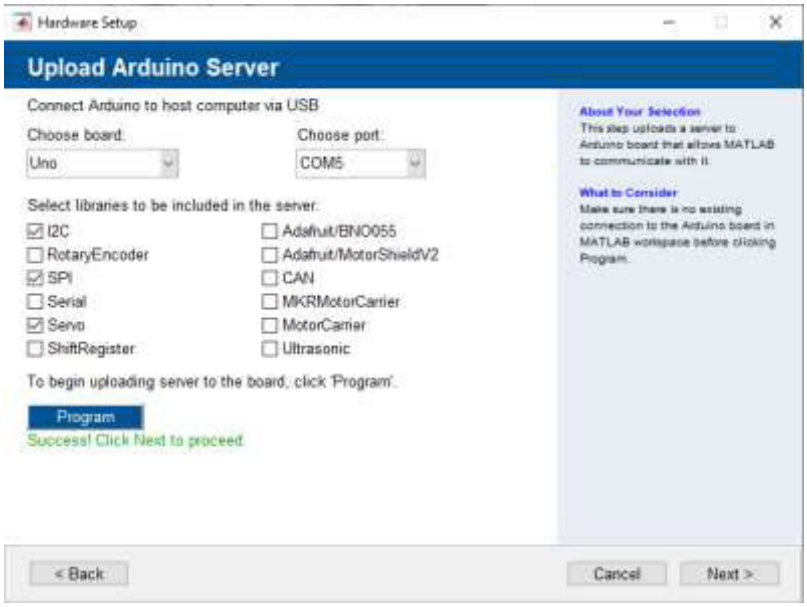


Figura 2.57. Ventana Emergente para escoger las librerías que se desee descargar.

Finalmente se realiza una prueba de todas las opciones que se han escogido para la instalación del toolbox y si todo está correcto, al final muestra un mensaje de descarga exitosa.

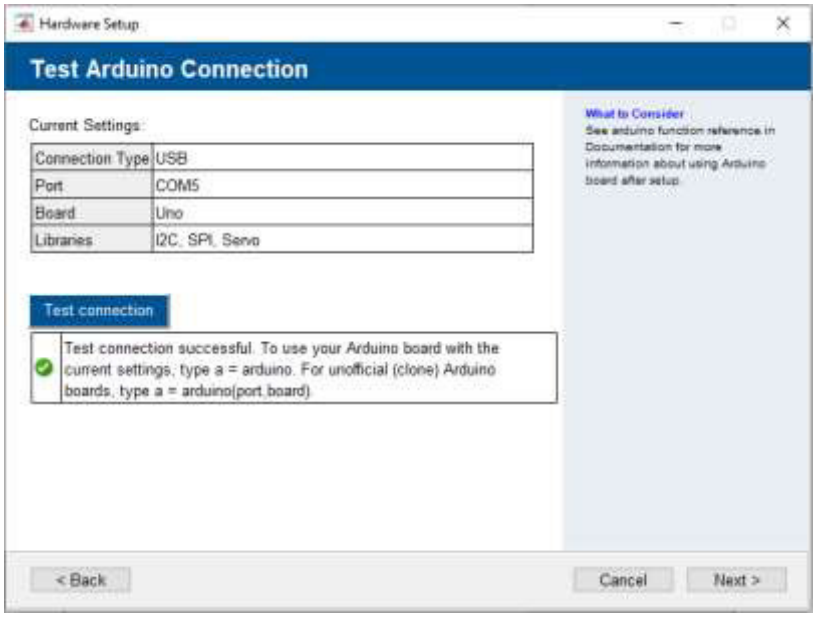


Figura 2.58. Ventana emergente de la prueba de conexión de Arduino con Matlab.

2.5.3.2. Comunicación Serial entre el dispositivo Arduino y Matlab

Para obtener el valor del peso que adquiere el dispositivo Arduino por el programa desarrollado en App Designer se crea una variable privada denominada "s" tal y como se muestra a continuación:

```
properties (Access = private)
        s
    end
```

Después se crea una función llamada "***startupFcn(app)***" en la cual se llama a la variable privada escrita anteriormente y que se la nombró "s". Para configurar los parámetros de la señal que se recibe del dispositivo Arduino se utiliza el comando "***serialport***". El primer argumento que se debe configurar es el puerto de comunicación, que en este caso es el puerto 5 (COM5). El segundo argumento es la velocidad de transmisión de datos, la cual es de 9600 baudios. Finalmente, se configura el tiempo para completar la operación de transmisión de datos. Además, se agregaron los comandos "***try***" y "***catch***" para la detección de errores en la adquisición de la señal del dispositivo Arduino.

```
% Callbacks that handle component events
    methods (Access = private)
        % Code that executes after component creation
        function startupFcn(app)
            try
```

```
            app.s = serialport("COM5",9600,"Timeout",5);
        catch ME
            msgbox(ME.message, ME.identifier);
        end
```

2.5.3.3. Precio de las frutas

El precio que se desea cobrar por las frutas no se puede determinar a ciencia cierta debido a que fluctúa y su valor cambia constantemente, sin embargo, se ha tomado como referencia el valor que se encuentran en supermercados como Supermaxi y de la página internacional que comunica los precios a turistas que piensan visitar Ecuador “preciosmundi” [31]. En la Tabla 2.8 se observan los precios de cada fruta por libra.

Tabla 2.8. Lista de precios por libra de cada fruta

Fruta	**Precio [$/lb]**
Banana Roja	0.50
Manzana Roja	0.80
Banana Amarilla	0.35
Pera Amarilla	1.16
Pera Verde	1.00
Manzana Verde	1.16

2.5.4. ETIQUETAS USADAS (LABEL Y EDIT FILE TEXT)

Las dos herramientas sirven para mostrar texto o etiquetas de datos recogidos y que se quieren mostrar en la interfaz. En el programa se utilizan varias veces ambas herramientas y se las detalla en la Tabla 2.9.

Tabla 2.9. Listado de herramientas Label y Edit File Text utilizados en la interfaz gráfica

Texto	Nombre de Objeto	Tipo de Herramienta
FRUTERÍA EPN	app.FRUTERAEPNLabel	Label
Detalle de Pago	app.lblDetalle	
Peso [lb]	app.lblPeso	
Fruta	app.lblFruta	
Total [$]	app.lblTotal	
[Muestra del valor del Peso]	app.txtPeso	Edit File Text
[Nombre de la fruta reconocida]	app.txt.Fruta	

[Precio a pagar]	app.lblTotal	

Las herramientas del tipo Edit File Text servirán para mostrar un dato al usuario. La herramienta ***app.txtPeso*** mostrará el valor que fue recogido desde el dispositivo Arduino y se lo llama de la siguiente manera:

```
   app.txtPeso.Value = "";
```

La herramienta ***app.txtFruta*** mostrará el tipo de fruta que ha sido reconocido por el módulo de clasificación de frutas y se escribe de la siguiente manera para cada caso:

Frutas Rojas

```
if Pixel_Rojo > 50000
   app.txtFruta.Value = 'BANANA ROJA';
else
   app.txtFruta.Value = 'MANZANA ROJA';
end
```

Frutas Verdes

```
if Relacion_Verde > 1.2
   app.txtFruta.Value = 'PERA VERDE';
else
   app.txtFruta.Value = 'MANZANA VERDE';
end
```

Frutas Amarillas

```
if Pixel_Amarillo > 20000
   app.txtFruta.Value = 'BANANA AMARILLA';
else
   app.txtFruta.Value = 'PERA AMARILLA';
end
```

Finalmente, la herramienta ***app.txtTotal*** mostrará el valor que debe pagar el usuario por la compra de las frutas.

```
  app.lblTotal.Text = num2str(sum(cell2mat(app.compra(:,5))));
```

Todas las herramientas que se utilizaron en la interfaz gráfica se encuentran enlistadas en la lista de componentes al principio del código. La lista de componentes del programa son los siguientes:

```
% Properties that correspond to app components
    properties (Access = public)
        UIFigure            matlab.ui.Figure
        GridLayout          matlab.ui.container.GridLayout
        btnAnadirFruta      matlab.ui.control.Button
        btnAceptar          matlab.ui.control.Button
        txtFruta            matlab.ui.control.EditField
        Label               matlab.ui.control.Label
        txtPeso             matlab.ui.control.EditField
        lblPeso             matlab.ui.control.Label
        lblFruta            matlab.ui.control.Label
        btnNuevaCompra      matlab.ui.control.Button
        btnTerminarCompra   matlab.ui.control.Button
        btnReiniciar        matlab.ui.control.Button
        tbDetalles          matlab.ui.control.Table
        lblDetalle          matlab.ui.control.Label
        TotalLabel          matlab.ui.control.Label
        lblTotal            matlab.ui.control.Label
        FRUTERAEPNLabel     matlab.ui.control.Label
        Image               matlab.ui.control.Image
        axes                matlab.ui.control.UIAxes
    end
```

Con estas líneas de código finalmente queda estructurado el diseño de la interfaz gráfica del sistema autónomo para determinar el precio de la fruta basado en visión artificial.

3. RESULTADOS

En este capítulo se muestran las pruebas realizadas para asegurar el correcto funcionamiento de todas las etapas del proceso para determinar el precio de frutas basado en visión artificial.

3.1. RESULTADOS DEL RECONOCIMIENTO DE FRUTAS

3.1.1 RECONOCIMIENTO EN BASE AL COLOR DE LA FRUTA

El MODULO III se encarga del reconocimiento del tipo de fruta a partir de su color y dimensiones. Cuando el programa reconoce un área con un determinado color, este lo enmarca con un cuadro del mismo tono. Es así, que en la Figura 3.1. Se observan los tres casos posibles de reconocimiento. Si el marco es diferente al color real de la fruta o, a su vez, no la encierra dentro de un cuadro, significa que el programa no está haciendo un reconocimiento correcto.

Figura 3.1. Reconocimiento de color a través de marcos que encierran el área en donde se encuentra dicho tono. a) Detección de color amarillo, b) detección de color rojo, c) detección de color verde.

En la Figura 3.2. se muestran imágenes que se adquirieron en la primera prueba del MODULO III. Cabe mencionar que se presentaron ciertos inconvenientes al momento de realizar el reconocimiento de color.

