Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO DE UN PROTOTIPO UTILIZANDO LA TECNOLOGÍA RASPBERRY-PI PARA EL CONTROL DE LUBRICACIÓN EN FAJAS TRANSPORTADORAS EN ENVASES PET

Selvin Iván Mejía Álvarez
Asesorado por el Ing. Carlos Eduardo Guzmán Salazar

Guatemala, marzo de 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE UN PROTOTIPO UTILIZANDO LA TECNOLOGÍA RASPBERRY-PI PARA EL CONTROL DE LUBRICACIÓN EN FAJAS TRANSPORTADORAS EN ENVASES PET

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

SELVIN IVÁN MEJÍA ÁLVAREZ
ASESORADO POR EL ING. CARLOS EDUARDO GUZMÁN SALAZAR

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO ELECTRÓNICO

GUATEMALA, MARZO DE 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Christian Moisés de la Cruz Leal
VOCAL V	Br. Kevin Vladimir Armando Cruz Lorente
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANA	Inga. Aurelia Anabela Cordova Estrada
EXAMINADOR	Ing. Byron Odilio Arrivillaga Méndez
EXAMINADOR	Ing. Hugo Leonel Tiul Valenzuela
EXAMINADORA	Inga. Ingrid Salomé Rodríguez de Loukota
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE UN PROTOTIPO UTILIZANDO LA TECNOLOGÍA RASPBERRY-PI PARA EL CONTROL DE LUBRICACIÓN EN FAJAS TRANSPORTADORAS EN ENVASES PET

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 15 de julio de 2020.

Selvin Iván Mejía Álvarez

Guatemala, 20 de julio 2020

Ingeniero
Julio Solares Peñate
Coordinador Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Ingeniero Solares:

Hago de su conocimiento por este medio que he concluido la revisión del trabajo de graduación del estudiante Selvin Iván Mejía Álvarez, titulado:

DISEÑO DE UN PROTOTIPO UTILIZANDO LA TECNOLOGÍA RASPBERRY-PI PARA EL CONTROL DE LUBRICACIÓN EN FAJAS TRANSPORTADORAS EN ENVASES PET

El cual cumple plenamente el propósito para el que fue concebido. Por lo que, en mi calidad de ASESOR nombrado por la Escuela de Ingeniería Mecánica Eléctrica, doy mi aprobación al mismo. Indicando que tanto el suscrito como el estudiante Mejía Álvarez somos responsables por el contenido del trabajo referido.

Reciba un cordial saludo,

Carlos Guzmán Salazar
ASESOR

CARLOS GUZMAN SALAZAR
Ingeniero Electricista
Col. No. 2762

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Guatemala, 24 de agosto de 2020

Señor Director
Armando Alonso Rivera Carrillo
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC

Estimado Señor Director:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado **DISEÑO DE UN PROTOTIPO UTILIZANDO LA TECNOLOGÍA RASPBERRY-PI PARA EL CONTROL DE LUBRICACIÓN EN FAJAS TRANSPORTADORAS EN ENVASES PET,** desarrollado por el estudiante **Selvin Iván Mejía Álvarez**, ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

Ing. Julio César Solares Peñate
Coordinador de Electrónica

ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

REF. EIME 014.2020.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante **Selvin Iván Mejía Álvarez**, titulado: **DISEÑO DE UN PROTOTIPO UTILIZANDO LA TECNOLOGÍA RASPBERRY-PI PARA EL CONTROL DE LUBRICACIÓN EN FAJAS TRANSPORTADORAS EN ENVASES PET,** procede a la autorización del mismo.

Ing. Armando Alonso Rivera Carrillo

Guatemala, 2 de febrero de 2021.

Decanato
Facultad de Ingeniería
24189101- 24189102

DTG. 098.2021.

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Eléctrica, al Trabajo de Graduación titulado: **DISEÑO DE UN PROTOTIPO UTILIZANDO LA TECNOLOGÍA RASPBERRY-PI PARA EL CONTROL DE LUBRICACIÓN EN FAJAS TRANSPORTADORAS EN ENVASES PET,** presentado por el estudiante universitario: **Selvin Iván Mejía Álvarez,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANA
FACULTAD DE INGENIERÍA

Inga. Anabela Cordova Estrada
Decana

Guatemala, marzo de 2021.

AACE/asga

ACTO QUE DEDICO A:

Dios	Por estar presente durante mi vida académica.
Mis padres	Por haberme apoyado en todo momento a alcanzar mis metas, y por haberme dado la oportunidad de tener una excelente educación en el transcurso de mi vida.
Mis hermanos	Por ser parte importante de mi vida, gracias por el apoyo y amistad.
Mis amigos	Alejandro Pérez, Daniel Chacón, Ervin López, Guillermo López y Kevin Monroy, por haberme apoyado durante mi carrera y haber hecho de mi vida universitaria una experiencia que no olvidaré.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser la casa de estudios que inculcó en mí la responsabilidad, el trabajo y la dedicación.
Facultad de Ingeniería	Por haberme permitido pasar por sus aulas viviendo buenos y difíciles momentos a lo largo de la carrera.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
A	Amperio
←	Asignación
Δt	Diferencia de tiempo
Y	Electroválvula
GB	Gigabyte
∫	Integral
I	Intensidad luminosa
∩	Intersección
λ	Longitud de onda
H	Matiz
$\boldsymbol{m_{ij}}$	Momentos discretos
$\boldsymbol{m_{pq}}$	Momentos de orden pq
$\boldsymbol{\mu_{pq}}$	Momentos centrales
$\boldsymbol{\eta_{pq}}$	Momentos centrales normalizados
nm	Nanómetro
T	Periodo
S	Saturación
∑	Sumatoria
THz	Tera Hertz
$\boldsymbol{t_a}$	Tiempo de encendido
$\boldsymbol{t_b}$	Tiempo de apagado
V	Valor a la amplitud de luz
V	Voltio

GLOSARIO

Actuador Dispositivo o máquina que activa el sistema de control de un proceso.

Algoritmo Serie de procedimientos establecidos con el propósito de resolver un determinado problema.

Amperio Unidad de medida de la intensidad de corriente eléctrica.

AND Compuerta lógica digital, genera una salida con estado lógico alto solamente cuando ambas entradas son estados lógicos altos.

Arreglos En programación, matrices y vectores.

Bandera Variable que puede tomar únicamente dos estados, verdadero o falso.

Binario Propiedad de representar dos estados o condiciones excluyentes entre sí.

Bit Digito que forma parte del sistema binario, un bit puede representar un cero o un uno.

Boot Arranque o inicio de un sistema operativo.

Comando	Orden dada al ordenador para que realice una acción determinada.
CPU	Es uno de los componentes más importantes dentro de un ordenador, encargado de traducir todas las instrucciones y procesar los datos.
CSI	Estándar que define la interfaz entre una cámara digital y un procesador anfitrión.
Electroválvula	Dispositivo electromecánico diseñado para controlar el flujo que circula por un conducto**,** accionado eléctricamente.
Ethernet	Estándar de transmisión de datos en redes de área local para la conexión de equipos terminales a una misma línea.
Falling Edge	En señales, es la transición de alto a bajo.
Frame	Es una imagen formada por la cantidad de cuadros por segundos que han sido capturados durante un determinado lapso.
Función	En programación, representa una subrutina dentro de un código de programación que tiene una tarea específica.

GNU/Linux	Conjunto de programas que permite la interacción entre el usuario y el ordenador.
GPU	Siglas en inglés para *Graphics Processing Unit,* es un coprocesador encargado de representar una imagen en pantalla.
Hardware	Conjunto de componentes materiales de un sistema informático.
HDMI	Siglas en inglés para *High Definition Multimedia Interface,* es una norma de audio y video digital.
IEEE	Siglas en inglés para *Institute of Electrical and Electronics Engineers*, asociacion de profesionales que aportan criterio de estandarización de dispositivos eléctricos y electrónicos.
Interfaz de potencia	Dispositivos intermedios entre el computador o cualquier dispositivo programable y dispositivos de potencia.
Dirección IP	Siglas en inglés para *Internet Protocol*, conjunto de números que identifica, de manera lógica y jerárquica, a una interfaz en red.
Parámetro	Conjunto de condiciones que modifican una variable de salida.

Píxel	Unidad más pequeña de representación en la que se subdivide una imagen.
Powershell	Interfaz de consola con posibilidad de escritura y unión de comandos por medio de instrucciones.
RAM	Siglas en inglés para *Random Access Memory*, encargado de almacenar los datos que un ordenador está utilizando.
Rasterización	Manipulación de una imagen vectorial a formato en pixeles.
Relé de estado sólido	Dispositivo semiconductor accionado con baja tensión encargado de conmutar elementos de mayor potencia.
Script	Documento que contiene instrucciones, escritas en código de programación.
SD	Siglas en inglés para *Secure Digital*, es un dispositivo en formato de tarjeta de memoria para dispositivos portátiles.
Software	Término que hace referencia a la intangibilidad de los programas y corporeidad de la máquina.
SSH	Siglas en inglés para *Secure Shell*, protocolo dedicado para establecer conexión remota de manera segura.

Thread	En programación, es una técnica para programar distintas tareas y que sean ejecutadas de manera simultánea.
USB	Siglas en inglés para *Universal Serial Bus*, estándar industrial que define los cables, conectores y protocolos usados en un bus serial para conectar dos dispositivos.
Volt	Unidades para voltaje.

RESUMEN

En el primer capítulo se estudian los conceptos fundamentales sobre fajas transportadoras, tipos, lubricación, lubricación en fajas transportadoras, tipos de envases, todo lo referente a un ordenador de placa reducida, sistemas en un solo chip y Raspberry Pi, como fundamentos esenciales previo a capítulos posteriores.

El segundo capítulo describe el hardware y software empleado. Se detalla las configuraciones iniciales de las Raspberry Pi, se especifica la disciplina empleada para el procesamiento de imagen. Además, se precisan las configuraciones iniciales de OpenCv en la Raspberry Pi.

El tercer capítulo detalla el diseño de cada una de las etapas del prototipo, diseño de software e interfaz de potencia. El cuarto capítulo desarrolla el análisis de resultados de las etapas de detección y algoritmo de lubricación del prototipo. El quinto capítulo explica el análisis financiero del costo total para el desarrollo del prototipo, basado en el modelo costo total de propiedad.

OBJETIVOS

General

Diseñar un prototipo utilizando la tecnología PI para el control de lubricación en fajas transportadoras en envases PET.

Específicos

1. Dar a conocer los fundamentos teóricos de una faja transportadora, los tipos de lubricación y tipos de envases para bebidas.

2. Poner en conocimiento a la plataforma PI como una alternativa para automatización en la industria.

3. Diseñar un algoritmo en Python usando visión artificial para procesar imágenes en tiempo real en la detección de envases PET.

4. Diseñar un algoritmo en Python para la toma de decisiones en el control de lubricante.

INTRODUCCIÓN

En la industria de bebidas, los sistemas de lubricación en fajas transportadoras son de gran importancia y siempre deben ser eficiente para que la fricción entre el envase y el transportador sea óptima, para un funcionamiento correcto. En la actualidad, estos sistemas de automatización requieren de eficiencia, bajo costo en su implementación y mantenimiento. Desde luego, la tecnología Raspberry PI cumple con estos requerimientos industriales.

En los últimos años, la tecnología Raspberry Pi se ha destacado como una herramienta de experimentación y educación para profesionales de ingeniería; sin embargo, esta tecnología se está extendiendo cada día más para aplicaciones industriales.

Surge, entonces, la necesidad de proponer un prototipo para el control de lubricación en fajas transportadoras en envases PET. El prototipo se coloca en puntos estratégicos en una línea de producción para controlar la dosificación de lubricante mediante el encendido y apagado de las electroválvulas. El intervalo de tiempo es ajustable fácilmente desde la interfaz de usuario; además, el prototipo integra un sistema de visión artificial para regular en automático el ciclo de operación de las electroválvulas en función de la demanda de producto.

