Portada

INDICE / Contenido

Agradecimiento/Dedicatoria (opcional) 1 c/u **esteban/robin**

Resumen **Esteban**

Palabras Claves **Esteban**

Capitulo 1 – Introduccion

Introducción **Robin**

Fundamentos **Robin**

Industria del limón, Exportación, Como debe ser el limón **Robin**

Objetivos **Esteban**

Capitulo 2 - Desarrollo

Software utilizado (Descripción General) **Esteban**

Python

Opencv

PyQt

Qt Designer

Visual Studio Code

Diagrama general del proyecto compuesto por cámara-pc(Python)-esp-32- servomotor-display-led

Hablar de cada componente **Robin**

Camara Web

Iluminacion

Esp-32

Servomotor

Display

Software Implementado

Conceptos previos (POO, clases, Hilos)

Captura de Imagen / Video **Esteban**

Selección de área para analizar **Esteban**

Detección de color **Esteban**

Encontrar centro y diámetro **Esteban**

Contador de limones **Esteban**

Interfaz gráfica **Robin**

Diagrama de flujo

Hardware Implementado

Control por serial **Esteban**

Manejo de servos – código **Esteban**

Código para display **Esteban**

Plaqueta del Proyecto

Cuarto de visión **Robin**

Capítulo 3 Resultados

Capítulo 4 Análisis de costos

Capítulo 5 Conclusiones

Anexos

Instalación de Python y librerías

Códigos Usados

Bibliografía

# Resumen

En el presente proyecto se realizó un sistema que utiliza el procesamiento de imágenes para la clasificación automática de limones.

Para llevarlo a cabo se utilizó el lenguaje de programación Python, junto a la librería OpenCV, con los cuales se procesan las imágenes procedentes del cuarto de visión, conformado por una web cam y el sistema de iluminación, el resultado del procesamiento es enviado al microcontrolador esp-32 mediante comunicación serial para que se encargue del accionamiento de los servomotores.

Se utilizó el entorno de desarrollo Visual Studio Code para el desarrollo del programa de PC, y QtDesigner para el desarrollo de la interfaz gráfica.

Una vez finalizado se logró clasificar los limones entre los que son aptos para exportación, los que son para consumo local y los que son descartados, conocer el diámetro de cada limón y calcular cuántos y que cantidad de kilogramos son procesados.

**Palabras Clave:** Clasificación de limones, procesamiento de imágenes, Python, OpenCV

# Objetivos

## Objetivo General

Diseñar y construir un sistema clasificador de limones, implementando la selección de color y tamaño por medio de visión artificial para la distinción del fruto tipo exportación.

## Objetivos Específicos

* Diseñar y construir un sistema que permita la clasificación de limones para seleccionar el limón adecuado según la norma de calidad internacional.
* Diseñar un programa de adquisición de imágenes que permita la clasificación del limón por el color y tamaño.
* Diseñar la interfaz gráfica (GUI) que permita ver la información resultante.
* Diseñar los circuitos de control para cada uno de los actuadores del sistema.
* Construir un prototipo funcional que permita realizar pruebas de funcionamiento de los sistemas que clasifican el fruto.

# Capítulo 1 - Introducción

## 1.0 Introducción

Este sistema para la clasificación de limones se presenta como Proyecto Final de la carrera de Ingeniería Electrónica de la Universidad Tecnológica Nacional, Facultad Regional Tucumán. En el presente informe se describirá el funcionamiento y marco teórico de dicho sistema.

Este proyecto tiene como objetivo agregar valor tecnológico al proceso de postcosecha del limón; con la finalidad de mejorar la calidad y velocidad en que se realiza dicha actividad, logrando así, aumentar la competitividad de nuestros agricultores y comerciantes en el mercado nacional y mundial.

Está conformado por un cuarto de visión, el cual cuenta con una cámara tipo webcam, encargada de obtener las imágenes, y un sistema de iluminación artificial para el correcto funcionamiento del sistema.

Por otro lado, tenemos el sistema de visión artificial en el cual se propone un conjunto de técnicas de procesamiento de imágenes que van desde binarización, morfología matemática hasta transformación del modelo de color RGB a HSV, con el fin de poder medir los diferentes parámetros utilizados. Las técnicas se implementarán en una PC mediante el uso de lenguaje Python y sus correspondientes librerías como OpenCV, que es la encargada de procesar la información de los limones tomada por la cámara y de identificar los mismos clasificándolos por color y tamaño, según lo requerido. Python es de uso libre y nos permite mejorar enormemente los costos al no ser necesario la adquisición de una licencia como ocurre con otros softwares.

Se desarrollará una interfaz gráfica a través de la herramienta Qt Designer que nos mostrará los datos obtenidos en pantalla.

Luego de ser procesada las imágenes teniendo en cuenta los resultados se enviará información vía serial a un microcontrolador el cual se encargará de controlar las diferentes salidas del sistema las cuales serán visuales mediante leds y un display lcd y otra mecánica a través de servomotores que se encargarán de mover una compuerta y así permitir la correcta separación de los diferentes tipos de limones.

## Bases Teóricas

## El Limón

El limonero, es un árbol perenne[[1]](#footnote-1) que pertenece a la familia de las Rutáceas. Su tronco puede alcanzar hasta los 6 metros de altura con gran cantidad de ramas donde se ubican espinas duras y gruesas, y presentando un porte abierto (no tan redondeado como el de otros árboles del mismo género). Sus hojas son simples, unifoliadas, elípticas, y de color verde oscuro cuando están maduras caracterizadas por un increíble aroma. Su llamativa flor de color blanco, con una leve coloración rosa, recibe el nombre de azahar y se presenta en forma aislada o en pequeños racimos. En cuanto al fruto, el limón, es un hesperidio de forma oblonga cuyo color es amarillo intenso cuando alcanza su madurez. Tiene cáscara gruesa de gran porosidad, y en su interior, la pulpa está dividida en gajos que poseen su característico jugo agrio. Sus semillas son pequeñas y puntiagudas. El limón es una fruta que presenta bajas calorías y es una fuente importante de vitamina C (ácido ascórbico), vitaminas del complejo B (B1, B2, B3, B5, B6) y ácido cítrico. Además, aporta ciertos minerales como potasio, sodio, fósforo, azufre, magnesio y calcio. Su consumo presenta varios beneficios para la salud humana, y algunos de ellos son:

• Reduce los síntomas del resfriado y acorta la duración del mismo.

• Protege al corazón, ya que es rico en flavonoides que reduce los triglicéridos.

• Tonifica los capilares sanguíneos por lo que reduce el riesgo de hemorragias.

• Favorece a los hipertensos y a quienes padecen enfermedades cardiovasculares.

• Cumple la función de bactericida y desintoxicante.

• Aumenta la absorción de calcio en los huesos.

• Es cicatrizante interno y externo.

• Ayuda a prevenir el cáncer de colon, mama, piel y estómago.