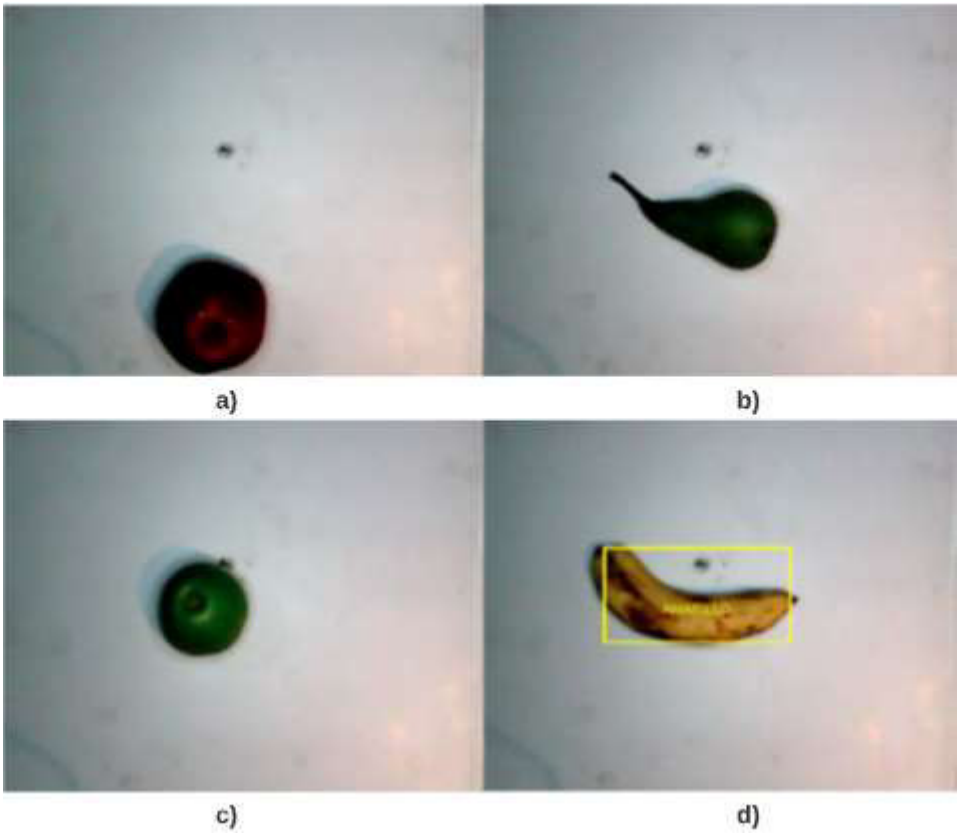


Figura 3.2. Primeras capturas para el reconocimiento de frutas. a) Manzana roja b) Pera verde c) Manzana verde d) Banana amarilla.

De la Figura 3.2. se observa que tanto las frutas rojas como verdes no fueron reconocidas por el módulo. Únicamente la fruta amarilla fue reconocida como tal y esto se evidencia al ser enmarcada por un cuadro amarillo.

3.1.1.1. Cambios en el soporte del sistema de adquisición de imágenes

Retomando el caso suscitado en las primeras adquisiciones de imágenes que se muestran Figura 3.2. se observa que, pese a contar con una lámpara circular que ilumina la bandeja, las frutas no cuentan con una iluminación adecuada y por lo tanto no fueron reconocidas (no se enmarcaron con ningún cuadro). Se optó por acercar la fruta a la cámara para comprobar si el problema era la iluminación o el algoritmo como tal. En la Figura 3.3. se observa el reconocimiento efectivo del color verde al momento acercar la fruta a la cámara al ser enmarcada con un cuadro verde. También se aprecia un reconocimiento erróneo de color amarillo debido al brillo de la fruta al estar tan cerca de la lámpara circular.

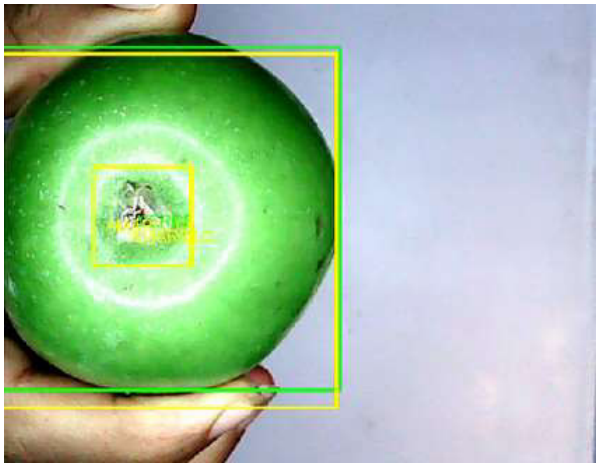


Figura 3.3. Reconocimiento de color verde y amarillo al momento de acercar la fruta a la cámara web.

Para resolver este inconveniente se optó por reducir la longitud del soporte del sistema de adquisición de imágenes, el cual tenía 60cm de largo. Ahora, el soporte consta de una longitud de 30cm y adicionalmente se le agregaron niveles para ir jugando con la longitud hasta conseguir una distancia que brinde los mejores resultados.

Figura 3.4. Implementación de niveles en el soporte para variar la longitud del mismo.

Después reducir la longitud del soporte del sistema de adquisición de imágenes, El reconocimiento de frutas mejoró significativamente. En la Figura 3.5. se observan capturas de imágenes que obtuvieron después de este cambio. Ahora todas las frutas ya se encuentran enmarcadas con el tono correspondiente de las mismas. Sin embargo, también aparecen áreas de la balanza en donde se reconoce el color amarillo erróneamente.

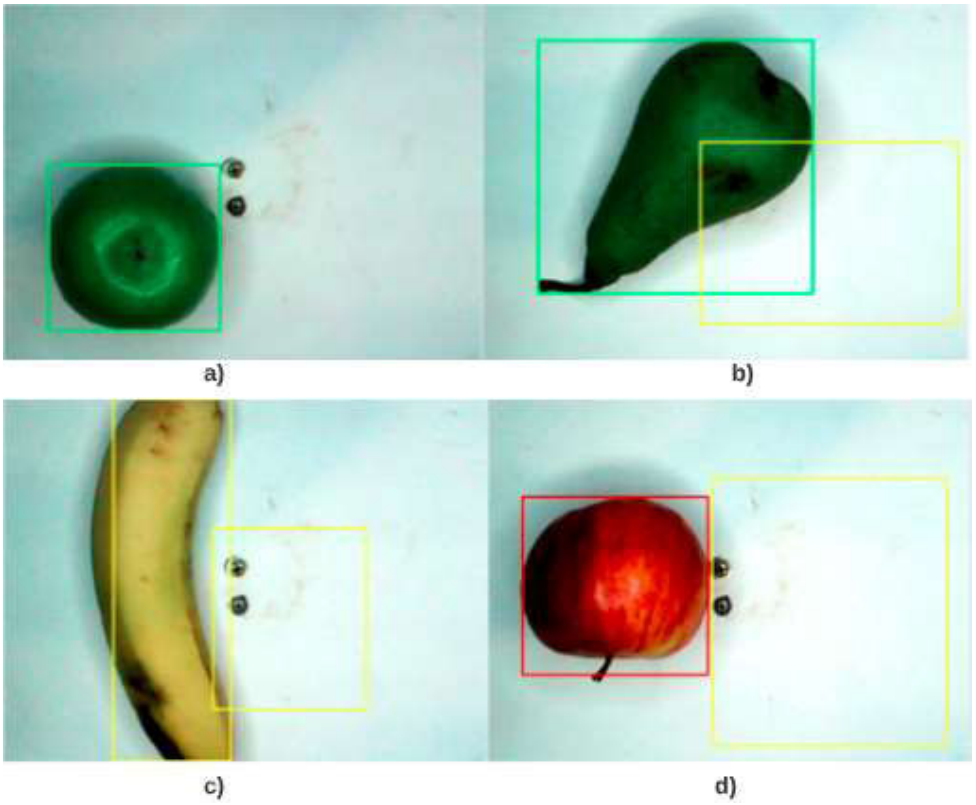


Figura 3.5. Imágenes de frutas adquiridas después de haber acortado la distancia entre la bandeja y el soporte del sistema de adquisición de imágenes. a) Manzana verde, b) Pera verde, c) Banana amarilla, d) Manzana roja

3.1.1.2. Brillo de la bandeja donde se colocan las frutas

El hecho de haber acercado el soporte que sostiene a la cámara web y la lámpara circular causó que aumente el brillo que reflejan las frutas y que las mismas sean reconocidas como una fruta amarilla. La superficie de la bandeja también presentó problemas al mostrar la mitad de la misma enmarcada con un cuadro amarillo. En la Figura 3.6. se observan las nuevas fallas que se representaron después de la decisión de reducir la distancia entre la bandeja y el soporte del sistema de adquisición de imágenes a la mitad.

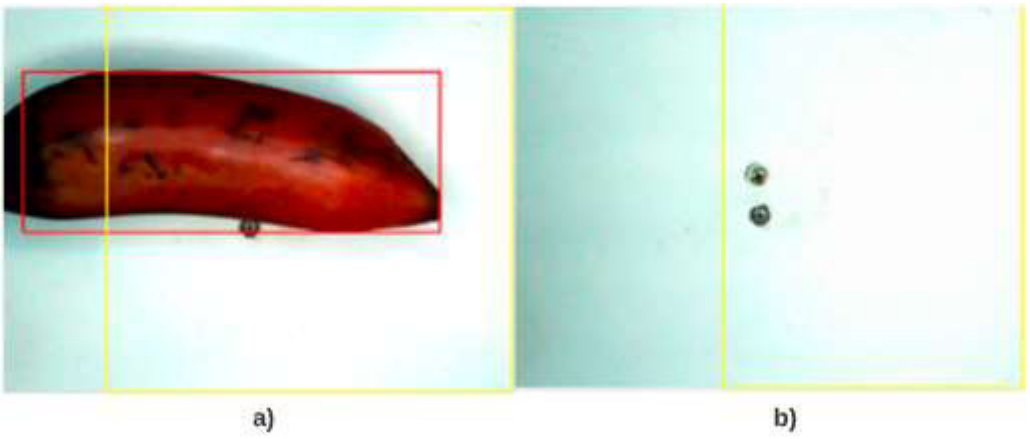


Figura 3.6. Reconocimiento erróneo del color amarillo en la superficie de la bandeja causado por acortar la distancia entre la bandeja y el soporte del sistema de adquisición de imágenes.

Para resolver este nuevo inconveniente se realizaron dos acciones:

i) Reducir la intensidad de la luz que emite la lampara circular: Cabe recordar que este dispositivo cuenta con un control para variar la intensidad y el color de la luz que se emite.

Figura 3.7. Conjunto de botones para configurar la intensidad de la luz emitida y el tipo de luz.

ii) Colocar papel difusor de luz sobre la lámpara circular para que la cantidad de luz que emite no se concentre en un solo lugar y esta intensidad se distribuya a lo largo de la bandeja.

Figura 3.8. Papel difusor colocado sobre la lámpara circular para distribuir la luz emitida de manera ecuánime sobre toda la bandeja de la balanza. a) Vista superior b) Vista inferior

3.1.1.3. Pruebas con diferentes tipos de color

Como se mencionó en el capítulo de construcción del prototipo del sistema autónomo para determinar el precio de frutas basado en visión artificial, una característica importante de la lámpara circular es que emite tres tipos de luces:

- Luz fría (4200-4500K).
- Luz cálida (3000-3500K).

- Luz diurna (6000-6500K).

Ahora bien, todas las pruebas que se han realizado hasta el momento han tenido a la luz fría como protagonista. Las tres tonalidades pueden ser utilizadas para realizar una serie de pruebas y así, determinar qué tipo de luz brinda mejores resultados al momento de reconocer colores y formas. En este apartado se muestran pruebas hechas con los otros dos tipos de luz para observar el comportamiento del sistema y determinar si es más conveniente para el reconocimiento y clasificación de frutas.