1. MARCO TEÓRICO

1.1. Fajas transportadoras

Las fajas transportadoras son sistemas electromecánicos cuyo objetivo es trasladar materiales de un destino a otro; funcionan por sí solas, por lo que no requieren de operadores. En un proceso industrial es común encontrar una serie de fajas transportadoras conectadas de manera continua. La importancia del uso de estos sistemas en la industria radica en transportar materiales de manera continua.

1.1.1. Tipos de fajas transportadoras

Existen fajas transportadoras en diversas formas y materiales, cuyas características pueden satisfacer cualquier necesidad industrial en diferentes procesos.

1.1.1.1. Fajas de rodillos

Estas fajas transportadoras trasladan materiales mediante el uso de rodillos de metal. Estos tienen como objetivo favorecer el traslado y manejo de los materiales.

Los contenedores o productos son trasladados de un punto a otro por el efecto rotatorio de los rodillos. El movimiento es comúnmente transmitido por motores, aunque en ocasiones simplemente por efecto de gravedad, ver la figura 1.

Figura 1. **Faja de rodillos**

Fuente: Maskepack. *Catálogo transportadores cintas*. p. 36.

1.1.1.2. Fajas con ruedas

A nivel de operación son similares a las fajas de rodillos. La traslación de contenedores se realiza mediante un arreglo de ruedas que se encuentran a lo largo de la cinta transportadora, como se observa en la figura 2. Se utilizan para trasladar producto, material o cualquier tipo de contenedores a otro. Por lo general se utilizan junto a mesas de ensamble.

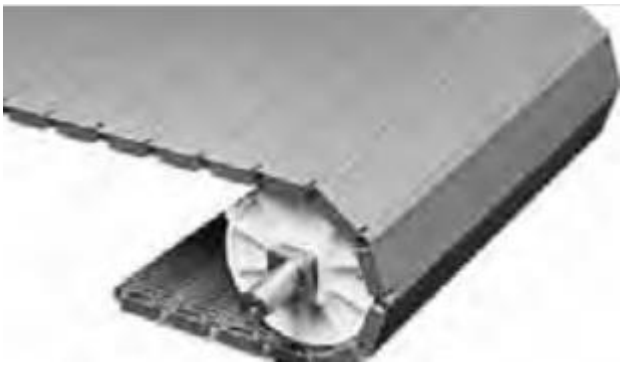
Figura 2. **Faja con ruedas**

Fuente: Intralox. *Manual de ingeniería de las bandas transportadoras*. p. 66.

1.1.1.3. Fajas planas

En este tipo de faja transportadora, los contenedores o productos viajan sobre una banda flexible y resistente. La traslación de la banda se realiza mediante motores y tensores encargados de mantener la estabilidad del transporte, observe la figura 3.

Figura 3. **Faja plana**



Fuente: Intralox. *Manual de ingeniería de las bandas transportadoras*. p. 58.

1.1.1.4. Fajas con cadenas

Este tipo de fajas transportadoras están formadas por una red de cadenas metálicas, como se observa en la figura 4. Ruedas dentadas conectadas al rotor de un motor son las encargadas de trasladar el movimiento a las cadenas. Este tipo de fajas son flexibles y robustas, empleadas en el traslado de contenedores o palés.

Figura 4. **Faja con cadena**

Fuente: Cambelt. *Bandas transportadoras metálicas con cadena.* p. 1.

1.1.2. Partes de una faja transportadora

Las fajas trasportadoras están compuestas principalmente por una banda cerrada, rodillos y tambores.

1.1.2.1. Banda cerrada

Es uno de los elementos más importantes, puesto que es el encargado de trasladar los contenedores de un punto a otro. Además, están en contacto directo con el material por transportar.

1.1.2.2. Rodillos

Su objetivo es facilitar el desplazamiento de la banda. La calidad de la faja transportadora depende en gran parte del buen funcionamiento de los rodillos. El mal funcionamiento de estos implica un aumento en el consumo de energía,

daños en la cinta transportadora y, por consiguiente, posibles daños en los contenedores.

1.1.2.3. Tambores

Una faja transportadora utiliza dos tipos de tambores: tambores motrices y no motrices, los cuales se encuentran en los extremos de la faja.

1.1.2.3.1. Tambores motrices

Están conectados directamente al rotor del motor y su función principal radica en transmitir fuerza a la banda.

1.1.2.3.2. Tambores no motrices

Su objetivo es cambiar la dirección destino de los contenedores. El movimiento es producido por la banda misma.

1.1.2.4. Bastidores

Los bastidores son los encargados de soportar el peso de la banda transportadora y los contenedores. Son estructuras rígidas metálicas, las cuales se encuentran colocadas de forma estratégica a lo largo del transporte.

1.1.2.5. Motorreductor

El motor es el encargado de transmitir movimiento rotacional y con acoples de poleas o engranajes transmitir movimiento traslacional a la banda. El motorreductor regula la velocidad del motor.

1.2. Lubricación

El objetivo de la lubricación es disminuir el coeficiente de fricción entre dos superficies, en este caso, entre el contenedor y la banda. Además, mitigar el efecto de fricción es menester que exista una delgada capa de lubricante entre las superficies, para evitar daños tanto en el contenedor como en el transporte. En la industria de alimentos, los lubricantes utilizados deben ser solubles en agua y libres de agentes contaminantes.

1.2.1. Tipos de lubricación

Hay varios métodos de lubricación utilizados en los sistemas de transporte, los cuales se pueden dividir en lubricación hidrodinámica e hidrostática.

1.2.1.1. Lubricación hidrodinámica

Consiste en aplicar una gruesa capa de lubricante entre dos superficies. Cuando estas entran en movimiento se crea una cuña que ejerce presión y evita el contacto directo entre las superficies.

1.2.1.2. Lubricación hidrostática

Consiste en aplicar de manera constante lubricante a presión entre las dos superficies que se desea mantener ligeramente separadas. Las electroválvulas son utilizadas en puntos estratégicos para dosificar lubricante.

1.2.2. Lubricación en fajas trasportadoras

Un efecto de fricción no controlado en cintas trasportadoras de envases PET contribuye a que los envases se vuelquen y provoquen frecuentes paradas de las fajas transportadoras, lo que se traduce en perjuicio en el proceso de producción. Las industrias de bebidas solo pueden conseguir sus objetivos si toda la instalación de las fajas transportadoras funciona sin problemas, es decir, sin altos índices de fricción ni residuos en las cadenas de transporte. Esto se consigue con una correcta lubricación.

1.3. Envases

El envase es el elemento encargado de almacenar un determinado producto final para que sea de fácil comercialización y llegue al consumidor. Por consiguiente, el envase es el encargado de proteger y de la presentación del producto que contiene.

1.3.1. Envase de metal

Un envase metálico es utilizado para almacenar productos sólidos, semisólidos o líquidos. Puede cerrarse herméticamente. Se manufactura a partir del aluminio y acero.

1.3.2. Envase de vidrio

Es utilizado para contener productos líquidos, cuyas propiedades no se alteran bajo los efectos de la luz. Sus dos mayores desventajas son la fragilidad mecánica y el peso elevado.

1.3.3. Envase de plástico

Los plásticos representan, en la actualidad, unos de los principales materiales para los envases. Este material ha tenido una gran influencia en el sector, dada su versatilidad, sus propiedades físicas y químicas.

1.3.3.1. Tipos de envases plásticos

En la industria se utilizan diferentes tipos de envases de plástico donde las propiedades químicas del material determinan el propósito del envase.

1.3.3.1.1. Material PET

El polietileno tereftalato (PET) es un material de alta calidad, también llamado el vidrio plástico, es un material de alta resistencia y rigidez. Soporta agentes químicos y tiene estabilidad a la intemperie, baja absorción de humedad y no se deteriora ni causa efectos de toxicidad a los productos. Puede ser llenado a temperaturas normales y en caliente. Es de gran aplicación en envases para bebidas carbonatadas, agua, productos farmacéuticos, cosméticos, conservas y aceite.

1.3.3.1.2. Material PEAD

El polietileno de alta densidad (PEAD) es un material cuyas propiedades lo hacen más resistente que el PET. Se utiliza en envases de detergentes, tanques, recubrimiento de cables y conos de señalización.

1.3.3.1.3. Material PP

El polipropileno (PP), debido a su bajo peso, se utiliza para la fabricación de bolsas, botellas y películas para envases flexibles, sacos, cartón ondulado, estuches, entre otras aplicaciones.

1.3.3.1.4. Material PC

El policarbonato (PC) tiene una alta transparencia y resistencia al impacto, presenta muy buen desempeño al calor. Es utilizado en botellones de agua y en bandejas de productos congelados.

1.4. SBCs y Raspberry Pi

Para comprender la diferencia entre un computador regular y una Raspberry Pi es preciso entender qué es un SBC.

1.4.1. SBC

Un SBC, *Single Board Computer*, es una computadora completamente funcional construida en un circuito impreso de tamaño reducido. Un SBC está conformado por microprocesadores, memoria, entradas y salidas, entre otras características requeridas de una computadora con un funcionamiento mínimo. La mayoría de SBC no tienen *slots* de expansión para funciones periféricas o de expansión. Los microprocesadores, RAM y GPU están integrados en un circuito impreso.

Tabla I. **Diferencia entre un SBC y una computadora regular**

SBC	Computadora regular
No modular	Modular
Componentes no pueden ser actualizados o reemplazados	Componentes pueden ser actualizados o reemplazados
Un sistema en chip (SoC)	No es un SoC
Tamaño reducido	Tamaño grande
Es portable	No portable o semi portable
Bajo consumo de energía	Consume más energía
Mas barato que una computadora regular	Mas costoso que un SBC

Fuente: PAJANKAR, Ashwin*. Raspberry Pi image processing programming*. p. 2.

1.4.1.1. SoC

Un SoC, sistema en un chip, es un circuito integrado que contiene todos los componentes de una computadora en un único chip, observe la figura 5. Los SoC son utilizados en teléfonos móviles, SBC, y en *hardware* embebido.

Figura 5. **El SoC de una Raspberry Pi**

Fuente: HALFACREE, Gareth. *Raspberry Pi beginner's guide*. p. 10 figure 1-2.

1.4.2. SoC frente a una CPU regular

La ventaja más grande de usar un SoC es su tamaño. Con una CPU es muy difícil hacer una computadora compacta, solo por el número de chips individuales y otros componentes que se necesitan para organizar una placa. Sin embargo, cuando se usa SoC se pueden colocar completos sistemas informáticos para aplicaciones específicas en teléfonos inteligentes y *tablets*, con mucho espacio para baterías, la antena y otros complementos necesarios para telefonía remota y comunicación de datos.

1.4.3. Familias de SBC corregir en toda la tesis

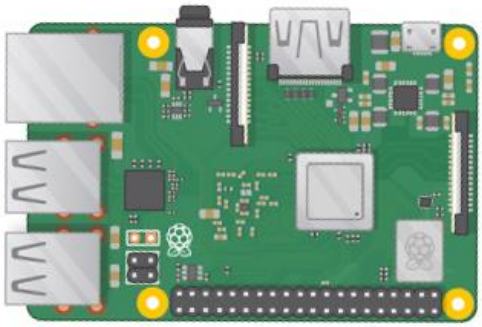
Basado en los fabricantes y diseñadores, los SBC se agrupan en familias, modelos y generaciones. A continuación, algunas familias populares de SBC:

- Raspberry Pi
- Intel Edison y Galileo
- CubieBoard

1.4.4. La Raspberry Pi

La PI, como la llamaremos de aquí en adelante, es una familia de SBC; es decir, una computadora del tamaño de una tarjeta de crédito. Su objetivo es estimular la enseñanza de las ciencias de la computación. Sin embargo, debido a su gran capacidad de procesamiento se encuentran aplicaciones en hogares, aulas, oficinas, *data centers,* fábricas e incluso en embarcaciones autoimpulsadas y globos espaciales. En la figura 6 se muestra una PI modelo B +.