El limonero presenta muy buena capacidad de adaptación al clima y puede ser cultivado en zonas de altas temperaturas, secas o de extremada humedad. Ambientes con variaciones de temperatura diurna y nocturna favorecen la maduración de la fruta con respecto a la producción de azúcares, disminución de acidez y el desarrollo del color. En cuanto a los suelos, requiere que sean profundos con textura liviana evitando los suelos pesados ya que de ser así tendríamos problemas de infiltración y en consecuencia pudriciones a nivel de las raíces a causa de patógenos como Phytophthora parasítica, entre otros.

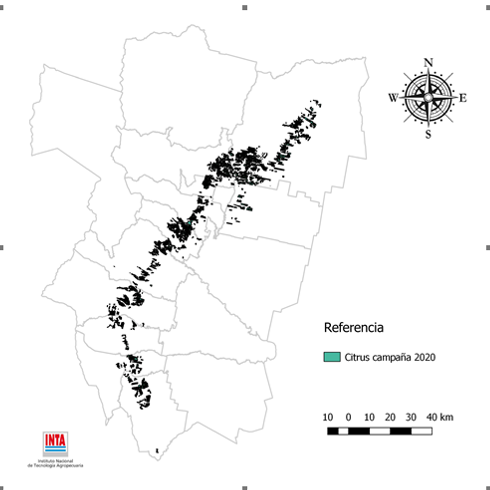


*Fig.1-* *Brotes, flores y frutos de limón[[2]](#footnote-2).*

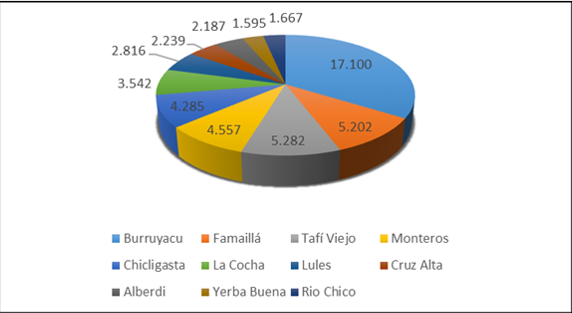
## Industria del Limón

En la Argentina podemos identificar dos regiones productoras de frutas cítricas: el NEA (Noreste Argentino) y el NOA (Noroeste Argentino). Los citrus dulces como son las naranjas, mandarinas y pomelos se producen principalmente en el NEA, región conformada por las provincias de Entre Ríos, Corrientes, Misiones y el norte de Buenos Aires. Entre las provincias del NOA, Jujuy y Salta también son productoras de citrus dulces. Sin embargo, en esta región, se destaca la provincia de Tucumán ya que su producción de dulces es escasa, pero es el principal centro productor de limón del país con más de 50.000 hectáreas destinadas al cultivo del mismo, actualmente el citrus más importante de Argentina.

Por lo dicho anteriormente, el limón se desarrolla excepcionalmente en la provincia de Tucumán donde nos encontramos con un clima subtropical. Los cítricos se distribuyen en la provincia, a lo largo de una angosta franja que se extiende desde Burruyacú, en el extremo noreste, hasta La Cocha, en el sur, limitados por las isoyetas[[3]](#footnote-3) de 800 a 1500 mm de precipitaciones, aunque hay zonas donde los niveles de precipitación anual superan dichos valores. En cuanto a la temperatura, la media anual ronda los 25 – 27 °C, habiendo una gran amplitud térmica entre el día y la noche. Esta zona descripta, corresponde al denominado Pedemonte, caracterizado por los suelos con relieve ondulado y de alta productividad.



*Fig.2 - Distribución espacial de las plantaciones citrícolas en la provincia de Tucumán.[[4]](#footnote-4)*



*Fig.3 - Superficie cultivada con citrus a nivel departamental (ha)[[5]](#footnote-5)*

Tucumán presenta muy buenas condiciones climatológicas que permiten el desarrollo del cultivo con bajos niveles de agroquímicos, lo que propicia el logro de fruta en cantidad y calidad que, además, cumpla con las exigentes normas de control y certificación para la exportación de la misma. Para lograr esto, es necesario llevar a cabo una correcta comercialización a partir de una reglamentación específica que varía según el mercado de destino (interno o externo) con el objetivo de realizar una entrega de la mercadería en tiempo y forma.

## 1.1.3 Requerimientos para el proceso de clasificación

La máquina diseñada debe tener la capacidad de separar y clasificar por medio de visión artificial las diferentes categorías de limón: EXPORTACIÓN, INSDUSTRIA LOCAL y RECHAZADOS. Estas categorías surgen teniendo en cuenta los diferentes parámetros de clasificación descriptos en “PROTOCOLO DE CALIDAD PARA LIMÓN FRESCO - Resolución SAGyP N°: 371/2015” los cuales se detallan a continuación:

Requerimientos generales

En todas las categorías los cítricos deberán estar:

• Enteros.

• Bien formados.

• Exentos de heridas y magulladuras cicatrizadas de importancia.

• Sanos, quedando excluidos los productos que presenten podredumbre u otras alteraciones que los hagan impropios para el consumo.

• Limpios, prácticamente exentos de materias extrañas visibles.

• Prácticamente exentos de plagas o de los daños que ellas causan.

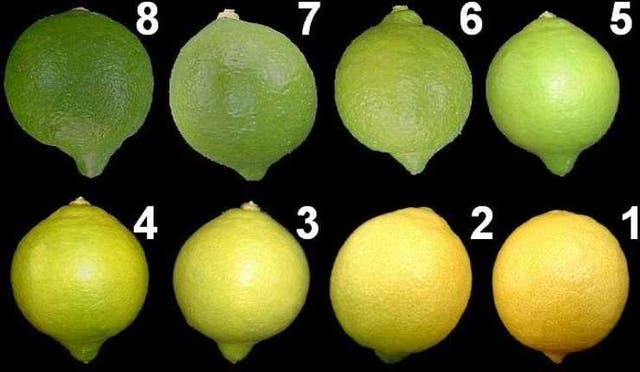
• Exentos de toda señal de desecación interna.

• Exentos de daños causados por bajas temperaturas o heladas.

• Exentos de un grado anormal de humedad exterior.

Requerimientos de color

El grado de coloración deberá ser tal que, al término de su proceso normal de desarrollo, alcancen en el lugar de destino el color típico de la variedad a la que pertenezcan en al menos el 70% del fruto. La uniformidad del color debe ser superior al 90%. Existen 8 tonalidades que se representan en la figura, siendo las 3-4-5-6- aptas para exportación, las demás si se cumplen los parámetros generales son aptas para consumo local.