3.1.1.3.1. Luz cálida

Se colocan frutas sobre la bandeja, mientras la lámpara se encuentra emitiendo luz cálida. Las capturas de las imágenes de frutas bajo esta luz se muestran a continuación:

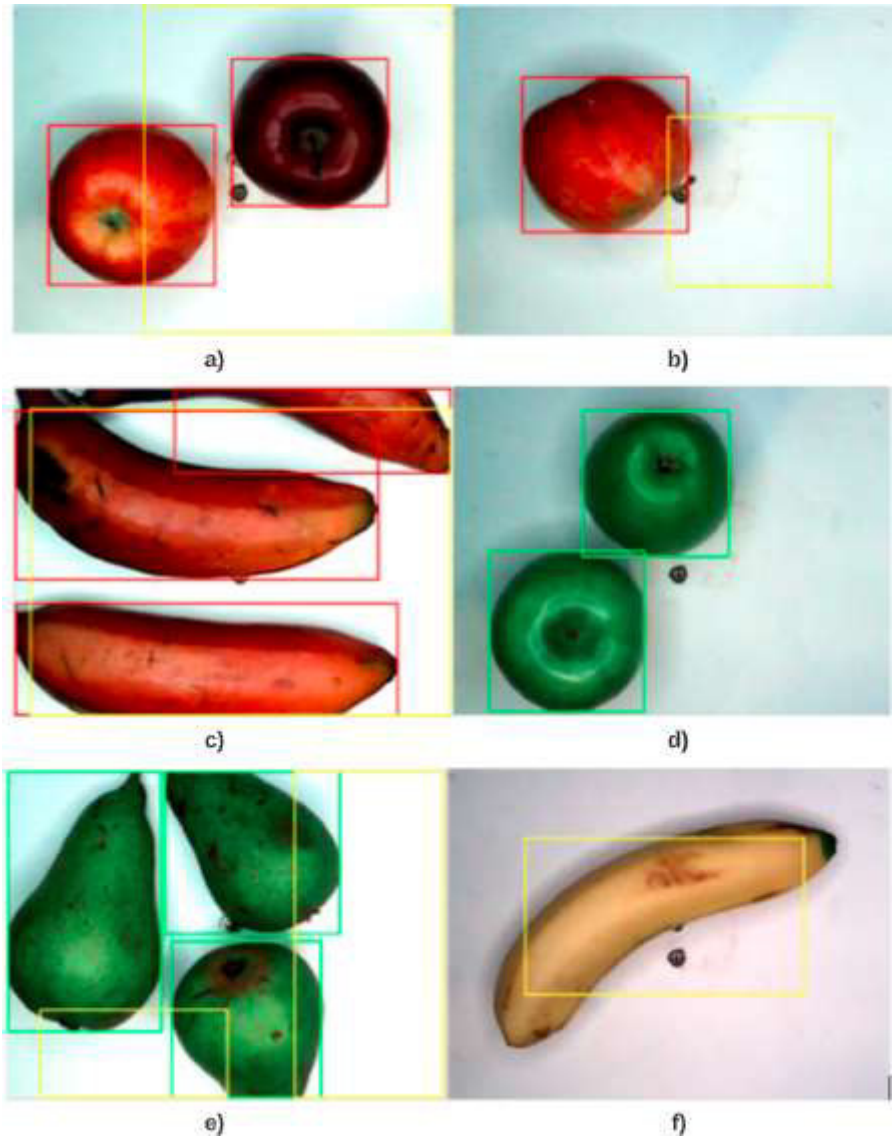


Figura 3.9. Adquisición de imágenes de frutas bajo la luz cálida. A) manzanas rojas b) manzana roja c) bananas amarillas d) manzanas verdes e) peras verdes f) banana amarilla.

De la Figura 3.9. se aprecia que el reconocimiento del color de las frutas de realiza de manera correcta. Sin embargo, regresa el inconveniente del brillo de la bandeja de la balanza para frutos rojos y verdes. Por lo que se descarta la utilización de este tipo de luz y se sigue utilizando el tipo de luz fría.

3.1.1.3.2. Luz de día

Finalmente, se harán pruebas del reconocimiento y clasificación de frutas bajo la luz de día. Los resultados se muestran en la Figura 3.10.

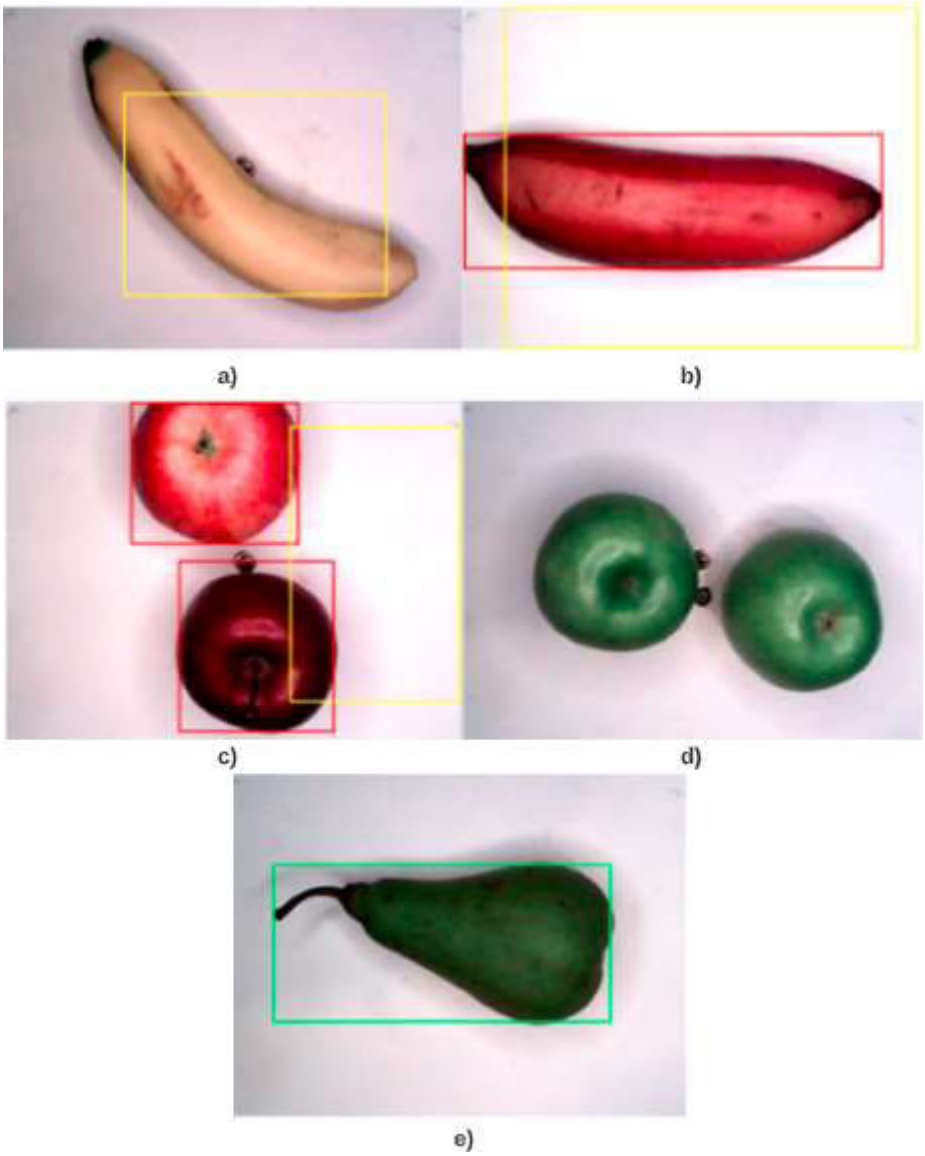


Figura 3.10. Adquisición de imágenes de frutas bajo la luz de día. a) banana amarilla, b) banana verde, c) manzanas rojas, d) manzanas verdes, e) peras verdes

De la Figura 3.10. se aprecia que el reconocimiento del color de las frutas se realiza de manera correcta, salvo en las manzanas verdes. Una razón puede ser que el tono verde de la manzana es un tanto amarillenta. Lo que se confunde con el tono amarillento de la luz y no

se reconoce como ninguna fruta al final. También vuelve a aparecer el problema del brillo de la bandeja por efecto de la luz, pero ahora con más fuerza.

Para tener más precisión acerca de la elección del tipo de luz idóneo, se recolectó un total de 4050 valores del reconocimiento de cada tipo de fruta con los tres tipos de luz y con sus respectivas intensidades. También se jugó con la cantidad de frutas sobre la balanza que fueron desde 1 hasta 5. Todos los valores que se obtuvieron se encuentran en el Anexo C. A continuación, en la Tabla 3.1. se mostrarán los porcentajes de precisión que se obtuvieron con cada tipo de luz.

Tabla 3.1. Porcentaje de precisión, reconocimiento erróneo, o no reconocimiento de frutas con cada tipo de luz.

LUZ	PRECISIÓN	ERRONEO	NO REC
FRIA	95%	3%	4%
DIA	94%	36%	1%
CALIDA	47%	39%	31%

Una vez que se determinó que la luz fría es la más óptima para el reconocimiento de frutas se observó su rendimiento de acuerdo a la intensidad de la luz y el número de frutas sobre la mesa.

Tabla 3.2. Precisión en el reconocimiento de frutas con cada intensidad de luz fría.

LUZ	INTENSIDAD	PRECISIÓN	P ERRONEO	NO REC
FRIA	1	85%	15%	17%
	2	80%	0%	17%
	3	100%	0%	0%
	4	99%	1%	0%
	5	99%	0%	0%
	6	100%	0%	0%
	7	96%	3%	0%
	8	97%	3%	0%
	9	99%	1%	0%

Tabla 3.3. Precisión en el reconocimiento de frutas de acuerdo a la cantidad de frutas sobre la balanza

LUZ	# FRUTA	Precisión	Reconocimiento Erróneo	No Reconocimiento
FRIA	1	100%	1%	0%
	2	97%	3%	3%
	3	95%	3%	4%
	4	92%	4%	4%
	5	90%	2%	12%

Es así que se justifica la utilización de la luz fría para el reconocimiento de frutas y se recomienda trabajar con una intensidad de luz mayor a 5 para obtener mejores resultados. También se observa que a medida que se aumenta el número de frutas sobre la balanza el porcentaje de precisión disminuye, aunque no de manera considerable.

3.1.2 RECONOCIMIENTO EN BASE A DIMENSIONES DE LA FRUTA

En cuanto al reconocimiento en base a dimensiones de la fruta, esto se lo aplicó únicamente a las frutas verdes, es decir, a las peras y manzanas verdes. El resto de frutas de otros colores se los clasificó únicamente en base al área que ocupaban dentro de la captura. El motivo por el cual no se utilizó el mismo criterio en el caso de las frutas verdes es que tanto peras como manzanas verdes ocupan la misma superficie aproximadamente por lo que se tuvo que emplear un método diferente para el reconocimiento. Es así, que se utilizaron las dimensiones de cada fruta para el reconocimiento, tomando en cuenta que la pera es más alargada que la manzana, mientras que la manzana es más bien circular.

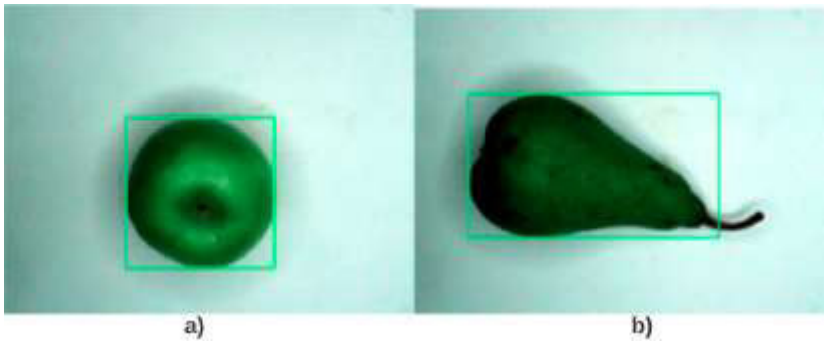


Figura 3.11. a) manzana verde b) pera verde

Lo mencionado anteriormente queda en evidencia en la Figura 3.11. Se hicieron 20 pruebas en total, utilizando diferentes frutas para comprobar el correcto funcionamiento, obteniendo resultados positivos en todos los casos. Gracias a las mejoras que se hicieron en el sistema de adquisición de imágenes, el reconocimiento de frutas en base a sus dimensiones no presenta ningún problema.