Figura 6. **PI Modelo B +**



Fuente: HALFACREE, Gareth. *Raspberry pi beginner's guide.* p. 9.

1.4.4.1. Hardware del PI

En la tabla II se detallan las especificaciones técnicas de la PI 3.

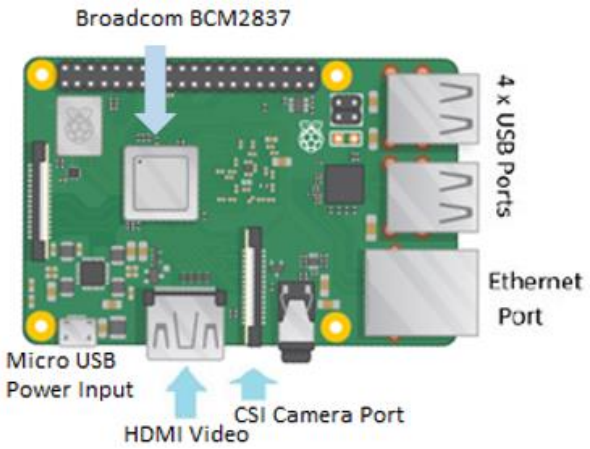
Tabla II. **Especificaciones técnicas PI 3**

	Raspberry pi 3 B +
Arquitectura	Cortex-A53 (ARMv8)
SoC broadcom	BCM2837B0
CPU	Cortex-A5 64-bit SoC @ 1.4GHz
Multimedia	H.264, MPEG-4 decode; (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 gráficos.
Memoria	1GB LPDDR2 SDRAM
USB	4 puertos 2.0
Salida de video	HDMI, salida estéreo 4 polos y puerto de video compuesto
Conectividad	2.4GHz y 5GHz IEEE 802.11.b/g/n/ac WLAN, Bluetooth 4.2, BLE
Fuente de alimentación	Micro USB 5V / 2.5A DC
Cámara	Puerto CSI cámara raspberry

Fuente: PAJANKAR, Ashwin. *Raspberry Pi image processing programming*. p.4.

En la figura 7 se muestra la parte superior de una PI 3 modelo B. Los componentes relevantes están etiquetados.

Figura 7. **PI 3 vista superior**



Fuente: PAJANKAR, Ashwin. *Raspberry Pi image processing programming*. p. 10.

1.4.4.2. Sistema operativo

Hay varias distribuciones diferentes para PI. Si lo que se requiere es hacer proyectos de *hardware* con PI, el sistema operativo recomendado es Raspbian. Este se basa en una distribución GNU/Linux llamada Debian, es un *software* de código abierto. Si se planea utilizar la PI como media center, OpenElec y XBian son de las distribuciones populares para este propósito en específico. PI permite usar otros sistemas operativos, incluyendo una versión de Windows 10.

2. MARCO METODOLÓGICO

2.1. *Hardware* utilizado

En este apartado se describen todos los elementos físicos que interactuaron en este trabajo.

2.1.1. PI 3 B +

La plataforma PI 3 B + es el componente principal en el diseño del prototipo, puesto que es el encargado de realizar todas las tareas computacionales del sistema. Las especificaciones técnicas se detallaron en el marco teórico, tabla II.

2.1.2. Fuente de alimentación

Para el suministro de energía se requiere una fuente de alimentación micro USB de 5 V y 2.5 A.

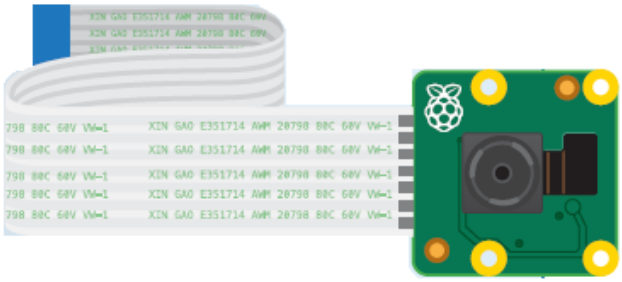
2.1.3. MicroSD

Es necesario instalar el sistema operativo en una tarjeta microSD. Se utilizó una tarjeta de 32 GB de capacidad. La tarjeta MicroSD actúa como el elemento de almacenamiento permanente de la PI; todos los archivos que sean creados y software que sean instalados junto con el propio sistema operativo, son almacenados en la tarjeta microSD. En la PI, la tarjeta SD se conecta al SoC a través de los pines GPIO 46 al 53 el SoC detecta la inserción de una tarjeta SD mediante el cierre de un socket.

2.1.4. PI cámara

La cámara que se utilizó es un módulo PI cámara v2. Se muestra en la figura 8 y se conecta al puerto de la interfaz serial de la cámara (CSI) en la PI. Proporciona alta resolución, captura de imágenes fijas y señales de video que pueden usarse tal cual o bien pueden ser procesadas en los programas creados.

Figura 8. **Módulo PI cámara**



Fuente: HALFACREE, Gareth. *PI beginner's guide.* p. 195.

2.2. *Software* utilizado

En este apartado se describe las herramientas de *software* empleadas en este trabajo.

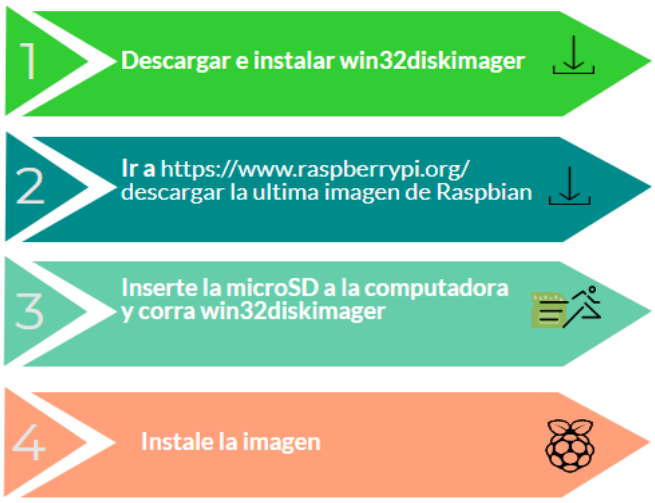
2.2.1. Raspbian

El sistema operativo utilizado en el prototipo es Raspbian. Este es el sistema operativo oficial para las tarjetas PI. Es un sistema operativo de código abierto basado en Debian optimizado para el hardware PI.

2.2.1.1. Instalación de Raspian en la microSD

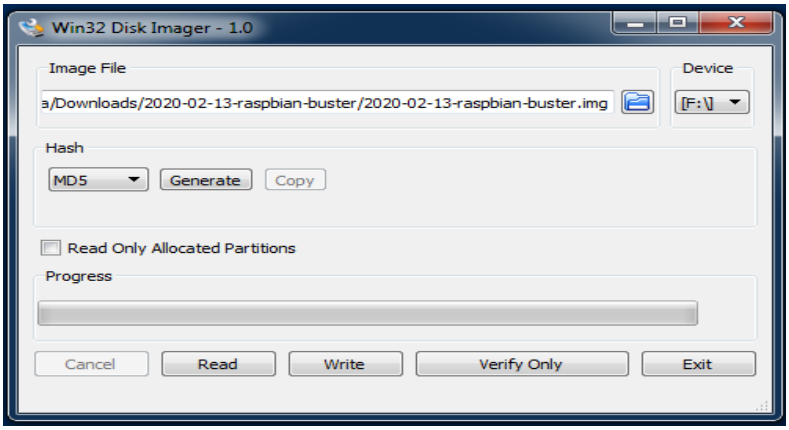
Para la instalación del sistema operativo es necesario realizar los pasos de la figura 9 y 10.

Figura 9. **Pasos para instalar Raspbian, usuarios Windows**



Fuente: elaboración propia, empleando PiktoChart.

Figura 10. **Captura del programa *Win32 Disk Imager***



Fuente: elaboración propia.

2.2.1.2. Configuraciones iniciales para PI

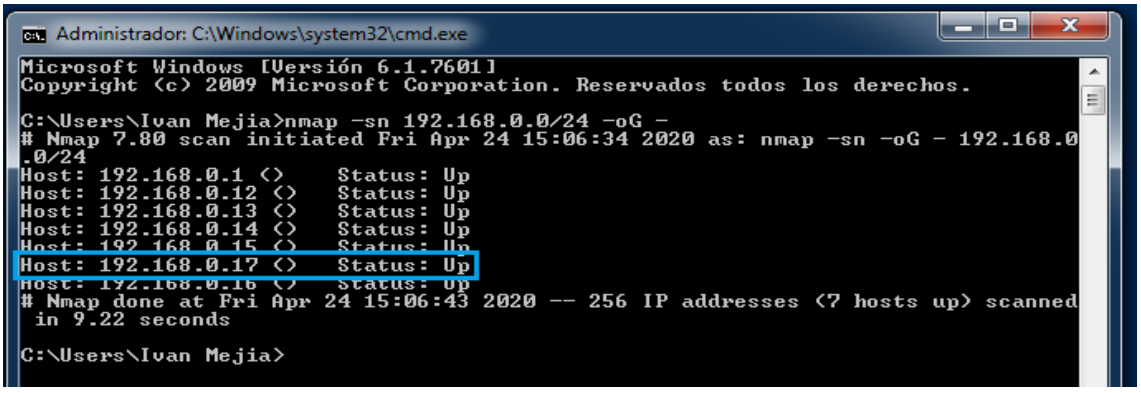
En esta sección se muestran todas las configuraciones necesarias para crear un escritorio virtual en *Windows.*

2.2.1.2.1. Cliente SSH

Para el desarrollo de este proyecto no se trabajó directamente en la PI sino desde otro dispositivo por medio de acceso remoto. Para ello fue necesario conectar la PI a una computadora por medio de conexión directa *eth ernet,* al encender la PI buscamos que se conecte a la red.

Para ver la dirección IP que se le asignó a la PI puede usarse el programa de código abierto NMAP, luego emplear el siguiente comando *nmap -sn <ip/mascara de red> -oG –*.

Figura 11. **Captura del comando *nmap -sn <ip/24> -oG* – en Windows**

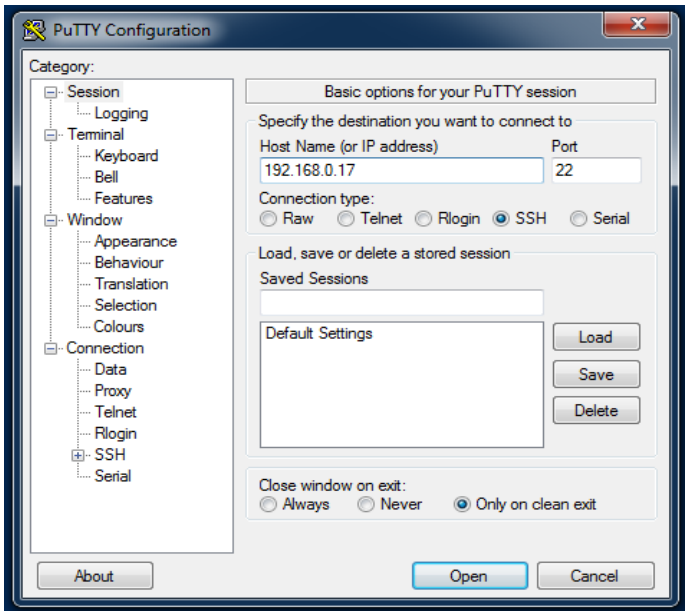


Fuente: elaboración propia.

En la salida de la aplicación se puede apreciar la dirección IP de esa PI en particular. El programa PUTTY permitirá hacer una conexión SSH (*Secure Shell)*, al introducir la dirección IP de la PI en PUTTY se establece la conexión remota.