*Fig.4 – Tonalidades de Limones[[6]](#footnote-6)*

Requerimientos de tamaño

El calibre se determina por el diámetro máximo de la sección ecuatorial del fruto de acuerdo con la siguiente tabla:

|  |  |
| --- | --- |
| Calibre | Escala de diámetro (mm) |
| 0 | 79-90 |
| 1 | 72-83 |
| 2 | 68-78 |
| 3 | 63-72 |
| 4 | 58-67 |
| 5 | 53-62 |
| 6 | 48-57 |
| 7 | 45-52 |

*Fig.5 – Tabla de Diámetros[[7]](#footnote-7)*

Nota: Limones con calibre menor a 45mm serán separados para industria local.

Requerimientos de Velocidad

La velocidad de funcionamiento del equipo debe ser adecuada, para que permita una clasificación mínima del producto. Si se realizase una separación de manera manual un operario calificado puede clasificar aproximadamente 40 limones por minuto, lo que sería menos de 1 limón por segundo. Se estima que la velocidad debe ser de entre 10 a 19 limones por segundo para garantizar una cantidad correcta de producción. Esto se calcula en base a que en 1kg hay aproximadamente 11limones (dependiendo de los calibres mencionados variará su peso, esta estimación se hace con un peso promedio de 90g por limón) teniendo en cuenta esto nos da que:

1s = 10 limones

1hora = 36.000 limones

11 limones = 1Kg

36.000 limones = 3.27 Tn

## 1.1.4 Visión Artificial

También conocida como visión por computador (del inglés computer vision) o visión técnica, es un subcampo de la inteligencia artificial y es el campo de acción más ambicioso del procesamiento digital de imágenes. Básicamente el objetivo es automatizar funciones de inspección visual, tradicionalmente utilizadas por el hombre.

Los objetivos típicos de la visión artificial incluyen:

La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes (por ejemplo, caras humanas), la evaluación de los resultados (ej.: segmentación, registro), registro de diferentes imágenes de una misma escena u objeto, hacer concordar un mismo objeto en diversas imágenes, seguimiento de un objeto en una secuencia de imágenes, entre otros.

## 1.1.5 Procesamiento de Imágenes

El procesamiento de imágenes es un conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información.

## 1.1.6 La Imagen Digital

Son el principal ingrediente de lo que se conoce como Visión Artificial y representan mediante algún tipo de codificación, normalmente en una matriz de números de dos dimensiones, una escena del entorno.

Existen dos tipos de imágenes utilizadas frecuentemente en Visión Artificial: imágenes de intensidad e imágenes de alcance (también llamadas imágenes de profundidad o perfiles de superficie). Las imágenes de intensidad miden la cantidad de luz que incide en un dispositivo fotosensible, mientras que las imágenes de alcance estiman directamente la estructura en tres dimensiones (3D) de la escena ya que su fundamento radica en el uso de sensores de alcance ópticos y algún fenómeno físico para adquirir la imagen. Un ejemplo típico de una imagen de intensidad es una fotografía, mientras que de una imagen de alcance es, por ejemplo, la imagen que obtiene el oftalmólogo sobre el grado de rugosidad de la córnea de un paciente o las imágenes de un radar.

# Capítulo 2 – Desarrollo

## 2.1 Software Utilizado

### 2.1.1 Python

Es un lenguaje de programación interpretado de alto nivel, interactivo y orientado a objetos. El código fuente está disponible bajo Licencia Publica General de GNU. La sintaxis de Python es muy limpia, con énfasis en la legibilidad y utiliza palabras clave estándar en inglés.

Python fue creado a finales de la década de los años 80 por un europeo llamado Guido Van Rossum en los Países Bajos. El objetivo de Guido era cubrir la necesidad de un lenguaje orientado a objetos de sencillo uso que sirviese para tratar diversas tareas dentro de la programación que habitualmente se hacía en Unix usando C.

Su formato no utiliza corchetes para delimitar bloques, sino que son las tabulaciones y los espacios los que definen en que bloque se encuentra una instrucción. Una de sus características es el uso de palabras donde otros lenguajes utilizarían símbolos. Por ejemplo, los operadores lógicos ‘!’, ‘||’ y ‘&&’ en Python se escriben ‘not’, ‘or’ y ‘and’, respectivamente.



*Fig.6 – Logo de Python[[8]](#footnote-8)*

## 2.1.2 Librerías

### 2.1.2.1 OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de maching learning y visión por computadora de código abierto. Al ser un producto con licencia BSD, OpenCV facilita a las empresas a utilizar y modificar el código.

La biblioteca fue lanzada oficialmente en 1999 y posee más de 2000 algoritmos optimizados que se pueden utilizar para detectar y reconocer colores, caras, identificar objetos, clasificar acciones humanas en videos, rastrear movimientos de cámara, etc.

Es actualmente utilizado por grandes compañías como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda y Toyota.



*Fig.7 – Logo de OpenCV [[9]](#footnote-9)*

2.1.2.2 Numpy

Es una librería especializada en el cálculo numérico y análisis de datos. Incorpora una nueva clase de objetos llamadas **arrays,** que permite representar colecciones de datos de un mismo tipo por ejemplo números.



*Fig.8 – Logo de Numpy [[10]](#footnote-10)*

### 2.1.2.3 PyQT

Es una biblioteca que nos permite la conexión entre la interfaz gráfica desarrollada en el software QT Designer y el código desarrollado en Python.

2.1.2.4 PySerial

Es una librería que nos permite realizar comunicación con el puerto serie de una manera simple.



*Fig.9 – Logo de Py Serial* [[11]](#footnote-11)

## 2.1.3 QT Desingner

Es una herramienta para diseñar y crear interfaces gráficas de usuario (GUI), permite componer y personalizar ventanas y cuadros de diálogos de manera *“what-you-see-is-what-you-get”* (Lo que vemos es lo que obtenemos).



*Fig.10 – Logo de QT [[12]](#footnote-12)*

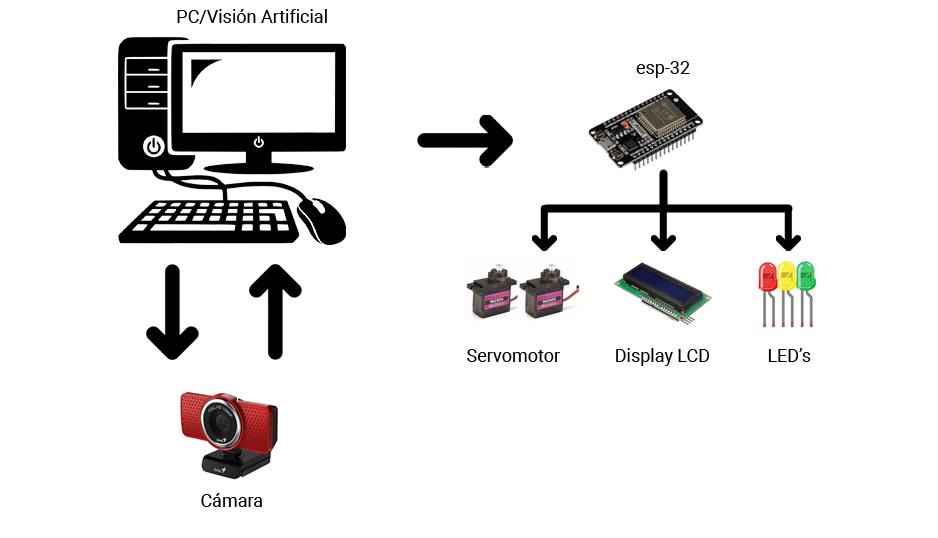
## 2.1.4 Visual Studio Code

Es un potente editor de código gratuito y de código abierto, desarrollado por Microsoft y lanzado en 2016, mediante sus extensiones permite convertirlo en un entorno de desarrollo que soporta múltiples lenguajes de programación como Python, Java Script, C, etc. Entre sus características más destacadas se puede mencionar la finalización de código inteligente, que nos permite agilizar la escritura de código, depuración optimizada, resaltado de sintaxis, entre otras.