3.1.3 RECONOCIMIENTO FINAL DE LA FRUTA

Al configurar temas como intensidad de la luz que emite la lámpara circular y agregar un difusor de luz, el reconocimiento de frutas para el sistema autónomo mejoró totalmente. Los resultados de la implementación de ambas soluciones, así como el reconocimiento de frutas en base al color y dimensiones de esta se los aprecia en la Figura 3.12.

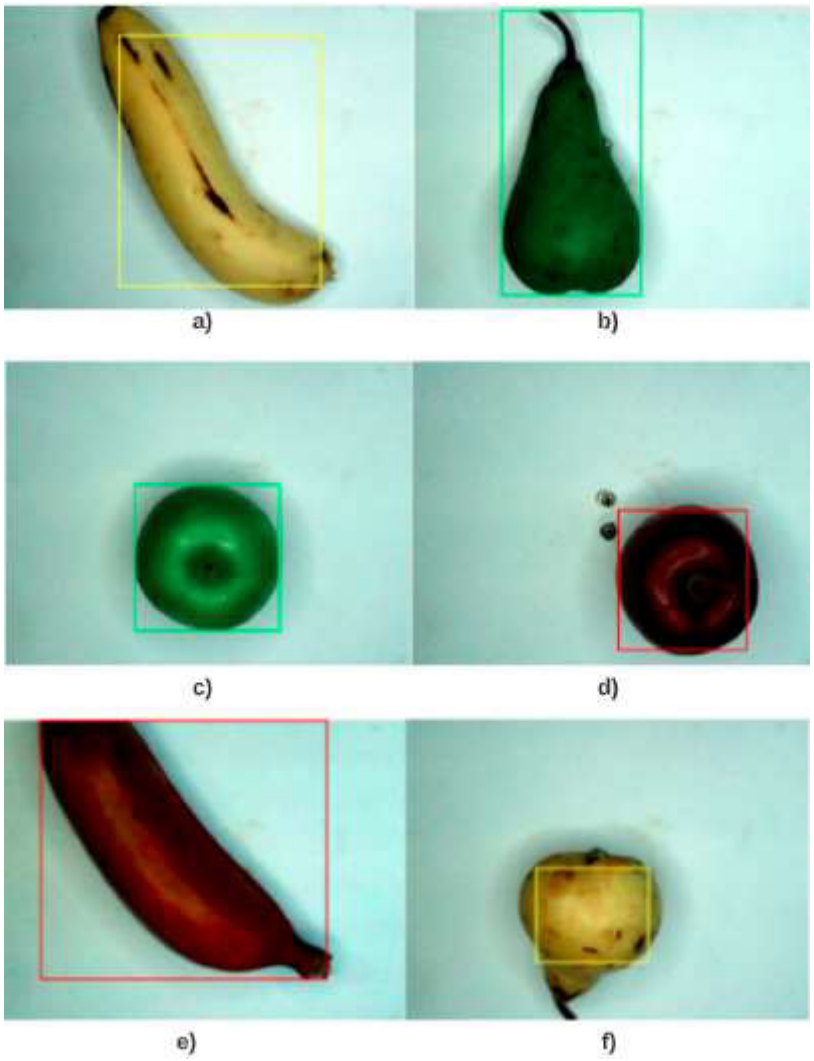


Figura 3.12. Captura de imágenes de frutas después de haber hecho los cambios mencionados anteriormente. a) banana amarilla, b) pera verde, c) manzana verde, d) manzana roja, e) banana roja, f) pera amarilla.

Cabe mencionar que el reconocimiento de frutas se realiza de forma individual y no importa si se coloca más de una fruta en la bandeja como se muestra en la Figura 3.13. El algoritmo seguirá reconociendo la fruta correcta para posteriormente, determinar el precio de las mismas.

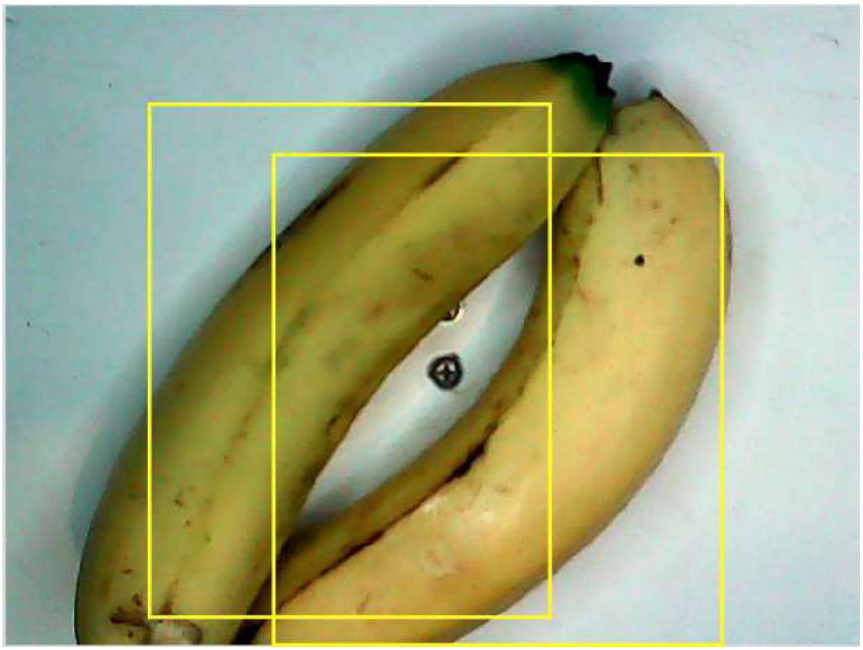


Figura 3.13. Reconocimiento de más de una fruta a la vez por el algoritmo para determinar el precio de las frutas basado en visión artificial.

3.2. PRUEBAS DEL MÓDULO DE PESAJE

En este apartado se hacen pruebas del MODULO III (Pesaje de las frutas) del sistema autónomo para determinar el precio de las frutas basado en visión artificial. En el capítulo de Metodología se explica la configuración que se hizo para adquirir el valor del peso, así como la calibración de la balanza para tener una correcta lectura del valor del peso de las frutas. Lo primero que se corroborará es si la lectura del peso es la correcta al utilizar pesos que ya fueron medidos con una balanza secundaria.

3.2.1. CALIBRACIÓN DEL PESO

La codificación para la adquisición del valor del peso se la realizó en IDE de Arduino. Nos enfocaremos en la toma de valores de peso para determinar si el valor de conversión estuvo correctamente calculado. Para lograr este objetivo se adquirieron cuatro pesos conocidos en los que constan elementos que no pueden faltar en la cocina. En la Tabla 3.4. Se detallan los pesos conocidos que se utilizaron para las pruebas del peso. Se eligieron dichos productos por su volumen, los cuales tienen diferentes dimensiones pero que su peso sigue siendo el mismo. Pueden presentarse fallas en la lectura del peso por el tema del área que cubre sobre la balanza, así que se pondrá a prueba este detalle.

Tabla 3.4. Pesos conocidos para probar la correcta calibración del módulo de peso.

Peso conocido	Valor del peso

Azúcar	0.5 [lb]
Maicena	0.5 [lb]
Máchica	1 [lb]

Las pruebas consistieron en ir colocando indiferentemente los pesos conocidos sobre la balanza para observar el valor del peso que era devuelto.

Maicena

Figura 3.14. Adquisición del peso de media libra de maicena.

Se observa que hace una correcta lectura del peso conocido de la maicena, el cual era de media libra (0.50 [lb]). También se hizo la lectura del peso con el programa IDE de Arduino, arrojando los siguientes valores:

```
No ponga ningun  objeto sobre la balanza
Destarando...
...
Listo para pesar
-0.00
-0.00
0.15
0.50
0.50
```

Figura 3.15. Adquisición del peso de media libra de azúcar con IDE de Arduino

Se observa que el valor inicial que arroja la lectura del peso es de cero libras y después de ubicar la maicena, el mismo se estabiliza en 0.50[lb] por lo que se concluye que la lectura del peso es correcta.

Azúcar

Figura 3.16. Adquisición del peso de media libra de azúcar.

Se observa que hace una correcta lectura del peso conocido del azúcar, el cual era de media libra (0.50 [lb]). También se hizo la lectura del peso con el programa IDE de Arduino, arrojando los siguientes valores:

```
No ponga ningun  objeto sobre la balanza
Destarando...
...
Listo para pesar
-0.00
-0.00
-0.00
0.23
0.50
0.50
```

Figura 3.17. Adquisición del peso de media libra de azúcar con IDE de Arduino

Se observa que el valor inicial que arroja la lectura del peso es de cero libras y después de ubicar el azúcar, el mismo se estabiliza en 0.50[lb] por lo que se concluye que la lectura del peso es correcta.

Machica

Figura 3.18. Adquisición del peso de una libra de máchica.

Se observa que hace una lectura del peso conocido del azúcar casi perfecto, se tiene un error de 0.01[lb], es decir un porcentaje de error del 1%. También se hizo la lectura del peso con el programa IDE de Arduino, arrojando los siguientes valores:

```
No ponga ningun  objeto sobre la balanza
Destarando...
...
Listo para pesar
-0.00
-0.00
0.16
0.99
0.99
0.99
```

Figura 3.19. Adquisición del peso de una libra de máchica con IDE de Arduino

Se observa que el valor inicial que arroja la lectura del peso es de cero libras y después de ubicar el azúcar, el mismo se estabiliza en 0.99[lb] dando un error de lectura del 1%.

En conclusión, se puede decir que la adquisición del valor del peso de la fruta es precisa en un 99%, por lo que se determina que no es necesario hacer cambios en este módulo y se conserva tal cual se lo estructuró en el capítulo 2.

Finalmente, se hace una prueba para determinar el máximo valor de peso que puede tolerar la balanza. En la Figura 3.20. se observa el esfuerzo al que fue sometida la balanza para llegar a su tolerancia máxima.

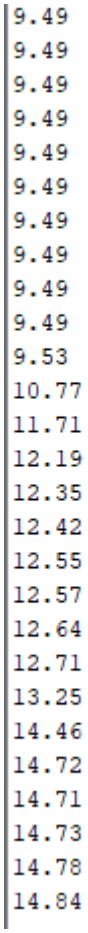


Figura 3.20. Rango de peso a las que fue sometida la balanza para determinar su máxima tolerancia.

De la Figura 3.20. se observa que el valor máximo de peso que logró transmitir la balanza es de 14.84[lb], aunque es recomendable no poner un peso mayor a 10[lb].