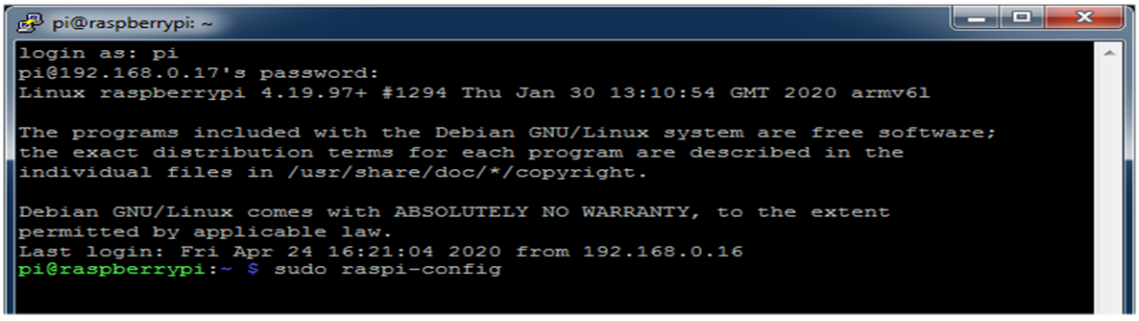
En las últimas versiones de PI la conexión SSH por defecto está desactivada. Para activarla hay que entrar a la partición *boot* de la microSD y abrir la venta de *power Shell* con las teclas *shift + click* derecho, empleando el comando dentro de la terminal type *nul > ssh.* Este comando crea un archivo SSH sin extensión y al conectar la PI ya es posible establecer una conexión SSH.

Figura 12. **Captura del programa PUTTY**



Fuente: elaboración propia.

Figura 13. **Captura de la conexión remota de la PI**



Fuente: elaboración propia.

En la salida de la aplicación se observa la terminal característica de PI. Al emplear el comando que se ve en la terminal *sudo raspi-config* es posible acceder a la herramienta de configuraciones de software.

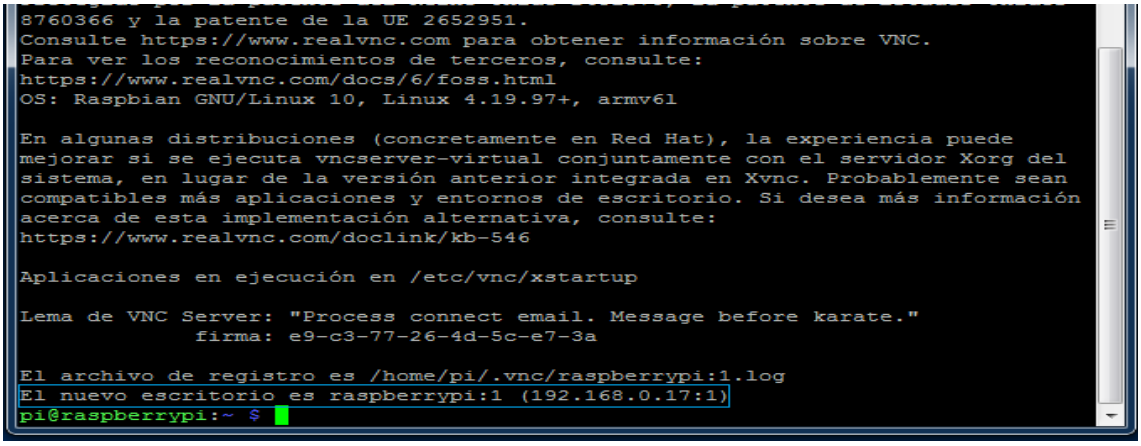
2.2.1.2.2. Habilitar puerto CSI de la PI

El puerto de la cámara PI por defecto se encuentra deshabilitado. Para habilitarlo se debe emplear la herramienta *raspi-config*, seleccionar *Interfacing Options* luego seleccionar *Camera* y escoger la opción habilitar. Luego, reiniciar.

2.2.1.2.3. Escritorio virtual

Para tener acceso a la interfaz gráfica es necesario activar una *Virtual Network Computing (*VNC*)*. Emplear el comando *sudo apt-get install -y realvnc-vnc-server realvnc-vnc-viewer* para la instalación del programa que permite ver la interfaz gráfica de forma remota. Una vez instalados los dos programas en la PI, se procede a emplear el comando *vncserver.*

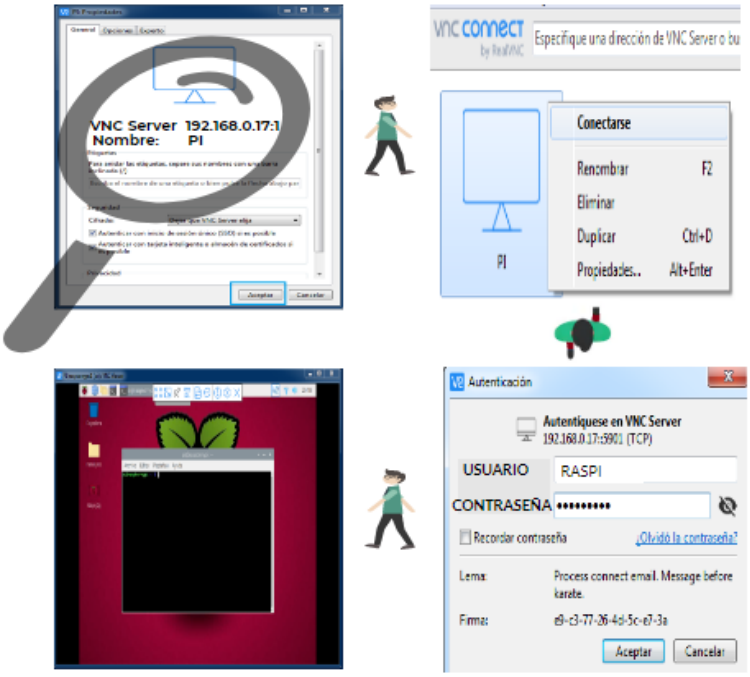
Figura 14. **Captura del comando *vncserver***



Fuente: elaboración propia.

El resultado de emplear el comando *vncserver* es que se ha creado un nuevo escritorio en el puerto 1, para conectarse a la vnc puede usarse el programa gratuito VNC Viewer.

Figura 15. **Captura de VNC bajo Raspbian, en Windows**



Fuente: elaboración propia, empleando PiktoChart.

2.2.2. Python

Python es un lenguaje de programación interpretado o de script; es decir, es un lenguaje que se ejecuta utilizando un programa intermedio llamado interprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora.

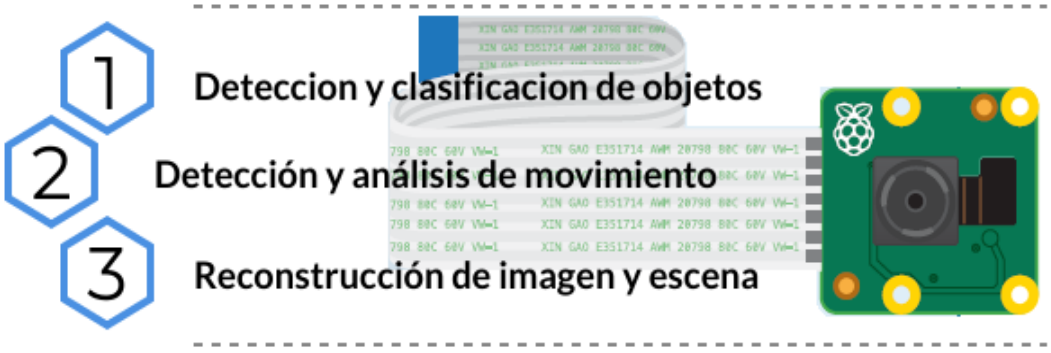
Python se caracteriza por ser un lenguaje simple, sencillo de aprender, fácil de leer. Su fuente está disponible gratuitamente; además, tiene una extendida

lista de librerías que vienen preinstaladas. En Raspbian los intérpretes Python 2 y Python 3 vienen preinstalados.

2.3. Disciplina empleada: visión artificial

La visión artificial comprende el área de ciencias de la computación, matemáticas e ingeniería eléctrica. La finalidad de los sistemas de visión artificial es el procesamiento, análisis y extracción de información útil de imágenes y videos del mundo que nos rodea. Estos sistemas buscan imitar la imagen percibida por los humanos.

Figura 16. **Tareas típicas de un sistema de visión artificial**



Fuente: elaboracion propia, empleando PiktoChart.

2.3.1. Color

El color es la percepción que tiene el ojo humano de las longitudes de onda en el espectro de luz visible. La extracción de estas características permite identificar objetos en una imagen.

2.3.1.1. Luz visible

Antes de entrar en detalle de los elementos de un sistema de visión es importante introducir el espectro electromagnético de luz visible. Esta es una forma de energía que se propaga en forma de radiación electromagnética que abarca una pequeña porción del espectro electromagnético total. El ojo humano percibe longitudes de onda que van aproximadamente desde 390 nm hasta 780 nm. Cada color específico de la luz tiene una longitud de onda y una frecuencia diferente, según la tabla III.

Tabla III. **Frecuencia y longitud de onda en el vacío para diferentes colores**

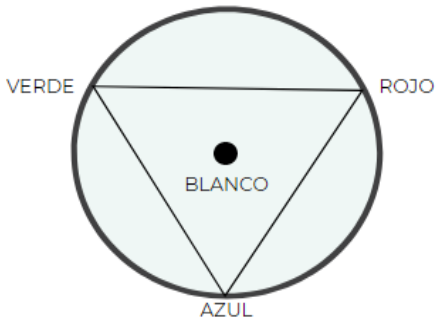
Color	λ_0(nm)	V(THz)
Rojo	780-622	384-482
Naranja	622-597	482-503
Amarillo	597-577	503-520
Verde	577-492	520-610
Azul	492-455	610-659
Violeta	455-390	659-769

Fuente: GARCÍA, José. *Fundamentos de Óptica ondulatoria.* p. 40.

2.3.1.2. Modelo RGB

Es un modelo de color en síntesis aditiva basada en la percepción de los colores del ojo humano. Considera los tres colores básicos y al combinarlos se consigue un nuevo conjunto de colores.

Figura 17. **Diagrama cromático**



Fuente: SUCAR, L. Enrique; GOMEZ, Giovani. *Visión computacional*. p.9.

2.3.1.3. Modelo HSI

En el modelo HSI (*Hue, Saturation, Intensity)* los colores se distinguen unos de otros por su tono, intensidad y saturación. Se considera el que más se aproxima a la percepción humana. Existe una forma directa de pasar la representación de color del modelo RGB al HSI y viceversa.

$$H = cos^{-1}\left(\frac{\frac{1}{2}(R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \quad (2.1)$$

$$S = 1-\left(\frac{3Min(R,G,B)}{R+G+B}\right) \quad (2.2)$$

$$I = \frac{1}{3}(R+G+B) \quad (2.3)$$

2.3.1.4. Modelo HSV

El modelo HSV (*Hue, Saturation, Value)* representa los colores combinando tres valores (tono, saturación y brillo) y representa mejor la forma en que las personas se relacionan con los colores que con el modelo RGB. La forma de pasar del color RGB a HSV se describe en las siguientes ecuaciones.

$$H \leftarrow \begin{cases} 60\dfrac{(G-B)}{(V\ -min(R,G,B))};\ si\ V=R \\ 120+60\dfrac{(B-R)}{(V\ -min(R,G,B))};\ si\ V=G \\ 240+60\dfrac{(R-G)}{(V\ -min(R,G,B))};\ si\ V=B \end{cases} \tag{2.4}$$

$$S \leftarrow \begin{cases} \dfrac{V-\ min(R,G,B)}{V};\ si\ V \neq 0 \\ 0;\ de\ otra\ manera \end{cases} \tag{2.5}$$

$$V\ \leftarrow max(R,G,B) \tag{2.6}$$

2.3.2. Procesamiento digital de imagen

Es imprescindible entender las técnicas empleadas en el procesamiento de imágenes para facilitar la extracción de información.