*Fig.11 – Logo de Visual Studio Code[[13]](#footnote-13)*

## 2.2 Esquema general del Proyecto



## *Fig.12 – Esquema General*

## 2.3 Hardware Utilizado

## 2.3.1 Cámara

En nuestro proyecto para la captura de imágenes se utilizará una cámara web la cual nos permite una conexión con la pc de manera directa o inalámbrica, y hace posible la visualización remota. Debido a la buena relación calidad – precio elegimos la siguiente cámara:

* **Cámara Genius ECam 8000**

Que cuenta con las siguientes características:

• Resolución máxima 1920\*1080

• Sensor CMOS

• Conexión USB 2.0

• Enfoque automático

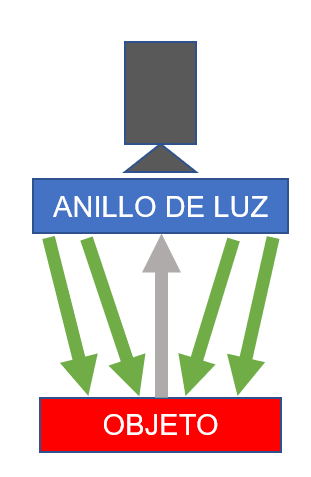
• 30 fotogramas por segundo



*Fig.13 Cámara Genius ECam 8000*

## 2.3.2 Sistema de iluminación

Para obtener buenos resultados en la adquisición de imágenes, un parámetro muy importante es la iluminación. Una adecuada iluminación permitirá resaltar los rasgos de interés del objeto, reducir la complejidad de la imagen a analizar y mejorar el tiempo de respuesta del procesamiento digital. Para el presente proyecto utilizamos iluminación LED de 5 V, nos permite mantener la luz emitida o varia muy poco con las horas de funcionamiento y de la temperatura. Dado a que la posición de la cámara con respecto al objeto a analizar, se encuentra de forma vertical, la mejor ubicación para generar una iluminación adecuada sería posicionar la fuente de luz en la misma dirección de la cámara. Esto reduce las sombras que se podrían generar al pasar el fruto por el cuarto de visión, también suaviza las texturas y minimiza la influencia de rayas, polvo e imperfecciones que pueda tener el fruto. La cámara recibe la luz reflejada del fruto.



*Fig.14 - Sistema de iluminación.*

Fuente: Elaboración propia.

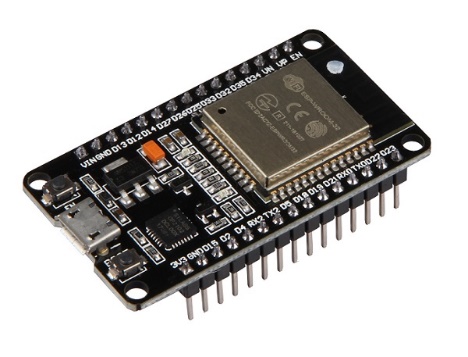
## 2.3.3 ESP32

El microcontrolador elegido será el ESP32, el cual es un microcontrolador de 32bits que nos permite la interconexión entre nuestro sistema desarrollado en Python y los demás componentes como servos, display, y leds. Además, posee otras características que pueden ser útiles para futuras modificaciones como lo es la conexión de Wi-Fi/Bluetooth todo en uno, integrada y certificada que proporciona no solo la radio inalámbrica, sino también un procesador integrado con interfaces para conectarse con varios periféricos. El procesador en realidad tiene dos núcleos de procesamiento cuyas frecuencias operativas pueden controlarse independientemente entre 80 megahercios (MHz) y 240 MHz. Los periféricos del procesador facilitan la conexión a una variedad de interfaces externas como:

* Interfaz periférica serial (SPI)
* I2C
* Transmisor receptor asíncrono universal (UART)
* I2S
* Ethernet
* Tarjetas SD
* Interfaces táctiles y capacitivas

Características del módulo utilizado:

* Voltaje de Alimentación: 5V (USB)
* Voltaje de Entrada / Salida: 3.3V
* SoC: ESP32
* CPU principal: Tensilica Xtensa 32-bit LX6
* Frecuencia de Reloj: hasta 240Mhz
* Desempeño: Hasta 600 DMIPS
* WiFi: 802.11 b/g/n/e/i
* Bluetooth:v4.2 BR/EDR y Bluetooth Low Energy (BLE)
* Pines Digitales GPIO: 24 (Algunos pines solo como entrada)
* Convesor Analogolico Digital
* UART: 2
* Chip USB-Serial: CP2102
* Antena en PCB
* 38 PINES
* DIMENSIONES: 48 x 25,5 mm (Largo y Ancho)



*Fig.15 – Microcontrolador ESP32*

## 2.3.4 Servomotor

Los actuadores encargados de realizar la separación del producto en distintas posiciones son un par de servomotores, que, en conjunto a un par de paletas mecánicas, nos darán la posibilidad de separar los limones según su clasificación.

Teniendo en cuenta dimensiones y peso del cítrico además de la velocidad de procesamiento del sistema, se utilizan servomotores con las siguientes características:

* Micro Servo MG90S
* Dimensiones: 23x12,2x29mm
* Velocidad de funcionamiento (4,8 V sin carga): 0,11seg/60 grados
* Velocidad de funcionamiento (6,0 V sin carga): 0,10 seg/60 grados
* Puesto Par (4,8 V): 2,0 kg/cm
* Puesto Par (6,0 V): 2,5 kg/cm
* Rango de temperatura: 0 a 55 grados C
* Muerto Ancho de banda: 15 microsegundos
* Voltaje de funcionamiento: 4,8 – 6 V