3.3. PRUEBAS FINALES DEL SISTEMA COMPLETO

Después de comprobar el correcto funcionamiento de los módulos del sistema autónomo para determinar el precio de frutas basado en visión artificial se procede a observar el funcionamiento de los tres módulos en conjunto a través de la interfaz gráfica. Para empezar, se muestra el diseño de la interfaz gráfica que se desarrolló en el capítulo 2.3. para visualizar sus partes:

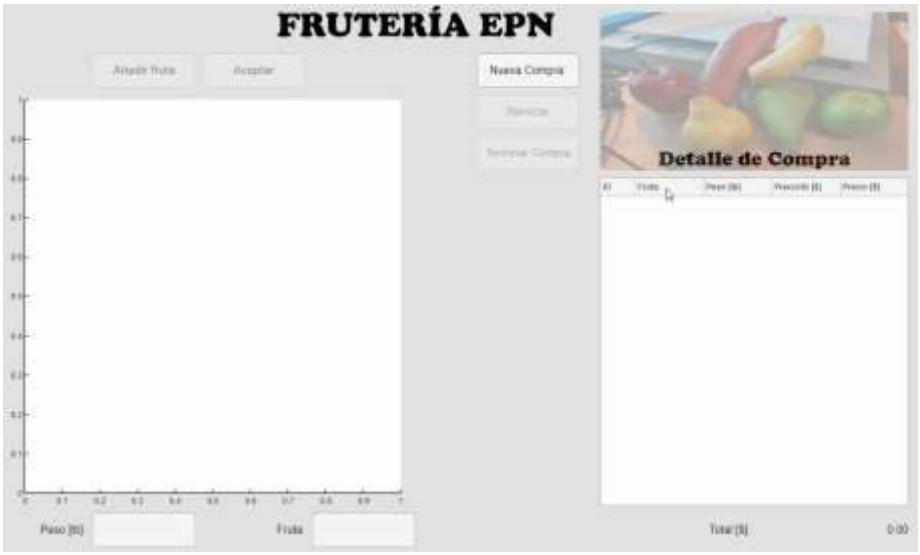


Figura 3.21. Interfaz gráfica al principio de la compra.

Al momento de dar click en “Nueva Compra” se habilitan tres botones: “Reiniciar”, “Terminar Compra”, y “Añadir Fruta”.

- El botón “Reiniciar” sirve para borrar todas las compras que se han hecho previamente. Como no se ha adquirido ninguna fruta, al momento de aplastar este botón no sucede nada.
- El botón “Terminar Compra” realiza la suma de todas las frutas que han sido adquiridas e imprime el valor que se debe pagar en la parte inferior derecha de la interfaz.
- El botón “Añadir Fruta” arranca todo el programa e inicia el reconocimiento de frutas, así como el pesaje de estas.

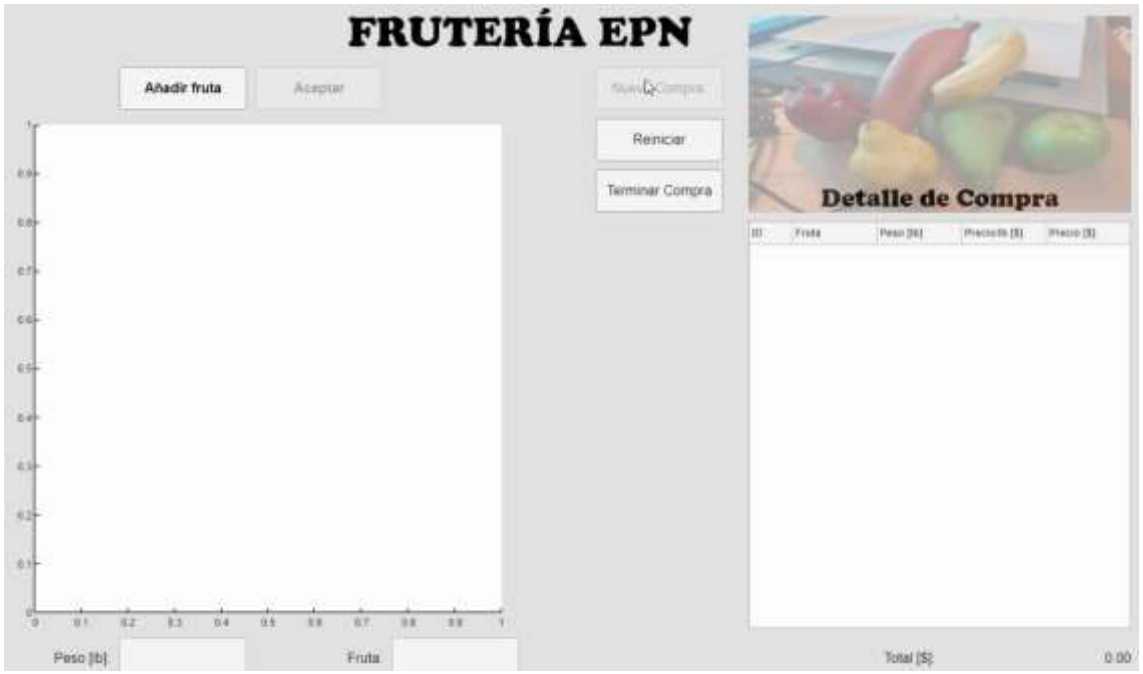


Figura 3.22. Habilitación de los botones Reiniciar", "Terminar Compra", y "Añadir Fruta" después de haber dado click en "Nueva Compra"

Se da click en "Añadir fruta" y se empiezan a colocar las frutas que se desea comprar sobre la bandeja de la balanza. En primer lugar, se coloca una banana amarilla.

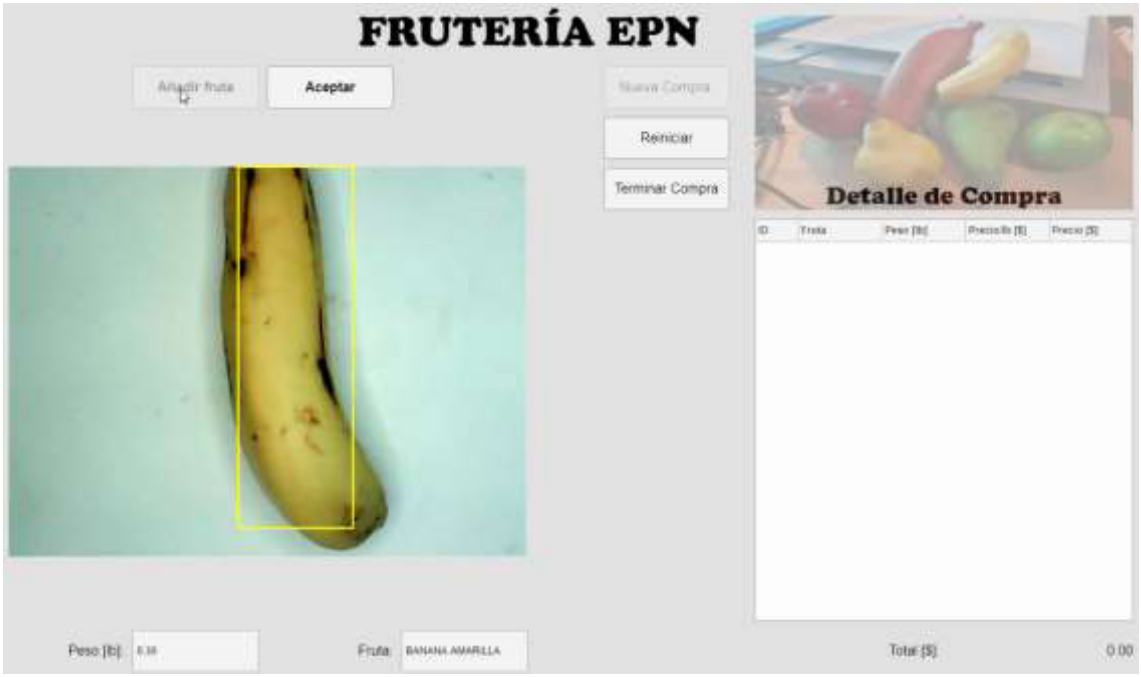


Figura 3.23. Banana amarilla colocada sobre la balanza.

De la **Figura 3.23.** se aprecia el reconocimiento de la fruta. En la parte inferior se muestra impreso el peso de esta y su nombre.

Si el usuario está de acuerdo con el peso y el reconocimiento que ha hecho el programa acerca de la fruta, se da click en “Aceptar”.

El botón “Aceptar” termina el proceso de reconocimiento y agrega la fruta al detalle de compra. Si el usuario no está de acuerdo esta compra puede hacer click en “Reiniciar” para borrar dicha compra o si no desea comprar más frutas debe dar click en “Terminar Compra” para saber el valor a pagar por su adquisición. Si desea comprar comprar más frutas dará click en “Añadir fruta” para correr de nuevo el programa

Figura 3.24. Fruta agregada al detalle de compra al haber dado click en el botón “Aceptar”

Se da click en “Añadir Fruta” para continuar la compra de otras. Ahora se desea comprar una pera amarilla.

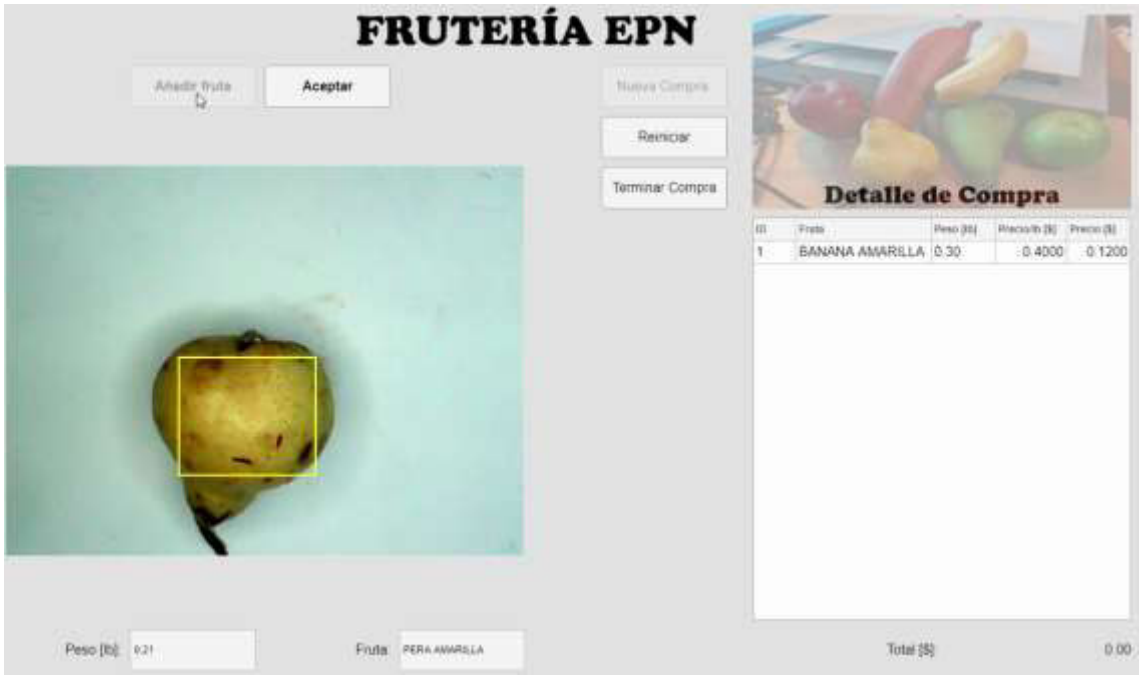


Figura 3.25. Reconocimiento de una pera verde al dar click en “Añadir Fruta”.

De la Figura 3.25. Se observa que el sistema ha reconocido una pera amarilla, cuyo peso es de 0.21[lb]. Como se desea agregar a la compra, damos click en “Aceptar” y se mostrará dicha fruta en Detalle de Compra.