2.3.2.1. Procesamiento de señales

Una señal es cualquier cosa que lleve información sobre algún fenómeno y pueda ser representado matemáticamente. El proceso o técnica usado para

extraer información útil y relevante de una señal dada es conocido como procesamiento de señal.

Los sistemas que realizan este tipo de tareas son llamados sistemas de procesamiento de señal. Funcionan a través de circuitos y sistemas electrónicos. Hay dos tipos de señales electrónicas: las analógicas y las digitales, las cuales muestran sus diferencias en la tabla IV.

Tabla IV. **Tipos de señales electrónicas**

Analógica	**Digital**
Señal continua	Señal discreta
Denotado por ondas senoidales	Denotado por ondas cuadradas
Deteriorada debido al ruido	No es afectada por el ruido
Bajo consumo de ancho de banda	Alto consumo de ancho de banda
Hay muchos errores en la captura, almacenamiento y transmisión	Menores errores y existen algoritmos de verificación de errores disponibles

Fuente: PAJANKAR, Ashwin. *Raspberry Pi image processing programming*. p.34.

2.3.2.2. Procesamiento de imagen

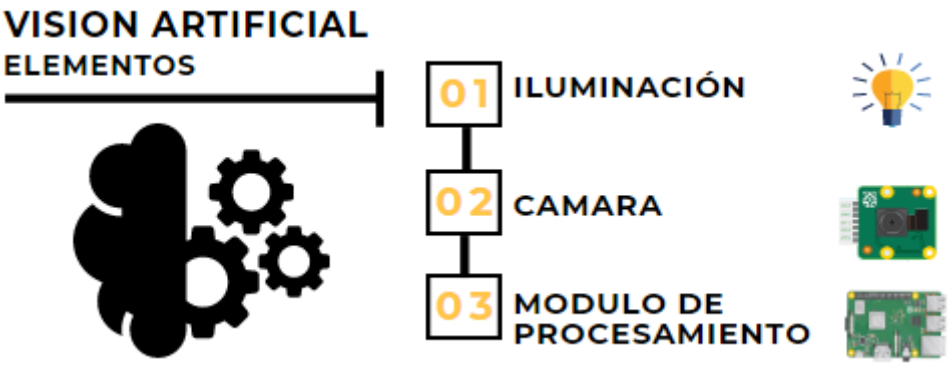
Es un tipo de sistema de procesamiento de señal, puesto que una imagen lleva información con ella. Existen dos tipos, el analógico y el digital. Con el advenimiento de los avances tecnológicos las imágenes se pueden capturar, almacenar y procesar en formatos digitales.

2.3.3. Elementos en un sistema de visión artificial

Los sistemas de visión artificial están compuestos por un conjunto de elementos que trabajan juntos para adquirir, procesar y analizar las imágenes.

Los principales componentes utilizados en el diseño del prototipo se describen en la siguiente figura.

Figura 18. **Elementos en un sistema de visión artificial**



Fuente: elaboración propia, empleando Piktochart.

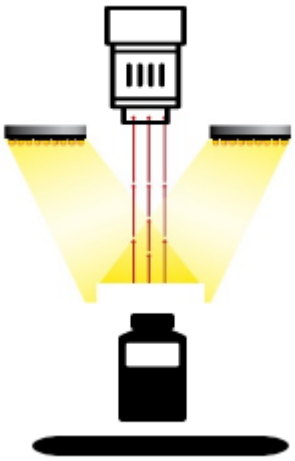
2.3.3.1. Iluminación

La iluminación es un aspecto importante para considerar en el diseño de sistemas de visión artificial. Una inadecuada iluminación puede traducirse en pérdida de información y productividad. Con la correcta técnica de iluminación se reducen las sombras, se consiguen texturas suaves y se reducen las imperfecciones. Por tanto, la captura de la cámara será capaz de conseguir imágenes fiables.

2.3.3.1.1. Iluminación frontal

La luz incide directamente sobre el objeto. Este tipo de iluminación permite distinguir a detalle ciertas características de los objetos, así como su forma, color, detalles internos, entre otras características.

Figura 19. **Iluminación frontal**

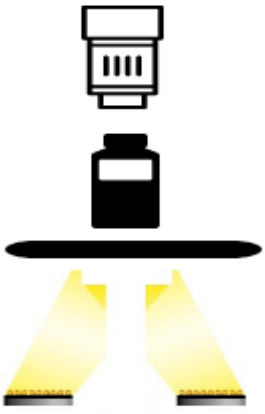


Fuente: elaboración propia, empleando PiktoChart.

2.3.3.1.2. Iluminación trasera

Esta técnica consiste en colocar el objeto entre la cámara y una pantalla. La iluminación se le aplica a la pantalla para obtener la sombra. Esta técnica sirve únicamente para detección de contornos, con eso se simplifica la etapa de segmentación.

Figura 20. **Iluminación trasera**

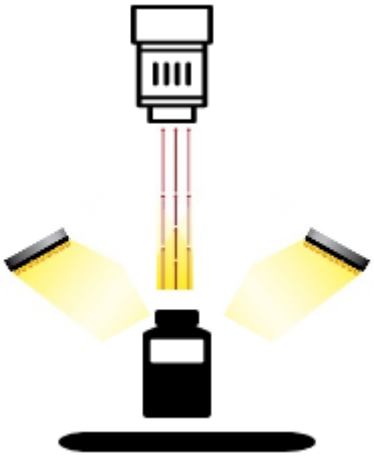


Fuente: elaboración propia, empleando PiktoChart.

2.3.3.1.3. Iluminación difusa

Este tipo de iluminación se emplea en superficies suaves y regulares. La luz especular se refleja alejándose de la cámara y la luz difusa procede de la superficie del objeto hacia la cámara.

Figura 21. **Iluminación difusa**



Fuente: elaboración propia, empleando PiktoChart.

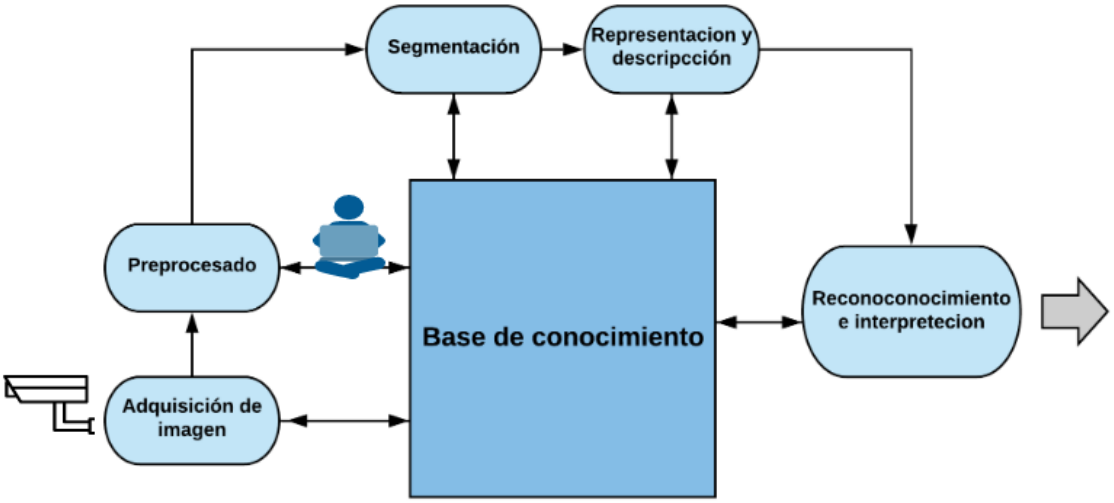
2.3.3.2. Módulo de procesamiento

El módulo de procesamiento es el encargado de recibir, almacenar y procesar las imágenes a través de algoritmos adecuados. Este módulo puede ser una computadora o un sistema integrado. En el diseño del prototipo de este trabajo, el módulo es la PI.

2.3.4. Etapas de un sistema de visión artificial

La secuencia de procesos en la obtención, caracterización e interpretación de información de imágenes capturadas de nuestro entorno pueden dividirse en cinco etapas. Un esquema general de las etapas se presenta en la figura 22.

Figura 22. **Etapas de un sistema de visión artificial**



Fuente: elaboración propia, empleando Lucidchart.

2.3.4.1. Adquisición de la imagen

Esta etapa consiste en el proceso de captura y digitalización de una imagen mediante una cámara PI. En esta etapa es necesario considerar los parámetros de la imagen como resolución, orientación y brillo, con el objetivo de lograr una adquisición de imagen de muy buena calidad. Se logra una correcta extracción de características de la imagen objetivo. El nivel de éxito en la identificación de envases, tanto de los que están volcados como de los que no, dependerá de una correcta adquisición de imágenes.

En la etapa de adquisición de imágenes se pueden presentar dos escenarios: captura de envases vista lateral y vista superior, lo que se interpreta como envase volcado y envase derecho, respectivamente. Por tal razón es necesario emplear una técnica para la extracción de características del contenido visual del envase en ambos escenarios.

2.3.4.2. Preprocesamiento

En la etapa de adquisición, toda imagen sufre en cierta medida los efectos de degradación, que se manifiestan en forma de ruido, pérdida y definición de la imagen. En esta etapa se utilizan técnicas de reducción de ruido y realce de detalles si se requiriera. De esta forma se aumentará la posibilidad de éxito en los procesos posteriores.

2.3.4.3. Segmentación

Etapa encargada de dividir la imagen ya digitalizada en sus partes u objetos constituyentes, es decir, regiones que contienen ciertas propiedades de interés o que sean el objetivo, como el color forma o textura. Con todo esto se obtiene más información relevante del objeto que se desea detectar.

2.3.4.3.1. Segmentación basados en puntos

Es una técnica de segmentación que utiliza la diferencia que existe entre el píxel central de interés y sus vecinos, considerando que cada píxel es un punto aislado siempre que discrepe significativamente de los demás puntos que lo rodean.

2.3.4.3.2. Segmentación por regiones

Esta técnica consiste en aumentar las regiones partiendo desde el centro de la imagen objetivo hasta que se encuentren con los puntos exteriores finales, para determinar los bordes que lo limitan.

2.3.4.3.3. Segmentación por bordes

Esta técnica busca identificar puntos en una imagen digital donde el brillo de la imagen cambia drásticamente; es decir, determina cada límite de las secciones presentes en una imagen.

2.3.4.4. Representación y descripción

En la etapa de segmentación se suele obtener una gran cantidad de pixeles, que constituyen los puntos de una región determinada. Es necesario convertir estos datos de tal forma que sean adecuados para el procesamiento por computadora. La representación dice cómo debe realizarse la segmentación, basada en puntos, en regiones o por segmentación de bordes. La descripción consiste en extraer características con información cuantitativa o cualitativa de interés.

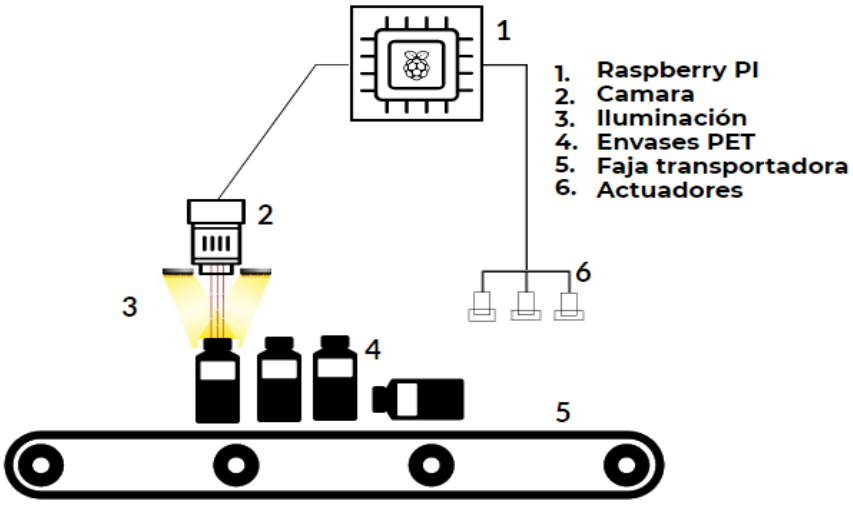
2.3.4.5. Reconocimiento e interpretación

Esta etapa consiste en asignar una etiqueta a un objeto con base en la información proporcionada; es decir, una vez que las características han sido extraídas, se procede a etiquetar cada punto o región para luego darle una interpretación o significado a un conjunto de objetos reconocidos.