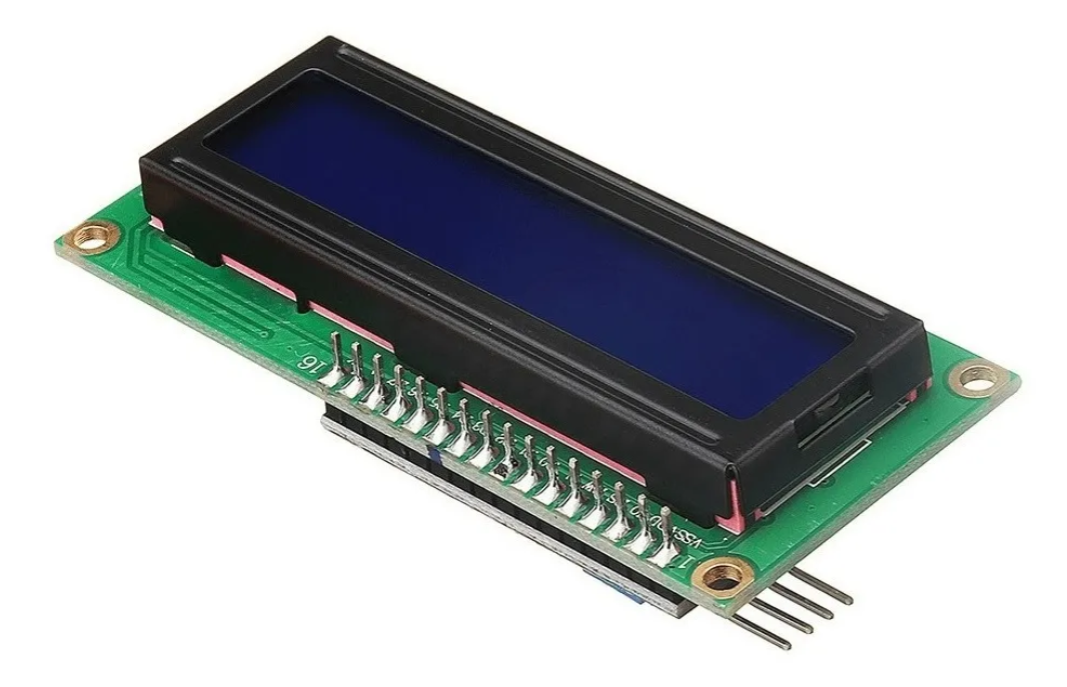


*Fig.16 - Servomotores*

## 2.3.5 Display

Además de la visualización de datos en la computadora a través del sistema, también podremos visualizar los mismos en un display LCD que cuenta con las siguientes características:

* Modelo: LCD 1602 Hd44780
* Voltaje de alimentación: 4,5v – 5,5v DC.
* Formato de presentación: 16x2 caracteres.
* Modo de operación: 4 u 8 bits.
* Color fondo: Azul.
* Color caracteres: Blanco.



*Fig.17 – Display LCD*

## 2.4 Software Implementado

## 2.4.1 Conocimientos Previos

Antes de empezar el desarrollo del software fue necesario aprender algunos conceptos previos. En un principio la programación básica en lenguaje Python y luego conceptos más complejos como la programación orientada a objetos(POO) y el uso de “hilos”

2.4.1.1 Programación básica

Algunas de los conceptos básicos de Python que usamos son los siguientes:

**Variable:** Nombre simbólico que apunta a un valor u objeto específico.

**Operadores Aritméticos:** Suma (+), resta (-), multiplicación (\*), división (/), división entera (//), módulo (%).

**Tipos de datos:** Numéricos (enteros, flotantes, complejos), Secuencias (cadenas, listas, tuplas), Booleanos (Verdadero, Falso), Diccionarios y Conjuntos.

**Expresiones Booleanas:** Expresiones en las que el resultado es True o False.

**Condicional:** Evalúa una expresión booleana y realiza algún proceso dependiendo del resultado. Se maneja mediante sentencias if/else.

**Bucle:** Ejecución repetida de bloques de código. Pueden ser bucles for o while.

**Funciones:** Bloque de código organizado y reutilizable. Se crean con la palabra clave def.

2.4.1.2 Programación orientada a objetos

La programación orientada a objetos es un paradigma de la programación donde podemos pensar los problemas complejos como objetos.

Un objeto en Python es un conjunto de datos (atributos) y comportamientos (métodos). Se puede relacionar a los objetos como cosas reales que nos rodean para entender mejor el concepto, por ejemplo: considerando como objeto un perro los atributos serían nombre, color, raza, etc. y los métodos ladrando, olfateando, corriendo, etc. Los atributos son siempre sustantivos mientras que los métodos son siempre verbos.

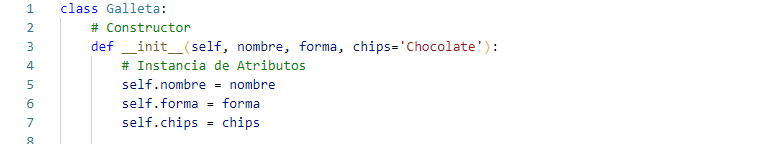
Clase: es el modelo sobre el cual nuestros objetos se construyen, las clases permiten generar más objetos basados en los atributos y métodos que se le define. Para definir una clase se utiliza la palabra clave **class** seguido de su nombre.

Nota: En Python, se utiliza la convención de nombres en mayúsculas para nombrar las clases.

Instancia: es un objeto individual de una clase que tiene una dirección de memoria única

El método \_\_init\_\_() también llamado constructor. Es utilizado por Python cada vez que instanciamos un objeto. El constructor se encarga de crear el estado inicial del objeto con un conjunto mínimo de parámetros que necesita para existir.

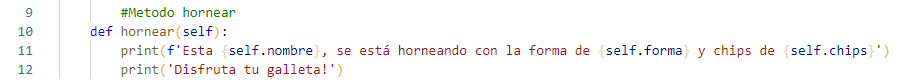
Ejemplo



En la clase Galleta, cada galleta debe tener un nombre, una forma y un tipo de chips. Hemos definido esta última como “Chocolate”.

Por otro lado, self se refiere a la instancia de la clase (el objeto en sí).

En el siguiente paso vamos a añadir el método hornear() como ejemplo, para mostrar su funcionamiento. El método hornear imprimirá por consola una frase teniendo en cuenta los atributos de la clase que definamos.



Para llamar a un método, primero se define una variable, la cual contendrá la clase y los correspondientes atributos que s ele quiera asignar al objeto creado, luego se utiliza la notación de punto y se lo invoca como una función.



Luego de ejecutar el programa veríamos en consola lo siguiente



### 2.4.2 Captura de Imágenes

En este punto nos centraremos en la captura de imágenes provenientes de la cámara para su posterior análisis y clasificación.

Para nuestro propósito el código hace uso de la librería OpenCV, la cual mediante la función *VideoCapture(0)* inicia la cámara para tomar una imagen (0 es el número de cámara en caso de tener más de una conectada), dicha imagen se almacena en una variable y la función *read* nos devuelve el frame capturado.





*Fig.18 – Captura de Imagen*

2.4.3 Captura de Video

Primer recordemos que un video no es más que un conjunto de imágenes en un determinado tiempo, generalmente 30 por segundo. Teniendo en cuenta esto para realizar la captura continua de video se pondrá las funciones anteriormente mencionadas en un siclo while infinito que permite la lectura continua de imágenes.