Figura 3.26. Pera amarilla agregada al detalle de compra después de haber dado click en “Aceptar”

Ahora se desea comprar una pera verde, por lo cual se da click nuevamente en “Añadir Fruta”. En la Figura 3.27. Se muestra a la fruta reconocida y cuyo peso es de 0.32[lb].

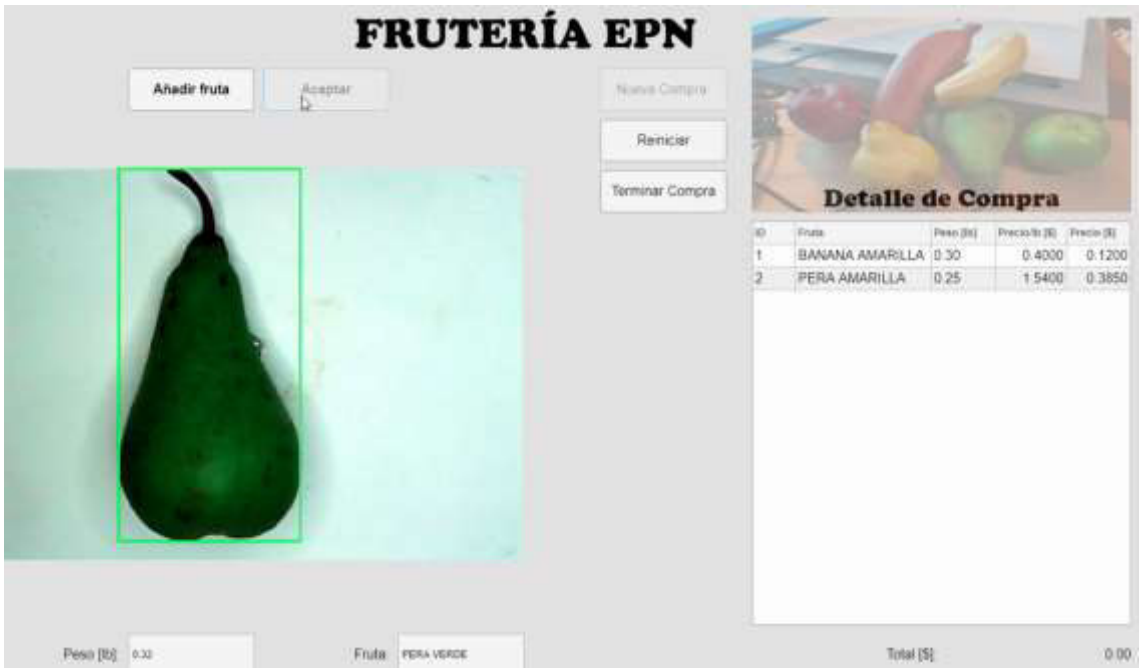
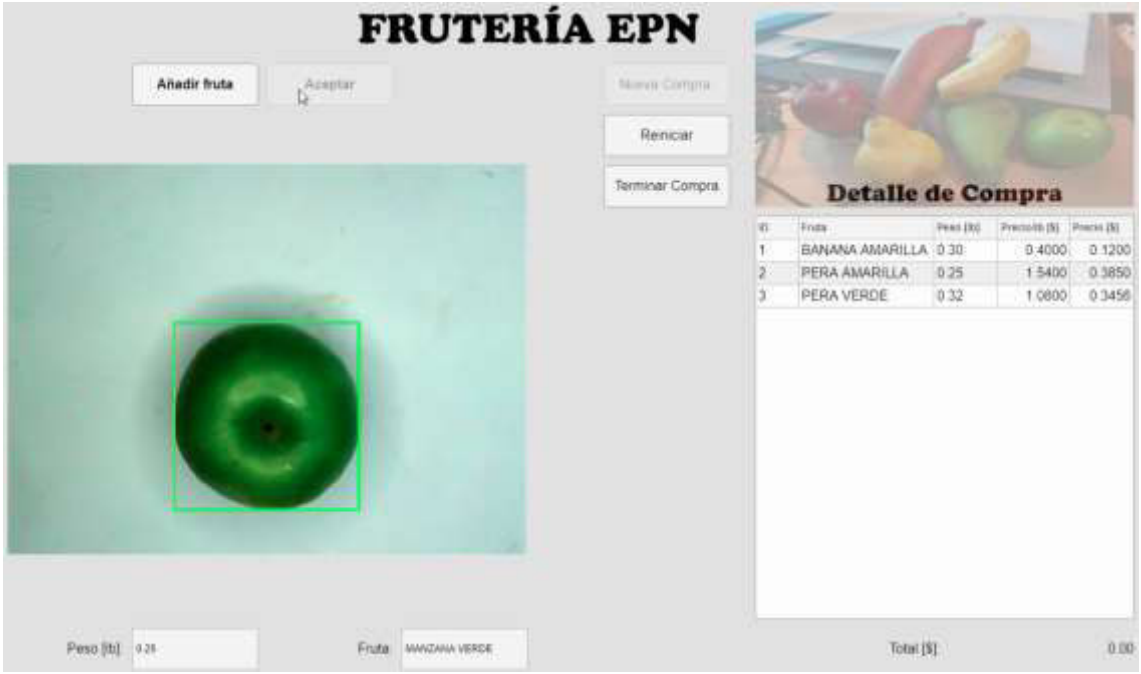
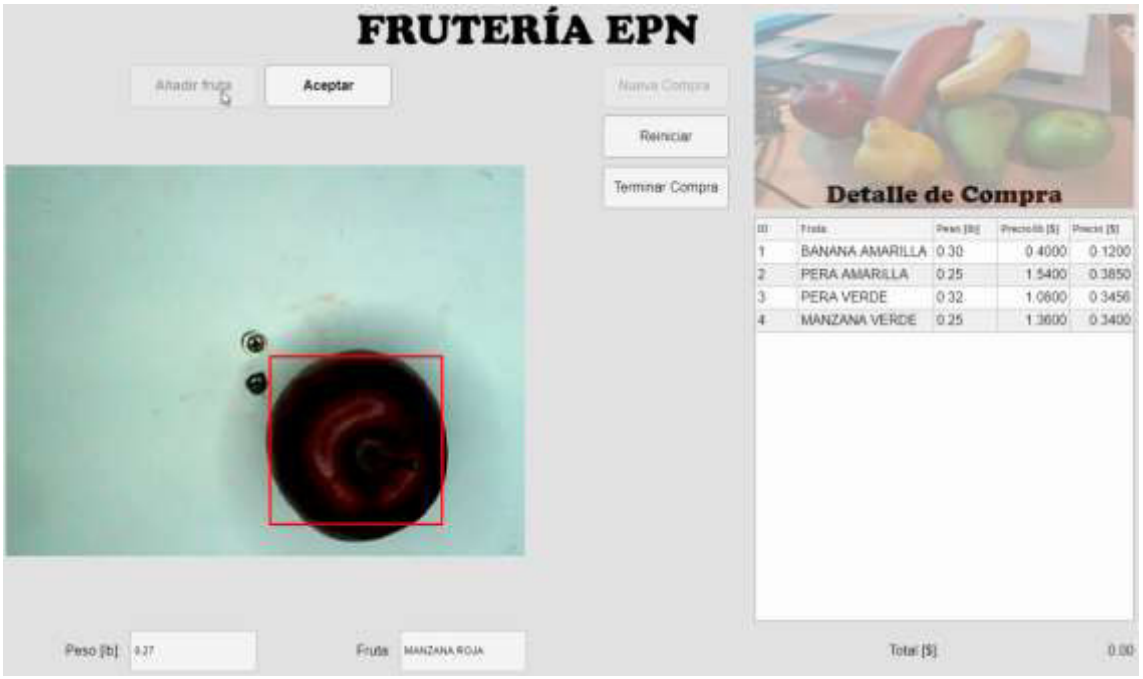


Figura 3.27. Reconocimiento de pera verde con peso de 0.32 [lb].

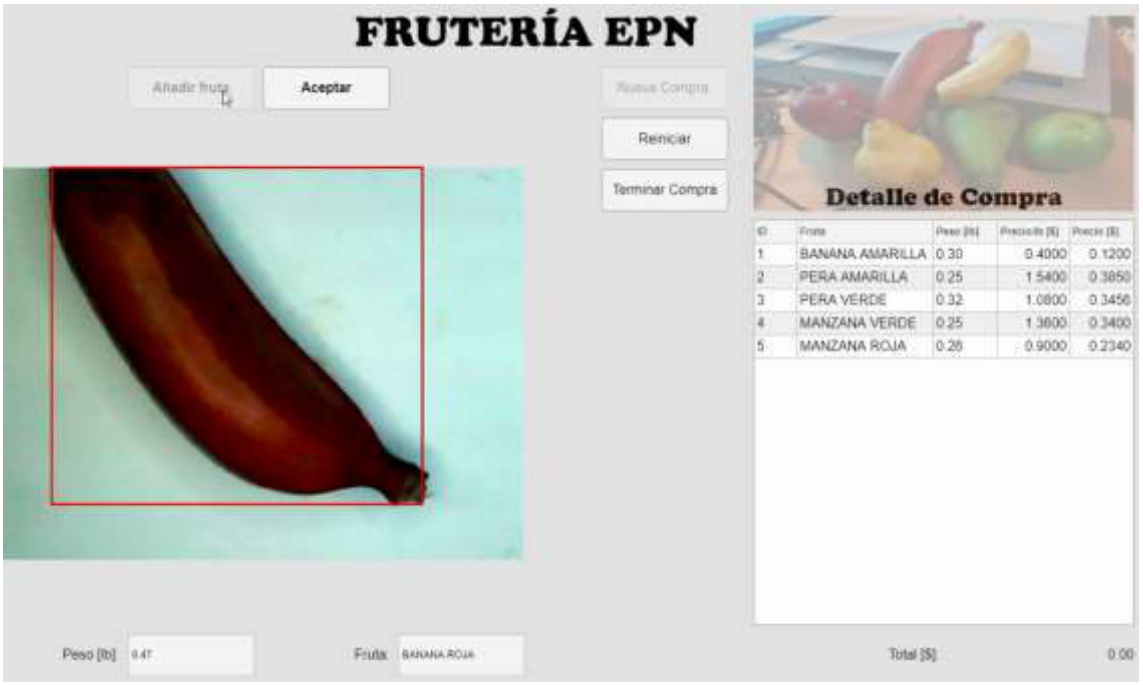
Es así, que se continúa con la adquisición de diferentes tipos de frutas como manzanas verdes, manzanas rojas y bananas rojas, que son los seis diferentes tipos de frutas que se propusieron reconocer al principio de la tesis.



a)



b)



c)

Figura 3.28. Reconocimiento y detalle de compra de a) una manzana verde, b) manzana roja, y c) banana roja respectivamente.