2.3.4.6. Base de conocimiento

La base de conocimiento detalla las regiones de una imagen donde se sabe que existe información de interés, con el objetivo de limitar la búsqueda de información. Para nuestro caso, la base de conocimiento contiene imágenes de envases PET en diferentes vistas, tanto superior como lateral del envase.

Figura 23. **Sistema de detección de envases**



Fuente: elaboración propria, empelando PiktoChart.

2.4. Visión artificial en la PI

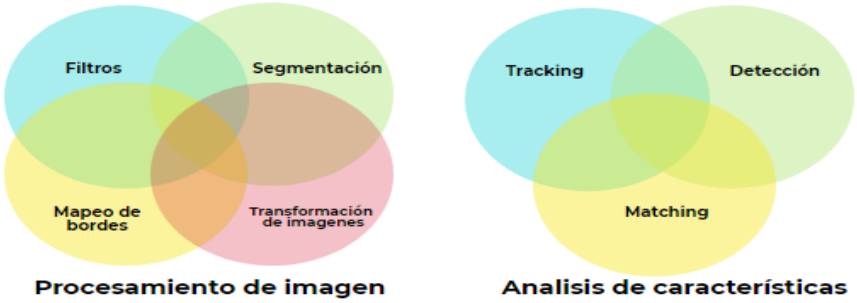
Luego que el acceso remoto ha sido configurado y está funcionando correctamente, es necesario preparar la PI para emplear visión artificial.

2.4.1. OpenCV

El diseño del prototipo de este trabajo se basa en gran medida en el procesamiento de imágenes, para lo cual existe una librería de código abierto

llamada OpenCV, *Open Source Computer Visión Library,* que consta de más de 2 500 algoritmos optimizados. El uso de esta librería varía desde reconocimiento facial, identificación de objetos y clasificaciones de acciones humanas en videos, logrados con filtros, mapeo de bordes, transformaciones de imágenes. Más características se muestran en la figura 24.

Figura 24. **Descripción parcial de lo que incluye OpenCV**



Fuente: elaboración propia, empleando PiktoChart.

Para instalar OpenCv en la PI emplear el comando *sudo apt-get install python-opencv.* Una vez instalada la librería OpenCv se debe realizar las conexiones de la cámara y la PI para emplear las diferentes etapas que conforman un sistema de visión artificial.

Tabla V. **Funciones de OpenCv más relevantes en la detección de objetos basados en colores, utilizando *python***

Función	Descripción
inRange	Comprueba si los elementos de la matriz de pixeles de una imagen se encuentran entre los elementos de otras dos matrices, inferior y superior, creando una máscara binaria.
Circle	Dibuja un círculo, esta función necesita una imagen de destino y el círculo dibujado se describe con una coordenada que marca el punto central, un valor de radio, el color y el grosor de su contorno.
Imshow	Muestra la imagen sobre la que se está trabajando.

Continuación tabla V.

cvtColor	Convierte una matriz de pixeles a otro modelo de color.
GaussianBlur	Desenfoca una imagen empleando un filtro gaussiano.
Blur	Técnica para eliminar el ruido de la imagen.
medianBlur	Elimina el ruido *salt-and-pepper.*
bilateralFilter	Elimina el ruido, pero a diferencia de los demás filtros, no difumina el contorno de la imagen.
moments	Calcula todos los momentos hasta el tercer orden de un polígono o forma rasterizada

Fuente: elaboración propia, empleando Partnership, Hardware, OpenCv. Opencv.org

3. DISEÑO DE PROTOTIPO

El prototipo diseñado consta de cinco partes: algoritmo de procesamiento de imagen, algoritmo para el control de lubricación, interfaz de usuario, interfaz de potencia y diseño de tablero electrónico.

3.1. Diseño de *software* de procesamiento de imagen

En esta sección se emplearon las técnicas de procesamiento de imagen para la detección de envase PET.

3.1.1. Adquisición de imagen y preprocesamiento

La adquisición de imágenes en tiempo real es el primer paso en un sistema de visión artificial, seguido de una etapa de filtrado para mejorar la calidad de la imagen y con ello facilitar la extracción de características.

3.1.1.1. Captura de video con el módulo PI cámara

Capturar el video directamente usando el módulo PI con OpenCv no requiere de ningún controlador ni otro software. La captura de video se inicializa llamando a la clase *VideoCapture* y asignándole un nombre.

Luego de inicializar la cámara es recomendable verificar si realmente se tuvo éxito o no. Esto se consigue con una sentencia *if,* porque la instancia de *VideoCapture* definida tendrá una función abierta que devuelve verdadero si la cámara está funcionando.

Figura 25. **Inicializando la clase *VideoVapture* en Python**

```
import cv2
from picamera import PiCamera
captura = cv2.VideoCapture()
while (True):
    res, frame = captura. read()
    if res == True:
```

Fuente: elaboración propia.

3.1.1.2. Conversión a espacio de color HSV

El primer paso para hacer que la imagen sea legible es convertirla en el espacio de color HSV, ya que es mucho más fácil aplicar un umbral (*threshold)* a una imagen. Además, este modelo es la mejor representación de cómo los humanos perciben los colores.

Figura 26. **Convirtiendo una imagen en el modelo HSV en Python**

```
ModeloHSV = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

Fuente: elaboración propia.

3.1.1.3. Filtrado

Para reducir el error de detección se utilizó un filtro de desenfoque gaussiano. Este tiene la forma de una curva en forma de campana, con un punto alto en el centro y una sección que se estrecha simétricamente a cada lado. En la siguiente figura se muestra la línea de código empleada para aplicar el filtro con un núcleo impar de 21 x 21.

Figura 27. **Filtrado Gaussian blur en opencv**

```
filtro = cv2.GaussianBlur(HSV, (21, 21), 0)
```

Fuente: elaboración propia.

3.1.2. Segmentación

Esta sección presenta las técnicas utilizadas para identificar objetos de interés en una imagen.

3.1.2.1. *Thresholding*

Es una técnica que consiste en eliminar los valores superiores e inferiores al ponerlos a cero respecto a un valor conocido como *threshold* o umbral. OpenCv trae la función *inRange* para obtener los rangos de un color en específico del modelo HSV. Para encontrar los valores HSV se puede emplear la función *cvtColor.* En lugar de pasar una imagen, pasar los valores BGR que se desean. Los valores estándar de BGR se describen en la tabla VII. La función *inRange* comprueba si los elementos de la matriz se encuentran entre los elementos de otras dos matrices.

$$dst(I) = inf(I)_0 \leq src(I)_0 \leq sup(I)_0 \cap inf(I)_1 \leq src(I)_1 \leq sup(I)_1 \qquad (3.1)$$

Tabla VI. **Parámetros de la función inRange**

Parámetro	Descripción
dst	matriz de salida del mismo tamaño que src, dst es el píxel actual en el *frame* de destino (donde se almacenara la imagen binaria)
inf	matriz de límite inferior inclusiva o un escalar
sup	matriz de límite superior inclusiva o un escalar
src	primera matriz de entrada

Fuente: elaboración propia.

Tabla VII. **Valores de color estándar BGR**

Color	BGR
Negro	(0,0,0)
Blanco	(255,255,255)
Rojo	(255,0,0)
Verde	(0,255,0)
Azul	(0,0,255)
Amarillo	(255,255,0)
Cian	(0,255,255)
Magenta	(255,0,255)

Fuente: MUÑOZ, Valero. *Principios de color y holipintura*. p. 162.

Con los valores conocidos de BGR y con la función *cvtColor* en la terminal de Python se puede emplear las siguientes líneas de comando para conocer los valores HSV.

Figura 28. **Ejemplo para encontrar el valor HSV del color azul en Python**

```
>>>  bgr_azul = np. uint8([[[ 0, 0, 255]]])
>>>  hsv_azul = cv2.cvtColor(bgr_azul, cv2.COLOR_BGR2HSV)
>>>  print hsv_azul
>>>  [[[ 0 255 255]]]
```

Fuente: elaboración propia.

En la salida de la terminal de la figura 28, el resultado es el rango que abarca el color azul en el modelo HSV.

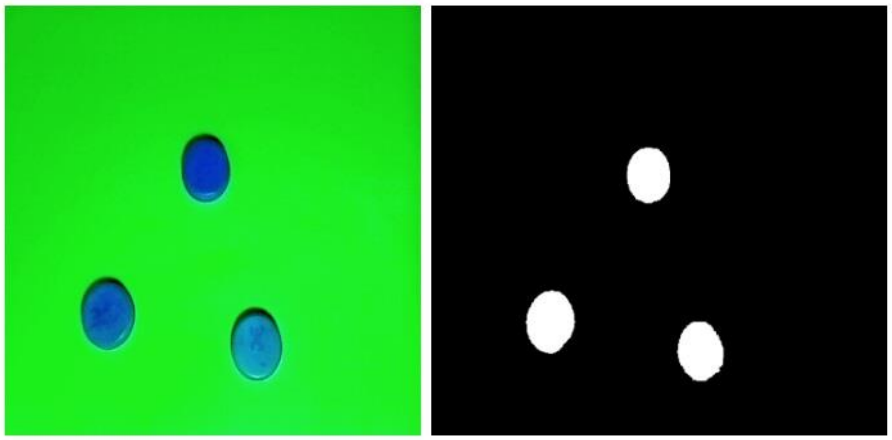
Figura 29. **Función *threshold* por rango de colores en Python**

```
inf_azul = np.array([100,50,100],np.uint8)
sup_azul = np.array([120,255,255],np.uint8)
mascara = cv2.inRange(ModeloHSV, inf_azul, sup_azul)
```

Fuente: elaboración propia.

En esta etapa se busca delimitar alguna de las características de los envases PET. Como se mencionó en el marco teórico, dichos envases son claros o brillantes, lo cual hace difícil su detección. En el diseño de esta etapa se extraerá las características de la tapa del envase, puesto que estas vienen en presentaciones de colores sólidos. En la figura 30 se muestra la imagen antes y después de aplicarle un *threshold*.

Figura 30. **Imagen original y después del *threshold***



Fuente: elaboración propia, empleando OpenCv en Python.

En la figura 30 se observa que en la imagen original hay dos colores que resaltan, azul y verde. Al aplicar el *threshold* por color se muestra una imagen

binaria, donde a todo aquello que esté en el rango de límite inferior y superior azul le asigna 255 y todo aquello que no se encuentre en el rango le asigna 0.

3.1.2.2. Operador *AND*

Existen diferentes operaciones lógicas. El operador *AND* es muy utilizado en máscaras que sirven para obtener o eliminar una porción determinada de la imagen. Utilizar este operador es opcional en caso de que se esté utilizando *threshold*. Sin embargo, es útil para verificar la salida de la imagen y con ella detectar imperfecciones que puedan dificultar las etapas posteriores. OpenCv trae la función *bitwise_and* que calcula la conjunción lógica a nivel de bit por elemento.

$$dst(I) = src1(I) \cap src2(I); \; si \; mask \neq 0 \qquad (3.2)$$

Tabla VIII. **Parámetros de la función *bitwise_and***

Parámetro	Descripción
Dst	matriz de salida que tiene el mismo tamaño y tipo que las matrices de entrada.
src1	primera matriz de entrada o una escala.
src2	segunda matriz de entrada o una escala.
Mask	mascara de operación opcional, matriz de un solo canal de 8 bits, que especifica los elementos de la matriz de salida a cambiar.

Fuente: elaboración propia.

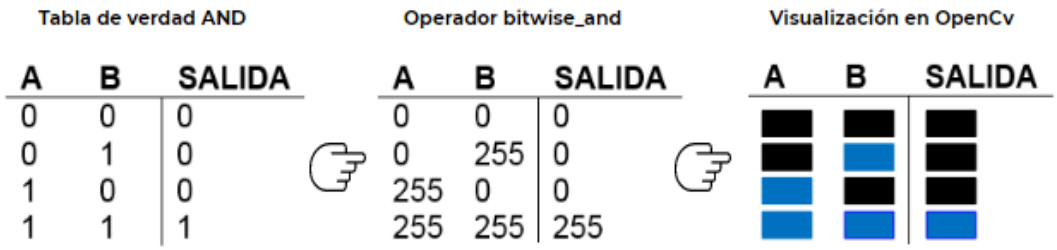
Figura 31. **Función *bitwise_and* en Python**