2.4.4 Selección del área a analizar

Utilizando la función *np.array,* la cual pertenece a la librería Numpy y nos permite crear arreglos de números, definiremos 4 puntos que ordenados de forma horaria formaran nuestra región de interés sobre la cual aplicaremos posteriormente las funciones que nos permitirán realizar el procesamiento.

****

2.4.5 Detección de Color

Para realizar la detección de color lo primero que se debe realizar es la conversión del espacio de color.

Por defecto OpenCV carga las imágenes en un espacio de color BGR (Blue, Green, Red), es necesario transformar las imágenes a un espacio HSV (Hue, Saturation, Value / Matiz, Saturación, Brillo).

Utilizaremos el espacio HSV debido a que nos permite identificar de forma más sencillas los colores. Para determinar un color nos centraremos en el componente H que corresponde a la Matiz, en la siguiente imagen podemos apreciar que el componente H va desde el rojo, pasando por el verde y azul hasta llegar al rojo nuevamente.



*Fig.19 – Espacio de color HSV [[14]](#footnote-14)*

Además, para trabajar con estén entorno de color es necesario saber que valores puede tomar cada componente en OpenCV:

H: 0 a 179

S: 0 a 255

V: 0 a 255

La función que nos permite realizar la transformación del espacio de color es *cv2.cvtColor* la cual recibe 2 parámetros primero la imagen a transformar el segundo entre que espacio de color se realizará la trasformación.

****

* Determinar los rangos donde se encuentra los colores a detectar

Los colores que nos interesan detectar son el amarillo y verde, los cuales son los colores característicos del limón, la presencia de otros colores como podría ser el marrón correspondería a alguna anomalía.



*Fig. 20 – Componentes HSV* [[15]](#footnote-15)

Como se observa en la figura 6 el componente H va desde 0 a 179, luego de diferentes pruebas se llegó a determinar los siguientes rangos:

Para el color Amarillo el rango en H va de 22 a 33, el de S va de 21-255 y V 0 a 255.

Para el color Verde el rango en H va de 35 a 89, el de S va de 50-255 y V 0 a 255.

Para Defecto el rango en H va de 0 a 20, el de S va de 150-255 y V 80 a 255.

Estos valores correspondientes a los diferentes rangos de colores son creados como arreglos de números mediante la función *np.array* y almacenados en variables.

****

Para buscar los rangos de colores en nuestra imagen se utiliza la función *cv2.inRange,* la cual recibe como parámetros la imagen en la cual se buscará los rangos, y los arreglos correspondientes al límite inicial y final del color que nos interesa.

****

* Consideraciones que se deben tener en cuenta

Se debe tener en cuenta dos aspectos muy importantes a la hora de aplicar la detección de colores:

**Iluminación:** es importante tener controlado este aspecto ya a que a más o menos iluminación presente sobre el objeto, puede variar el resultado de la detección.

**Fondo:** el fondo también es un aspecto a tener en cuenta debido a que si en este se encuentran presentes alguno de los colores dentro de los rangos no solo se detectaría nuestro objeto de interés sino también será detectado parte del fondo. Se recomienda en lo posible usar un fondo uniforme y que no presente ningún color de interés.

2.4.6 Detección de contornos

Una vez que se tiene identificado los grupos de pixeles que nos interesan mediante la función *cv2.findCountours* encontramos los contornos que forman el área de estos grupos, luego es necesario analizar uno a uno los contornos para asegurarnos que se traten de limones. Los contornos de un objeto se pueden explicar simplemente como una curva que une todos los puntos continuos (a lo largo del límite o borde) que tienen el mismo color o intensidad.

****

Procederemos a analizar cada contorno contenido en la variable contornos, compararemos si el área del contorno es lo suficientemente grande para ser considerado un limón. Para ellos se compara el área en pixeles con 1500, de modo que si el área es mayor a este valor se continua con el procesamiento de lo contrario lo descartaremos.



2.4.7 Encontrar Centro y diámetro

Al aplicar la función *cv2.boundingReact* podemos encontrar 4 puntos útiles, las coordenadas X, Y y los valores W y H que corresponden al ancho y altura respectivamente. Estos valores nos serán útiles después para poder encontrar el diámetro, contar y dibujar un rectángulo al contorno de nuestro objeto.



Manteniendo fija la distancia de la cámara al objeto, para hallar el diámetro de cada limón se calcula el promedio del ancho (W) y la altura (H). Este dato esta expresado en función de numero de pixeles, para convertirlo en milímetros lo dividiremos por un factor. Este factor lo estableceremos mediante pruebas, se calculará el diámetro mediante el software y se comparará el valor con una medición real del limón en milímetros, con el promedio de todas las mediciones realizadas se establecerá una relación que nos permite definir el factor.



|  |  |
| --- | --- |
| Diámetro Referencia [Px] | Diámetro Real [mm] |
| 276 | 56 |
| 232 | 51,5 |
| 261 | 60 |
| 240 | 59 |
| 270 | 58 |
| 239 | 50 |
| 262 | 57 |
| 296 | 62 |
| 220 | 50 |
| 246 | 59 |
| **PROMEDIO 254,2** | **PROMEDIO 56,25** |
| **FACTOR 4,52** | |

Dándole los parámetros obtenidos con boundingRec a la función *cv.rectangle* obtendremos un rectángulo al contorno de nuestro objeto, este rectángulo nos será útil para entender mejor el proceso de conteo de limones.

2.4.8 Contador de Limones

Para poder contar los limones, tomaremos el punto “Y” del recuadro delimitador antes mencionado y veremos si está presente entre los pixeles 300 y 400, si es así el limón será contado haciendo que la línea amarilla central se torne verde y las variables destinadas a llevar las cuentas de los limones aumentaran.



*Fig. 21 – Contador Limones*

Es importante mencionar que los contadores podrían fallar si un limón se desplaza demasiado lento, lo ideal sería que la velocidad se mantenga constante. Luego de pruebas y error se llegó a determinar cuál sería la velocidad óptima para los rangos determinados.

Tanto el rectángulo del contorno como la línea verde del contador son solamente con fines visuales para una mejor apreciación del proceso, podrían ser excluidas del código sin modificar su funcionamiento.

2.4.9 Comunicación Serial

La comunicación con el microcontrolador se realizará por serial, la librería PySerial es la encargada de establecer esta comunicación. En primer lugar se deberá inicializar la comunicación, esto se hace con la función *serial* que recibe como parámetros el puerto donde esta conectado nuestro dispositivo y la velocidad de comunicación.



Después de haber realizado el correspondiente procesamiento a la imagen se usa la función *write* para iniciarr la conexión entre Python y el microcontrolador. Esta función se encarga de imprimir el contenido de la variable “la” (‘A’) en el puerto serie lo cual desencadenará una serie de funciones en el microcontrolador las cuales se verán a detalle en la próxima sección.