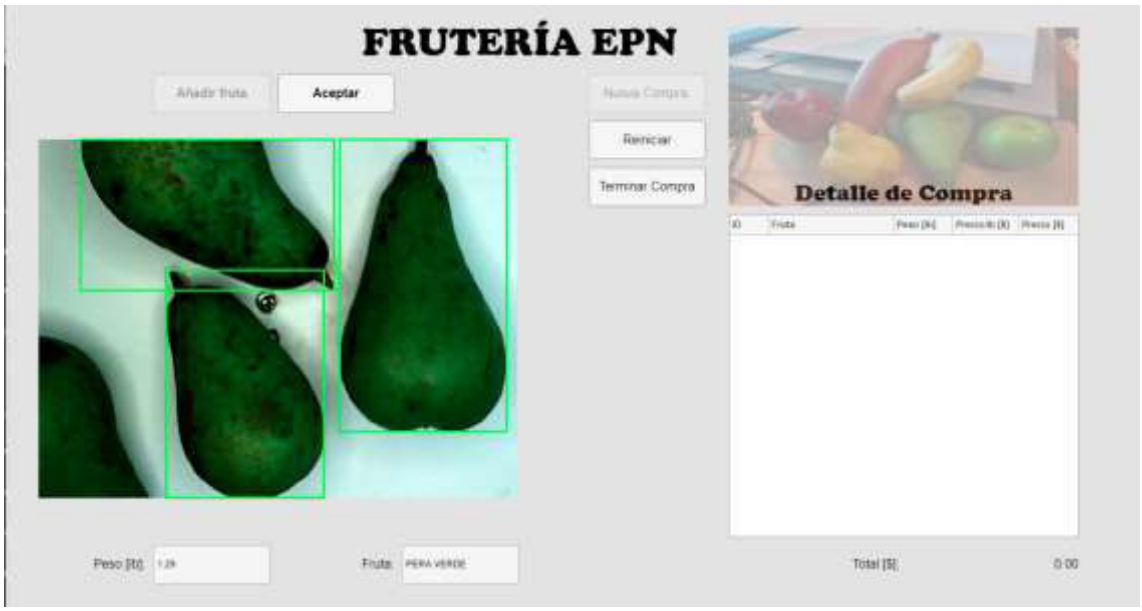
Una vez agregadas todas las frutas que se desean adquirir, damos click en “Terminar Compra” y el programa imprimirá el valor del precio a pagar por las frutas en la parte inferior

del detalle compra. En este caso se debe pagar $1.70 por las 6 frutas que reconoce el sistema autónomo para determinar el precio de frutas basado en visión artificial.

Figura 3.29. Detalle del pago por las seis frutas que se desean adquirir.

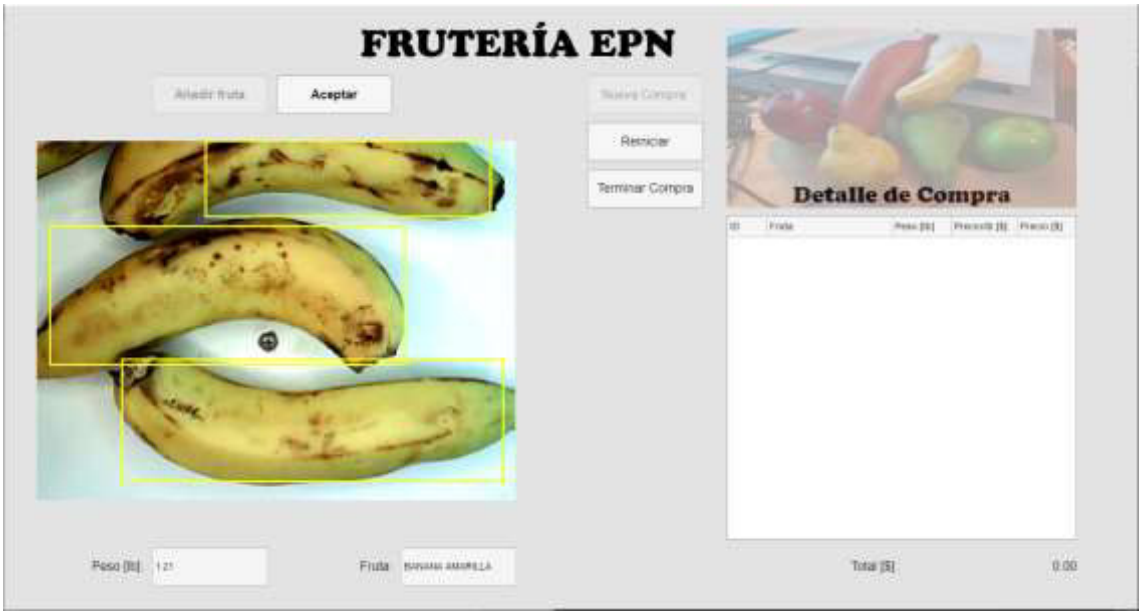
Después de dar click en “Terminar compra”, solo se habilita el botón “Nueva Compra” para nuevos usuarios que deseen hacer una compra.

Cabe recalcar que el programa también reconoce varias frutas de la misma especie a la vez, como se muestran en las **Figura 3.30.**



a)

b)



c)

Figura 3.30. Reconocimiento de varias frutas de la misma especie a la vez, a) peras verdes, b) bananas rojas, y c) bananas amarillas.

3.3.1. RECONOCIMIENTO DE FRUTAS DE DIFERENTE TIPO SOBRE LA BALANZA

Un tema de vulnerabilidad que puede ocurrir al realizar una compra. Puede darse el caso en que el usuario coloque dos frutas de diferente tipo sobre la balanza y el sistema continúe su ejecución con normalidad. Para solucionar este inconveniente se aumentó una sección en donde se advierte al usuario que existen al menos dos frutas de diferente tipo sobre la balanza y que debe quitar una para continuar con su compra. El sistema bloquea el botón “Aceptar”

que agrega las frutas al “Detalle de compra” hasta que no se quite una de las frutas de diferente tipo, dando cierto nivel robustez al sistema autónomo para determinar el precio de frutas basado en Visión Artificial.

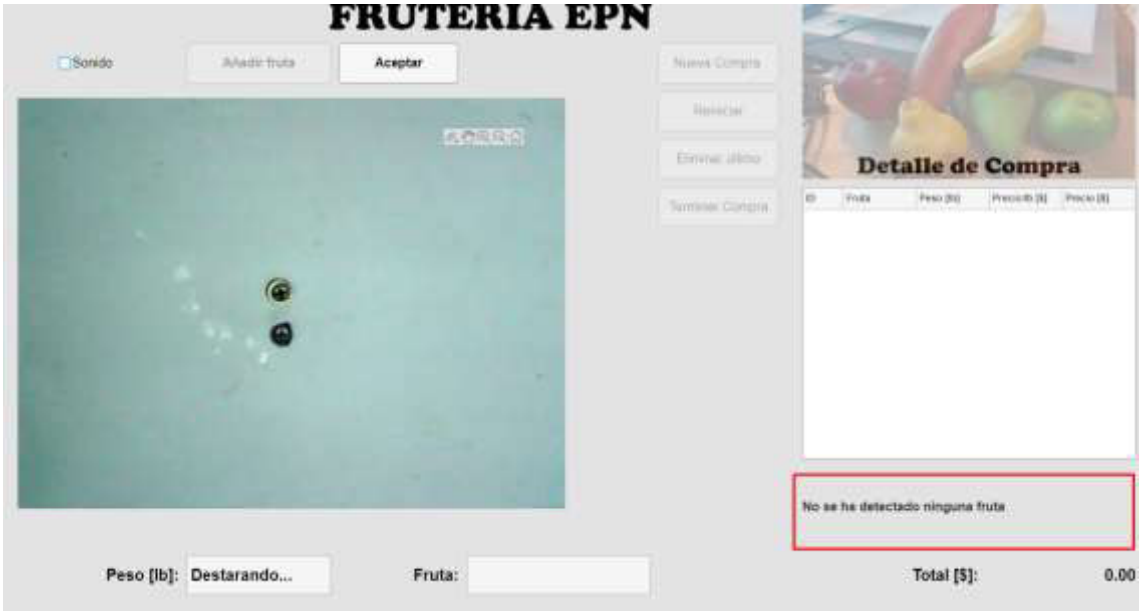


Figura 3.31. Sección de notificaciones que indica que no se ha colocado ninguna fruta sobre la balanza.

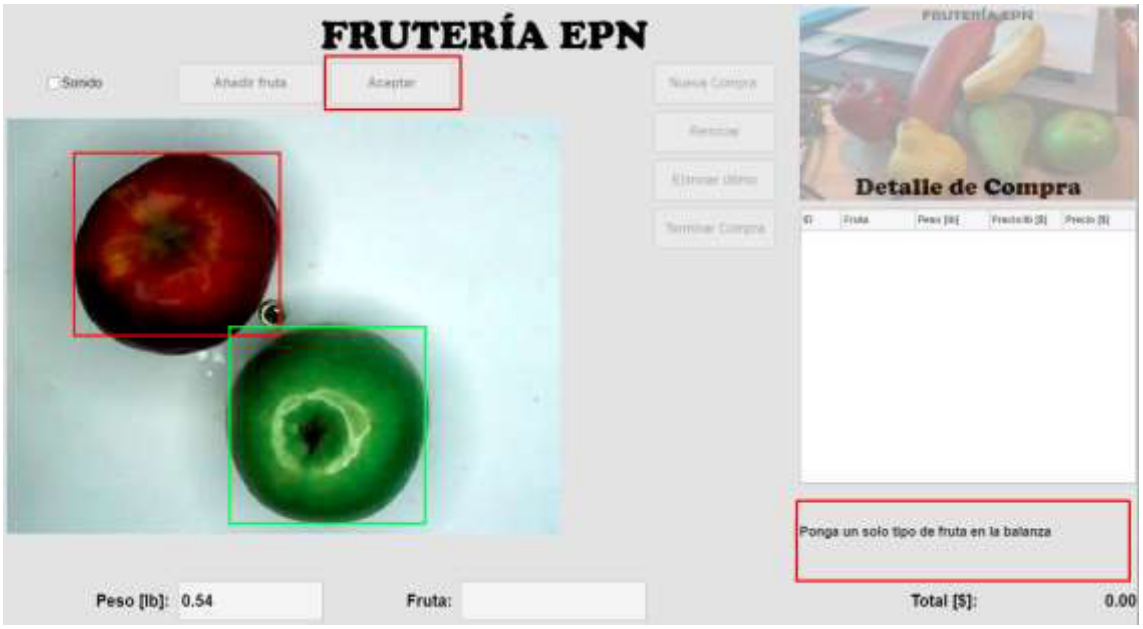


Figura 3.32. Sección solicita al usuario que coloque un solo tipo de fruta. Se bloquea el botón “Aceptar” que agrega las frutas al “Detalle de compra”

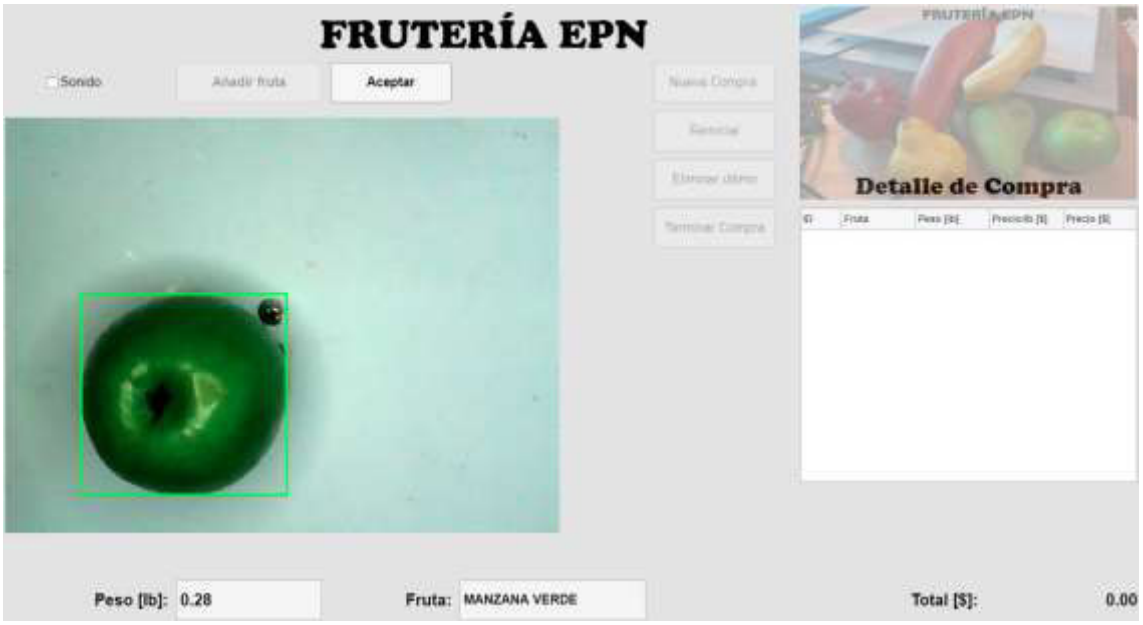


Figura 3.33. El botón “Aceptar” se desbloquea cuando el sistema reconoce un solo tipo de fruta.