```
img_and = cv2.bitwise_and (frame, frame, mask= mask)
```

Fuente: elaboración propia.

Para entender mejor cómo funciona este operador en OpenCv es importante recordar la tabla de verdad del operador lógico *AND.* En la figura 32 se muestra una analogía de cómo trabaja el operador *bitwise_and* en OpenCv. Se usa como ejemplo el color azul.

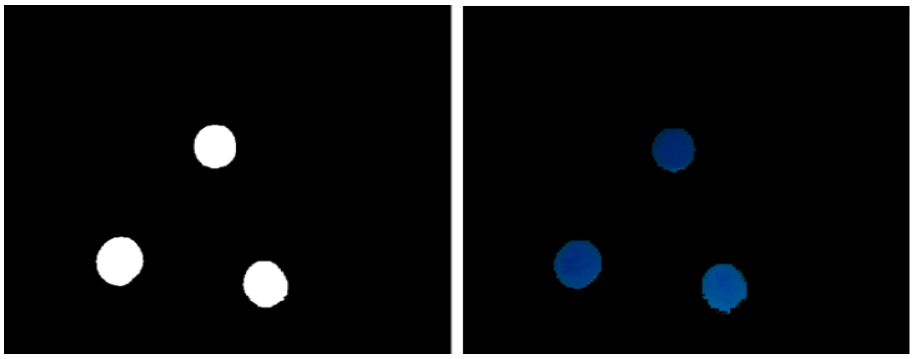
Figura 32. **Operador *bitwise_and***



Fuente: elaboración propia, empleando PiktoChart.

En la figura 33 se muestra la comparación del resultado de emplear un *threshold* y el operador *bitwise_and*. El resultado para ambos es una imagen binaria, con la diferencia que el operador *bitwis_and* asigna al color negro 0 y al color azul 255.

Figura 33. **Imagen con *threshold* y después de aplicar el operador *bitwise_and***



Fuente: elaboración propia, empleando OpenCv en Python.

3.1.3. Representación y descripción

Esta etapa consta de dos partes: aproximación de contorno y diseño del algoritmo de seguimiento.

3.1.3.1. Detección de contorno

Los contornos son curvas que unen los puntos continuos a lo largo del límite, que tienen el mismo color o intensidad. La detección de contorno es una técnica de segmentación de objetos, especialmente utilizada para el análisis de formas, detección y reconocimiento de objetos. OpenCv tiene incluido el algoritmo *findContours*. El objetivo final consiste en alcanzar al menos una segmentación parcial. Los parámetros empleados se explican en la tabla IX y en la figura 34 se muestra la función empleada en Python.

Tabla IX. **Parámetros utilizados en el algoritmo de detección de contorno**

Parámetro	Descripción
Imagen	Imagen (mascara) con el *threshold* de color previamente aplicado.
Modo: *RETR_EXTERNAL*	Método utilizado para recuperar únicamente los contornos externos.
Método: *CHAIN_APPROX_SIMPLE*	Permite realizar el método de aproximación para recuperar las formas de los contornos, este método comprime toda la horizontal, segmentos verticales y diagonales que almacenan solo los puntos de partida y puntos finales.

Fuente: elaboración propia.

Figura 34. **Función *findContours* en Python**

```
_, contorno, _ = cv2.findContours(mascara,
                                  cv2.RETR_EXTERNAL,
                                  cv2.CHAIN_APPROX_SIMPLE)
```

Fuente: elaboración propia.

3.1.3.2. Algoritmo Douglas-Peucker

Después de recopilar información sobre todos los contornos capturados, el programa puede aproximar una forma de contorno a otra al reducir el número de vértices, dependiendo de la precisión que se especifique. Para ello es necesario calcular el polígono delimitador aproximado de una forma. OpenCv ofrece una implementación del algoritmo Douglas-Peucker mediante la función *approxPolyDP.*

Lo que se busca con esta función es que las tareas de visión por computadora se vuelvan más simples al definir polígonos para delimitar regiones para una mayor manipulación y procesamiento. Esta función toma tres parámetros, los cuales se describen en la tabla X.

Tabla X. **Parámetros de la función *approxPolyDP***

Parámetro	Descripción
Matriz de entrada	Un contorno.
Épsilon	Un valor épsilon que representa la máxima discrepancia entre el contorno original y el polígono aproximado, cuanto menor sea el valor, más cerca estará el valor aproximado del contorno original.
Bandera booleana	Una bandera booleana. Si es *True*, significa que el polígono está cerca del valor aproximado.

Fuente: elaboración propia.

Épsilon es la máxima diferencia entre el perímetro aproximado del polígono y el perímetro del contorno original. El valor épsilon es de vital importancia para obtener un contorno útil. Cuanto menor sea esta diferencia, más se aproximará al contorno original que es lo que se busca. Para obtener el valor de épsilon es necesario obtener información del perímetro del contorno como valor de referencia, esto se consigue empleando la función *arcLength.*

Tabla XI. **Parámetros de la función *arcLength***

Parámetro	Descripción
Matriz de entrada	Un contorno.
Bandera booleana	Una bandera booleana. Si es *True*, significa que el polígono está cerca del valor aproximado.

Fuente: elaboración propia.

En la figura 35 se muestran las dos funciones empleadas en el algoritmo Douglas-Peucker, la función que calcula el perímetro de un contorno y la de aproximación de polígono.

Figura 35. **Función *approxPolyDP* en Python**

```
epsilon = 0,01*cv2.arcLength(contorno, True)
aproxContorno = cv2.approxPolyDP(contorno, epsilon, True)
```

Fuente: elaboración propia.

Tabla XII. **Comparación del resultado de las funciones *findContours* y *approxPolyDP***

No.	Contorno sin aproximación	No.	Contorno con aproximación
	Coordenadas (x, y)		**Coordenadas (x, y)**
1	[[[320, 256]],	1	[[[302, 260]],

Continuación tabla XII.

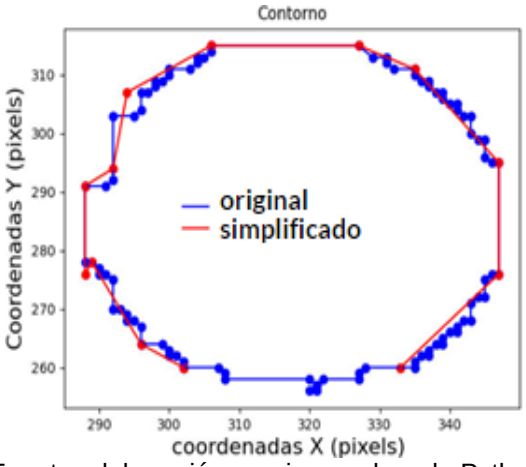
2	[[321, 257]],	2	[[296, 264]],
3	[[320, 258]],	3	[[289, 278]],
4	[[308, 258]],	4	[[288, 276]],
5	[[308, 259]],	5	[[288, 291]],
6	[[307, 260]],	6	[[292, 294]],
7	[[302, 260]],	7	[[294, 307]],
8	[[302, 261]],	8	[[306, 315]],
9	[[301, 262]],	9	[[327, 315]],
10	[[300, 262]],	10	[[335, 311]],
11	[[300, 263]],	11	[[347, 295]],
12	[[299, 264]],	12	[[347, 276]],
93	[[321, 256]]]	13	[[333, 260]]]
$Número\ de\ vertices\ =\ 93$		$Número\ de\ vertices\ =\ 13$	

Fuente: elaboración propia.

En la tabla XII se muestran dos matriz de coordenadas (x,y) de puntos límite del objeto generadas por la función *findContours*, y la función *approxPolyDP*. El resultado de la salida de la función *findContours* es una matriz de 93×2 y para la función *approxPolyDP* es una matriz 13×2. Es evidente que mediante la implementación del algoritmo de Douglas-Peucker se ha aproximado una forma de contorno con menos números de vértices.

En la figura 36 se observa la representación gráfica de las coordenadas obtenidas y expuestas en la tabla XII. En color azul se muestra el contorno original, es decir, la salida de la función *findContours,* y en color rojo, el contorno simplificado.

Figura 36. **Gráfico del resultado de las funciones *findContours* y *approxPolyPD***



Fuente: elaboración propia, empleando Python.

3.1.3.3. Algoritmo de seguimiento de centro

El algoritmo de seguimiento consiste en generar las trayectorias de los objetos en movimiento, calculando su movimiento en una secuencia de imágenes.

3.1.3.3.1. Cuadro delimitador

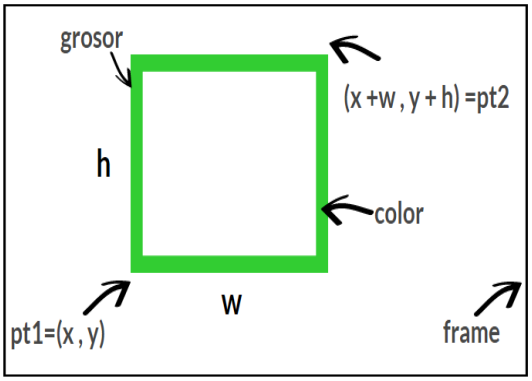
Para cada objeto detectado en un cuadro de video se crea un conjunto de coordenadas (x,y) de cuadro delimitador, el cual circunscribe el objeto detectado. Estos cuadros delimitadores necesitan de las características extraídas previamente, *threshold* de color y la extracción de contornos. OpenCv contiene la función *boundingRect*. Esta función calcula el rectángulo delimitador superior derecho de un conjunto de puntos. Para dibujar el cuadro delimitador OpenCv contiene la función de dibujo *Rectangle.*

Figura 37. **Función *boundingRec* en Python**

```
(x, y, w, h) = cv2.boundingRect(contorno)
cv2.rectangle(frame, pt1, pt2, color, grosor)
```

Fuente: elaboración propia.

Figura 38. **Parámetros empleados en la función *rectangle***



Fuente: elaboración propia, empleando PiktoChart.

3.1.3.3.2. Momentos

Los momentos son cantidades escalares que se utilizan ampliamente en el análisis de formas y el reconocimiento de patrones. El teorema de Green evalúa una integral doble sobre la región de un objeto mediante una simple integración a lo largo del límite del objeto, es decir, a lo largo de la periferia del objeto detectado. El teorema se ha utilizado para el cálculo de momentos, puesto que la forma del objeto binario está totalmente determinada por su límite. El momento de orden $(p + q)$ de una imagen se define como:

$$m_{pq} = \int_y \int_x g(x,y)\, x^p y^q \, dx \, dy \tag{3,3}$$

Donde $g(x,y)$ es la intensidad como función espacial. La integral doble a menudo se reemplaza por una doble suma en imágenes discretas, los momentos discretos bidimensionales estándar de $g(x,y)$:

$$m_{pq} = \sum_{y}\sum_{x} g(x,y)x^p y^q \tag{3,4}$$

Los momentos centrales se definen como:

$$\mu_{pq} = \int_{y}\int_{x} g(x,y)\,(x-\bar{x})^p\,(y-\bar{y})^q dx\,dy \tag{3,5}$$

Donde $(\bar{x},\bar{y})$ es el centro de masa:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}} \tag{3,6}$$

Los momentos centrales normalizados se calculan como:

$$\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\gamma}}, \gamma = \frac{p+q}{2} + 1, p+q \geq 2 \tag{3,7}$$

Adaptación de la ecuación 3,4 en el dominio discreto de una imagen, estos momentos espaciales son de orden 0 y 1:

$$m_{ji} = \sum_{y}\sum_{x} g(x,y)x^j y^i \tag{3,8}$$

Para el cálculo de todos los momentos de un polígono o forma rasterizada, OpenCv contiene la función *moments*. En la figura 38 se muestra el uso de la función *moments* en Python, el centro del objeto se describe como (C_x, C_y). Puesto que los momentos del contorno se calculan utilizando la fórmula del

teorema de *Green*, puede obtener resultados aparentemente extraños para los contornos con autointersecciones, p. ej. un área cero (m00) para contornos en forma de mariposa.

Figura 39. **Cálculo de momentos en Python**