2.4.10 Software del Microcontrolador

La programación del microcontrolador se realiza en el lenguaje C++, basándonos en los principios de la programación orientada a objetos, podemos separar en 3 grupo principales por un lado los servos, en segundo lugar el display y por último los leds.

Para el desarrollo del programa nos ayudaremos con las siguientes librerías:



Vamos a inicializar nuestros objetos, en primer lugar, se definen los leds, mediante variables son asignados a los pines 12, 13 y 14 de nuestro Esp32.



En segundo lugar, vamos a crear el objeto servo y asignarlo al pin 26 para su posterior control



Y por último vamos a definir el objeto display el cual tendrá una comunicación con el microcontrolador por i2c en la dirección 0x27. Aquí también se definen los parámetros de cuantas filas y columnas tendrá nuestro display en este caso es de 16x2



Después de tener nuestros objetos definidos e inicializados se procede a establecer las configuraciones (estas instrucciones solo se realizan una vez cuando es encendido el microcontrolador).

Lo primero es establecer una comunicación serial la cual nos permite tener la conexión con el software desarrollado en Python. Es importante que en el desarrollo de ambos programas se utilice la misma velocidad de trasmisión (por ejemplo 9600).



Después configuramos nuestros pines como salidas digitales para poder realizar el encendido y apagado de los leds



Seguido iniciamos el servomotor y lo ponemos en la posición 0



Y por último encendemos y mostramos un primer mensaje de bienvenida en el display



Continuando con nuestro programa entramos a definir lo que se ejecutara indefinidamente en nuestro loop o lazo infinito.

En este caso lo que hace el código es leer constantemente el puerto serial y de acuerdo a lo que haya imprimido el código en Python realizara una determinada secuencia de acciones.

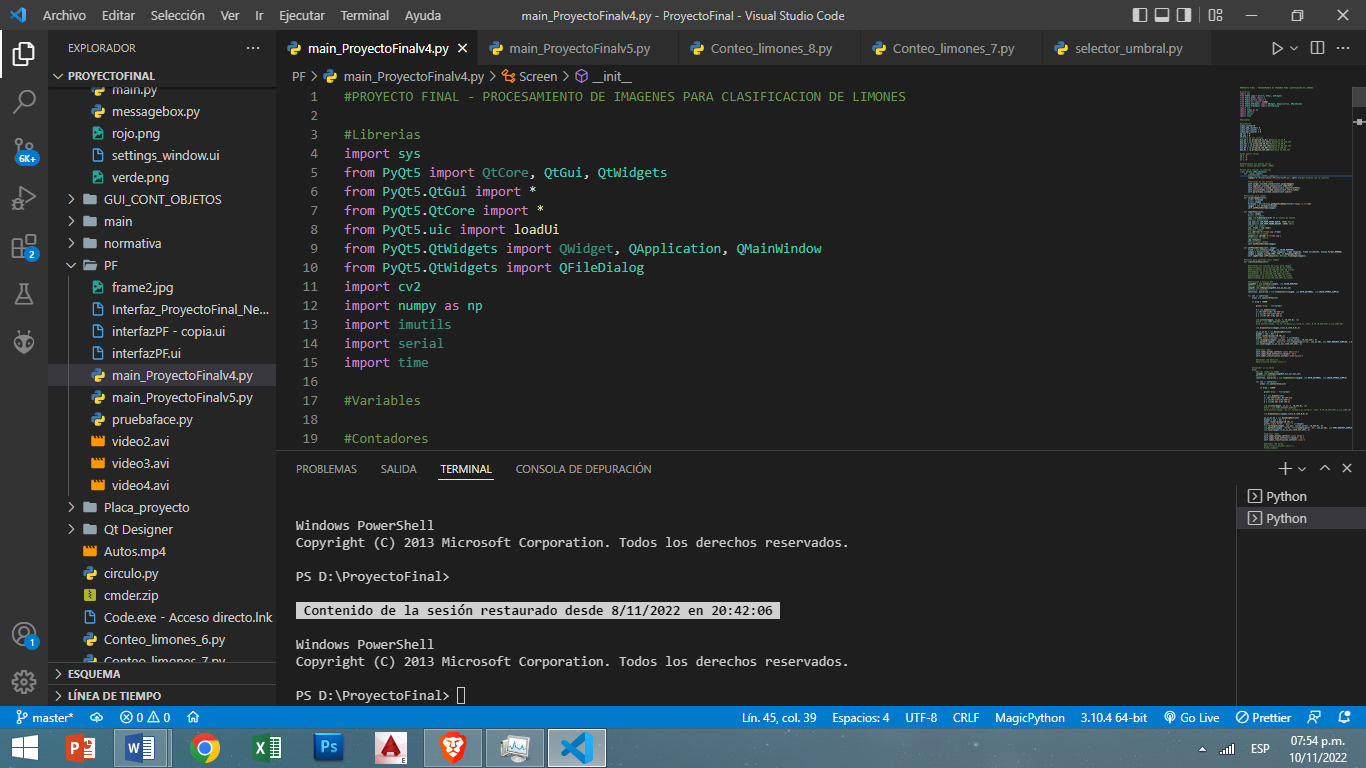
Por ejemplo, para el caso de que el programa nos haya enviado un carácter ‘A’ se ejecutaran las acciones para un limón amarillo las cuales son: encender led amarillo, mover servo 180° e imprimir en el display un mensaje de que el limón es aceptado para exportación, esto después se replica para los casos en el que el limón es verde o rechazado.