3.3.2. TIEMPO TOTAL DE PROCESAMIENTO

El saber el tiempo total de procesamiento que toma reconocer una fruta. Se utilizaron los comandos “tic” y “toc” y así, verificar los tiempos exactos. Es así que se obtuvieron los siguientes valores:

Tiempo de encendido de la cámara: 1.96 segundos.

Procesamiento de imagen: 0.374976 segundos.

Reconocimiento de fruta: 0.001476 segundos.

Tiempo total: 2.34 segundos.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

- La tasa de exactitud en el reconocimiento de colores difiere entre cada tono. El color verde no presenta fallas al momento de su reconocimiento, asegurando un correcto reconocimiento en el 100% de ocasiones. Esta exactitud disminuye al trabajar con frutos rojos y amarillos, detectando correctamente el color en 38 de 40 frutas probadas. Esto disminuye su exactitud a un 95% de efectividad.
- El algoritmo para el reconocimiento de frutas por sus colores alcanzó una gran exactitud después de hacer los cambios en la adquisición de imágenes, llegando a reconocer correctamente el color de 58 frutas de 60 probadas. Este es un indicador de que reconocerá correctamente el color en el 98% de veces.
- El reconocimiento de frutas por su tamaño no presentó problemas de exactitud. Se probaron un total de 24 frutas por cada color para determinar un umbral que separase a cada tipo de fruta de manera óptima. Los valores de las superficies que ocupa cada tipo de fruta son tan alejados que la probabilidad de un error en el reconocimiento de la fruta es nula.
- El porcentaje de exactitud en cuanto al reconocimiento de frutas en base a su tamaño es del 100% siempre y cuando haya detectado correctamente el color de la fruta. Caso contrario, dicha fruta no será reconocida correctamente.
- El pesaje de las frutas tiene una exactitud del 99%, ya que difiere en una centésima de libra del peso real de las frutas. Se puede afirmar que el prototipo cumple satisfactoriamente con el objetivo de determinar el peso de la fruta para calcular su precio.
- Debido al 1% de error que presenta el sistema de pesaje, el costo total que el usuario debe pagar por sus frutas presentará el mismo error. El problema no es tan significativo para pequeñas compras. Sin embargo, a medida que la compra sea más grande, el problema del costo será más evidente.
- Se creó un prototipo de un sistema autónomo para determinar el precio de frutas de manera óptima al utilizar la visión artificial como base para el desarrollo del algoritmo. Con esto se logra reducir el procesamiento excesivo de trabajos anteriores y disminuir el tiempo de respuesta para entregar al usuario el precio que debe pagar por su compra.
- Se demuestra que la implementación de métodos modernos tales como Inteligencia Artificial no siempre son la mejor opción para dar solución a temas tales como el reconocimiento de frutas. Utilizar librerías que demandan mucho procesamiento para

su funcionamiento hace que el programa final carezca de exactitud y optimización de recursos.

- Controlar el entorno completo de la adquisición de imágenes es el punto clave para realizar un correcto reconocimiento de frutas. Al momento, ningún algoritmo, por más robusto que éste sea, será capaz de realizar la tarea para la que fue creada si los datos que recibe son de mala calidad.
- El mercado de frutas es un área de negocios que no ha sido potenciado por la tecnología actual. Por lo que este tipo de sistemas para determinar el precio de frutas representa una mejora en el servicio al cliente, así como la supresión de personal para cobrar por la compra representa una mejora para el dueño de la frutería.

4.2. RECOMENDACIONES

- Al momento de iniciar la captura de imágenes, el programa reconocerá una fruta amarilla como defecto. Esto debido a que la cámara lanza un flash cuando se activa. Este pequeño detalle no afecta al módulo de reconocimiento, ya que cambiará inmediatamente cuando se coloque una fruta sobre la bandeja de la balanza.
- El estado de madurez, así como el deterioro de la fruta repercute directamente en el reconocimiento correcto de esta, por lo que se recomienda trabajar con frutas que se encuentren en buen estado.
- Se recomienda variar la intensidad de la luz de acuerdo a la iluminación del entorno en donde se encuentra el prototipo, así como la hora del día. Si se hace el reconocimiento de día se debe bajar la intensidad de luz, mientras que se debe aumentar durante la noche.
- Limpiar la bandeja después de cierto tiempo ya que, al mancharse la superficie de la bandeja, ciertas áreas manchadas podrían ser reconocidas como una fruta, cuando no se encuentra nada ahí.
- Para un trabajo futuro, se recomienda ir cambiando los valores de los umbrales para que el programa sea capaz de diferenciar entre más tipos de frutas, haciéndola más atractiva para los dueños de fruterías.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] C. Pinasco, Los Supermercados del Futuro, 2019. [En línea]. Available: https://www.youtube.com/watch?v=UpO1QBkKbJA. [Último acceso: 25 02 2021].

[2] D. Jarrín y V. Darwin, Desarrollo de un sistema prototipo de pago de frutas a través del entrenamiento de redes neuronales artificiales convolucionales, Escuela Politécnica Nacional, Quito - Ecuador, 2019.

[3] R. Fisher y S. Perkins, Morfological Operators. Hypermedia Image, New York: J. Wiley & Sons, Ltd 1°ed, 2003.

[4] The MathWorks, Inc., Desarrollar apps mediante App Designer, MathWorks, 2021. [En línea]. Available: https://la.mathworks.com/help/matlab/app-designer.html. [Último acceso: 11 04 2021].

[5] Mechatronics, Tutorial transmisor de celda de carga HX711, Balanza Digital, 2014. [En línea]. Available: https://naylampmechatronics.com/blog/25_tutoria1-trasmisor-de-celda-de-carga-hx711-. [Último acceso: 27 02 2021].

[6] J. Guerrero Hernandez y G. Pajares Martinsanz, Técnicas de Procesamiento de imágenes, Universidad Complutense de Madrid, Madrid, 2010.

[7] C. Aspiazu, Aplicación de Histogramas y Filtros Matlab, Escuela Politécnica del Litoral, Guayaquil, 2012.

[8] J. Esqueda, Fundamentos de Procesamiento de Imágenes, Universidad Autónoma de Baja California, Tijuana, 2002.

[9] V. Borja, Herramientas computacionales para la matemática, UTM, 03 2013. [En línea]. Available: http://www.utm.mx/~vero0304/HCPM/7-hipermatrices.pdf. [Último acceso: 12 05 2021].

[10] A. Pallas, Sensores y Acondicionadores de Señal, Marcombo S.A., Barcelona, 2001.

[11] J. Ferrero, Instrumentación Electrónica, Sensores, Servicio de publicaciones UPV, 1994.

[12] F. Zambrano, Diseño e Implementación de un Sistema Automatizado de Dosificación por Peso de Agua y Aceite para la Elaboración de Salsas para la Empresa MARCSEAL S.A., Quito, 2018.

[13] J. Diaz, Mi Arduino, 21 01 2016. [En línea]. Available: http://www.iescamp.es/miarduino/2016/01/21/placa-arduino-uno/. [Último acceso: 06 04 2021].

[14] M. Ferazi y A. Dahoud, Integrated Development Enviroment "IDE" for Arduino, Research Gate, New York, 2018.

[15] G. Flores y R. Hecker, Diseño Preliminar de una celda de carga para maquinado, Asociación Argentina de Mecánica Computacional, Santa Fe, 2006.

[16] Luis del Valle, Amplificador HX711 con Arduino para crear una báscula digital, Programar fácil, 2018. [En línea]. Available: https://programarfacil.com/blog/arduino-blog/hx711-arduino-bascula-digital/. [Último acceso: 12 04 2021].

[17] O. Fernández, Arduino celda de carga con hx711, Código Electrónica, 2021. [En línea]. Available: http://codigoelectronica.com/blog/arduino-celda-de-carga-con-hx711. [Último acceso: 12 04 2021].

[18] L. d. Valle, Amplificador HX711 con Arduino para crear una báscula digital, Programar Fácil, 2015. [En línea]. Available: https://programarfacil.com/blog/arduino-blog/hx711-arduino-bascula-digital/. [Último acceso: 13 04 2021].

[19] 2021 Arduino, Downloads, Arduino CC, 2021. [En línea]. Available: https://www.arduino.cc/en/software. [Último acceso: 12 04 2021].

[20] P. Jasen y E. González, Introducción a la Teoría de Errores de Medición, EUDEBA, Buenos Aires, 2002.

[21] Made-in-China, Focus Technology Co., 2021. [En línea]. Available: https://www.made-in-china.com/products-search/hot-china-products/PC_Camera.html?gclid=Cj0KCQiAvvKBBhCXARIsACTePW-9hUlFTHGJCbrMBw42jUgFAerTqr5X-eqT47PxY3yTJ8FVeqBih8YaAmJWEALw_wcB. [Último acceso: 01 Marzo 2021].

[22] Amazon, Aro de luz LED, 2021. [En línea]. Available: https://www.amazon.com/-/es/pulgadas-transmisi%C3%B3n-escritorio-maquillaje-fotograf%C3%ADa/dp/B07PPRX94L. [Último acceso: 01 Marzo 2021].

[23] TIRE RACK, ¿Qué es la escala Kelvin y qué valor de bombilla debo escoger?, 2021. [En línea]. Available: https://m.tirerack.com/tires/tiretech/techpage.jsp?techid=170&ln=sp#:~:text=Kelvin%20es%20una%20unidad%20de,una%20fuente%20espec%C3%ADfica%20de%20luz.&text=Bombillas%20con%204200K%20o%20m%C3%A1s,de%20la%20luz%20del%20sol.. [Último acceso: 01 Marzo 2021].

[24] Preciosmundi 2013, Precios en Ecuador 2021, preciosmundi, 2021. [En línea]. Available: https://preciosmundi.com/ecuador/. [Último acceso: 24 04 2021].

ANEXOS

ANEXO A: IMÁGENES DONDE SE EVIDENCIAN PROBLEMAS EN EL RECONOCIMIENTO DE COLORES

ANEXO B: HISTOGRAMAS DE LAS IMÁGENES DEL ANEXO A

ANEXO C: TABLA DE VALORES DE RECONOCIMIENTO DE FRUTAS CON CADA TIPO DE LUZ Y SUS INTENSIDADES

ORDEN DE EMPASTADO