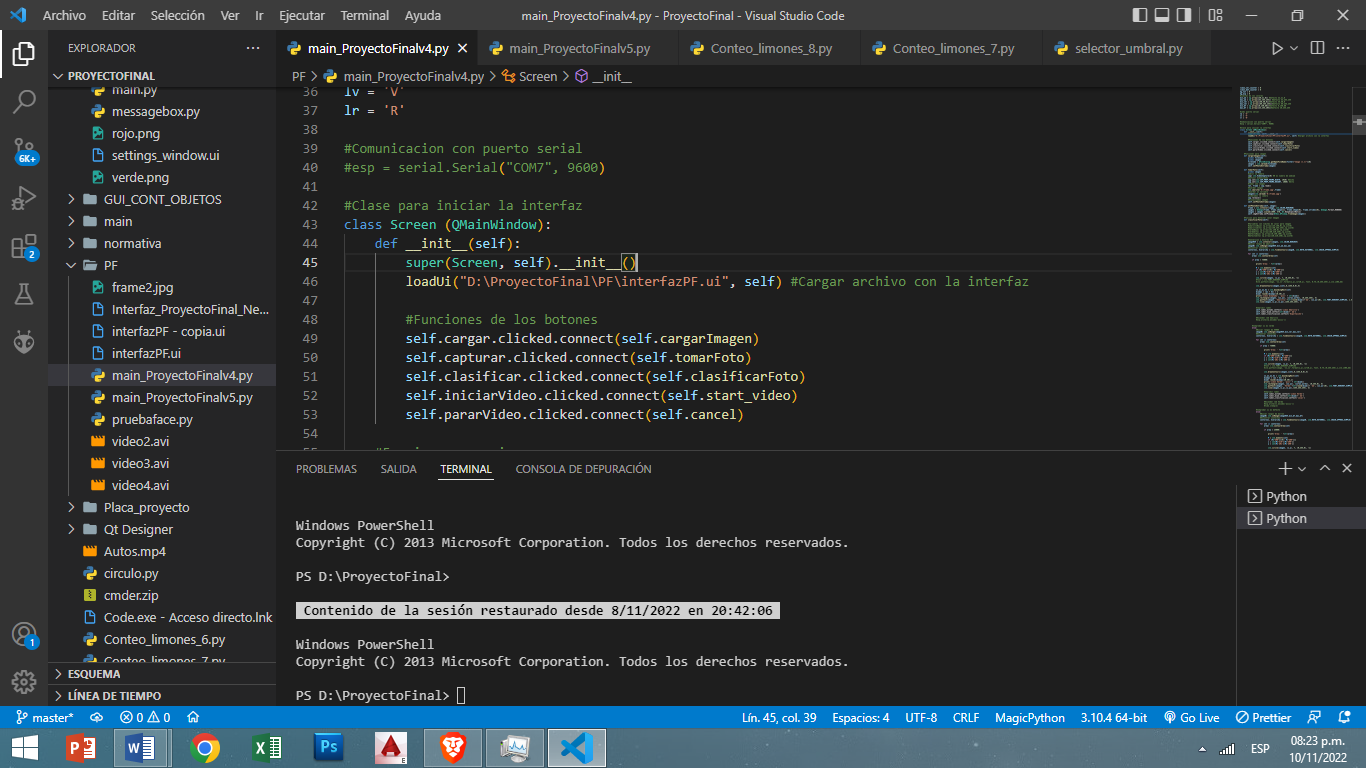
2.4.11 Interfaz grafica

Una buena practica es tener separado el código de la interaz grafica esto debido a qe si se tiene todo en n solo archivo se haría muy complicado corregir errores ya que tendríamos demasiadas funciones, otro aspecto interesante es que cuando se lo tiene en un mismo archivo no se puede moiicar libremente la interfaz grafica las veces que sean necesaria. Para evitar estos inconvenientes a nestro proyecto s elo organia en dos archivos uno con extención “.py” que contiene todas la funciones y otro con extencios .ui qe contiene la interfaz grafica y que puede ser reemplazado las veces que sea necesario luego de editarlo con QtDesing.

Luego la conexión de los dos archivos se realiza importando las correspondientes librerías de qt designer en Python



la clase UI\_MainWindow que es la que contiene las propiedades de nuestro diseño (el que realizamos en QtDesigner). Dicha clase se modifica cada vez que cambiamos nuestro diseñoy cuando cambiemos el diseño se importara la clase con el nuevo código y obtendremos el nuevo diseño



vamos a colocarla como herencia dentro de nuestra clase **Screen** que hemos creado al principio! Quedando así:

## 2.5 Diagrama de flujo del proceso de clasificación





## Diagrama de flujo del microcontrolador

## Cuarto de visión

El cuarto de visión es una de las piezas más importantes que posee la máquina clasificadora de limón, ya que por medio de este van a pasar los diferentes tipos. En el mismo se encuentra incorporado una cámara que permite detectar si el limón posee defectos o no para su posterior clasificación. La distancia a la que debe ser ubicada dicha cámara depende del tamaño del fruto, del área que deseamos capturar en las imágenes, teniendo en cuenta que, a una mayor distancia, capturamos un área mayor, área que al momento del procesamiento podría llevar un mayor tiempo en el análisis y no tener información relevante. Para asegurarnos que la cámara capture toda el área deseada, tomando en cuenta el tamaño y resolución del sensor, hemos ubicado la cámara a una distancia de 22 cm del contenedor del fruto. Esta distancia la hemos determinado asumiendo un tamaño promedio de limón.

Para el diseño del cuarto de visión, se analizaron y probaron diferentes medidas del mismo, teniendo en cuenta las características de nuestra cámara. Se llegó a la conclusión de que éste debe tener las siguientes dimensiones:

* Largo: 250 mm
* Ancho: 180 mm
* Alto: 220 mm

Con respecto a la abertura por la cual pasarán los cítricos para ser capturados por la cámara, teniendo en cuenta que el diámetro del limón con mayor calibre (clase 0) es igual a 90 mm, se optó por definir a la misma con las siguientes dimensiones:

* Ancho: 120 mm
* Alto: 90 mm

Para el armado, se utilizó madera MDF laminada (melamina) color blanco liso de 18 mm de espesor.

Cabe aclarar que las medidas detallas anteriormente están analizadas con respecto al vano interior del cuarto de visión, y teniendo en cuenta el espesor de la madera, las medidas finales del mismo quedan dispuestas de la siguiente manera:

El resultado final del cuarto de visión es el siguiente:

**Foto del cuarto de visión**

Placa del proyecto

La placa del proyecto fue realizada con el software PCB Wizard de la siguiente manera:

Resultado Final de la placa:

Resultados

Los resultados obtenidos al realizar este proyecto fueron satisfactorios ya que se

cumplió el objetivo principal de detectar patentes y los caracteres que las componen,

conseguir un correcto sensado de los lugares para determinar si un vehículo se

encuentra estacionado o no, y lograr comunicar todas las unidades incluyendo el cartel

de forma inmediata y sin errores.

Conclusiones

1. Perenne: es una planta que tiene un follaje que permanece verde y funcional durante más de una temporada de crecimiento. [↑](#footnote-ref-1)
2. <https://es.wikipedia.org/wiki/Citrus_%C3%97_limon> 17-10-22 [↑](#footnote-ref-2)
3. Isoyeta: o isohieta es una isolínea que une los puntos en un plano cartográfico que presentan la misma precipitación en la unidad de tiempo considerada. [↑](#footnote-ref-3)
4. Fuente: INTA (2020). [↑](#footnote-ref-4)
5. Fuente: INTA (2020). [↑](#footnote-ref-5)
6. Fuente: Elaboración Propia [↑](#footnote-ref-6)
7. Fuente: Elaboración propia [↑](#footnote-ref-7)
8. https://www.python.org/ [↑](#footnote-ref-8)
9. https://opencv.org/ [↑](#footnote-ref-9)
10. https://numpy.org/ [↑](#footnote-ref-10)
11. https://pythonhosted.org/pyserial/ [↑](#footnote-ref-11)
12. https://www.qt.io/ [↑](#footnote-ref-12)
13. https://code.visualstudio.com/ [↑](#footnote-ref-13)
14. <http://omes-va.com/deteccion-de-colores/> 17-09-2022 [↑](#footnote-ref-14)
15. <http://omes-va.com/deteccion-de-colores/> 17-09-2022 [↑](#footnote-ref-15)