```
M = cv2.moments(c)
if (M["m00”] ==0): M["m00”] =1
C_x = int(M[“m10”] / (M[“m00”] + 1e-5))
C_y = int(M[“m01”] / (M[“m00”] + 1e-5))
```

Fuente: elaboración propia.

3.1.3.3.3. Identificador

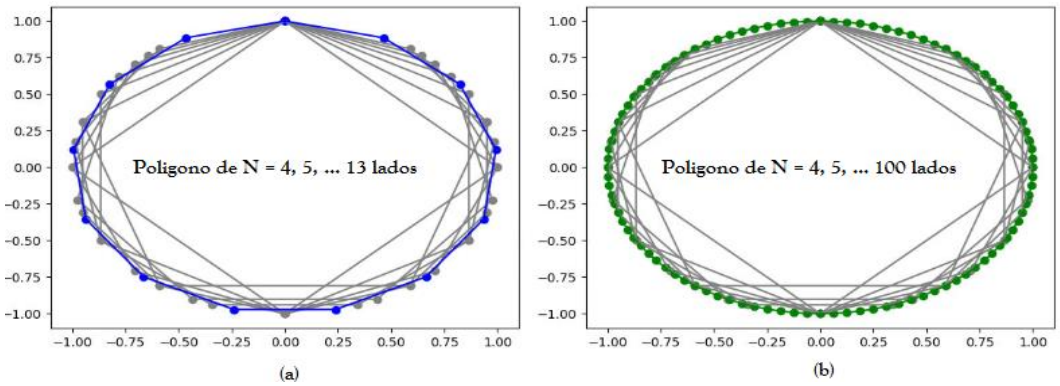
Esta etapa consiste en crear un identificador o una etiqueta que identificará al envase; la etiqueta llevará el nombre PET. Para crear este identificador se utilizará dos condiciones, el área y la cantidad de vértices del objeto.

El cálculo de áreas de contornos se utilizará para delimitar los objetos que se desea detectar. Con ello se consigue que el algoritmo detecte y etiquete únicamente aquellos objetos que estén dentro de un rango de área en específico. Para este caso se quiere que detecte envases PET, por lo que es importante hacer un muestreo de todas las posibles áreas de la tapa del envase.

OpenCv contiene la función *contourArea,* que calcula un área de contorno. De manera similar a los momentos, el área se calcula utilizando la fórmula de Green. Por lo tanto, esta devuelve un número de pixeles distintos de cero. La función seguramente dará resultados incorrectos para contornos con auto intersecciones, lo cual no representa un problema para el prototipo.

Como se observa en la figura 36, el contorno resultante es aproximadamente el de un círculo que coincide con la forma de la tapa del envase PET. Un círculo se representa como un polígono con muchos vértices. En la medida que aumentan los vértices de un polígono, este se aproxima más a un círculo, como se muestra en la figura 40, por lo que la cantidad de vértices se tomó en consideración en el diseño del *software*.

Figura 40. **Polígono de N lados**



Fuente: elaboración propia, empleando Python.

Por tanto, al medir la longitud de la cadena resultante de la función *approxPolyDP* se obtiene la cantidad de vértices. Esto se consigue empleando la función *len*, que entrega la cantidad de elementos que contiene una lista o un array.

Aunando todas las características extraídas se ha conseguido crear un identificador para este prototipo. Una vez las condiciones de área y cantidad de vértices se cumplan, mediante la función *putTex* de OpenCv se pude imprimir un texto en la parte superior del cuadro delimitador. Los parámetros para esta función se detallan en la tabla XIII. En la figura 41 se muestran las funciones descritas anteriormente, *contourArea, len* y *putText* empleadas en Python.

Tabla XIII. **Parámetros de la función *putText***

Parámetro	Descripción
Img	Imagen
Texto	Cadena de texto a dibujar
Ubicación	Ubicación de esquina inferior izquierda de la cadena de texto en la imagen
Fuente	Tipo de fuente para el texto
Color	Color del texto en RGB

Fuente: elaboración propia.

Figura 41. **Funciones empleadas en el identificador**

```
area = cv2.contourArea(contorno)
Vertices = len(aproxContorno)
cv2.putText(frame, "PET", (Cx-20, Cy-20), 1, 1, (0, 255, 0))
```

Fuente: elaboración propia.

3.1.4. Conteo de elementos detectados

Esta etapa consiste en contar la cantidad de envases PET que han sido detectados. Esta variable se utilizó para el control de lubricación en función de la cantidad de elementos detectados en la faja transportadora. El contador se realizó empleando un ciclo *for* y la función *enumerate,* como se ve en la figura 42, para contar cada contorno detectado.

Figura 42. **Conteo de objetos detectados**

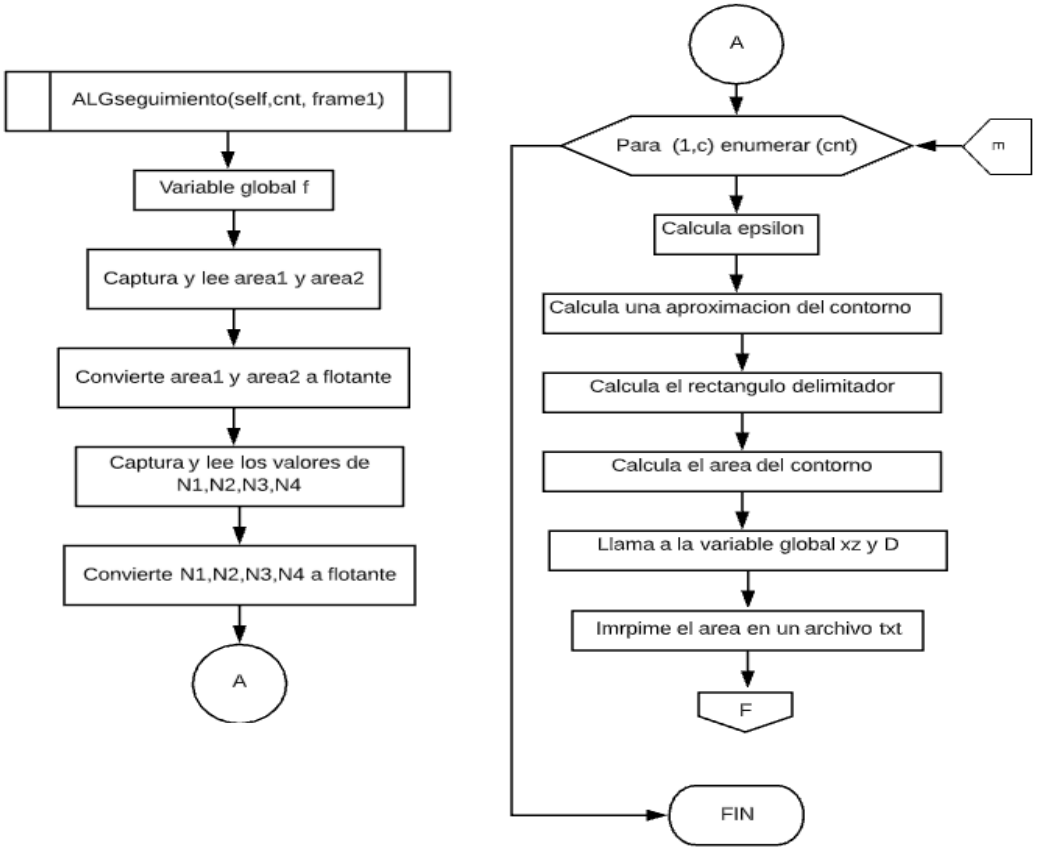
```
for (i, c) in enumerate(contorno):
```

Fuente: elaboración propia.

3.1.5. Diagramas de flujo

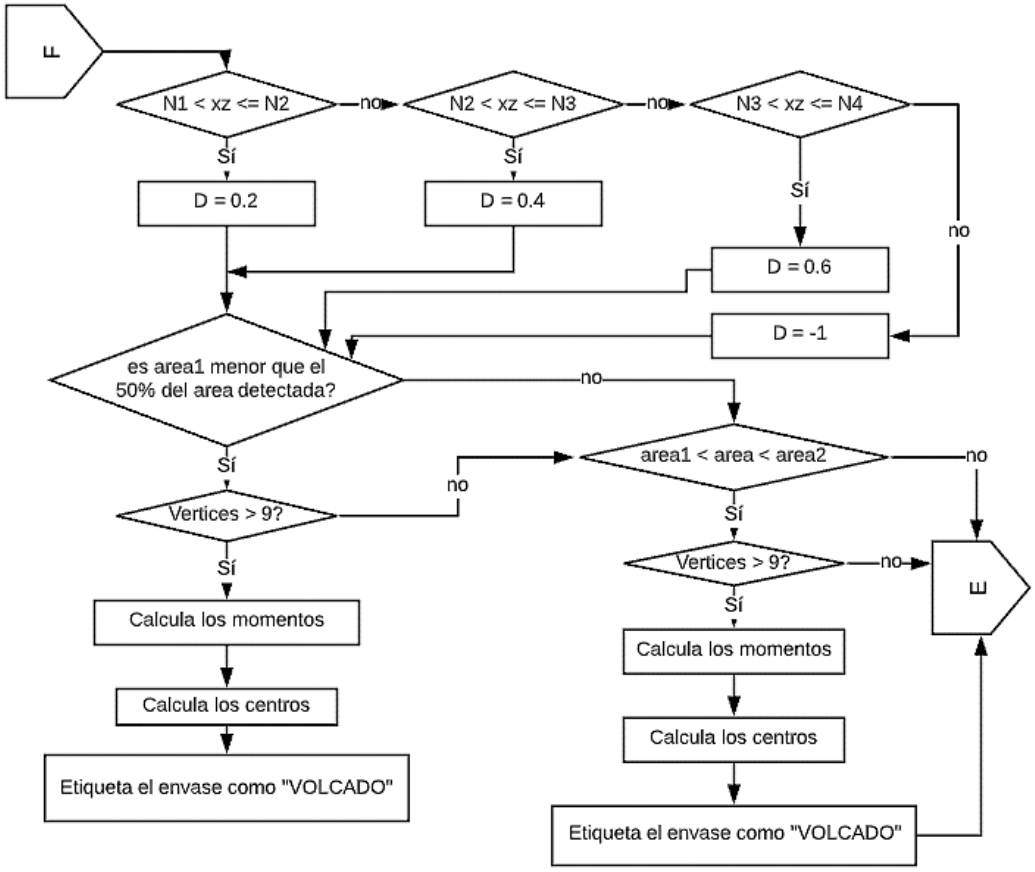
Los diagramas de flujo descritos utilizan símbolos de referencia en página para indicar que el flujo sigue donde se ha colocado el símbolo que contiene la misma letra. También indican que hay una referencia cruzada en otra página de este mismo proceso.

Figura 43. **Algoritmo de seguimiento y conteo de objetos parte A**



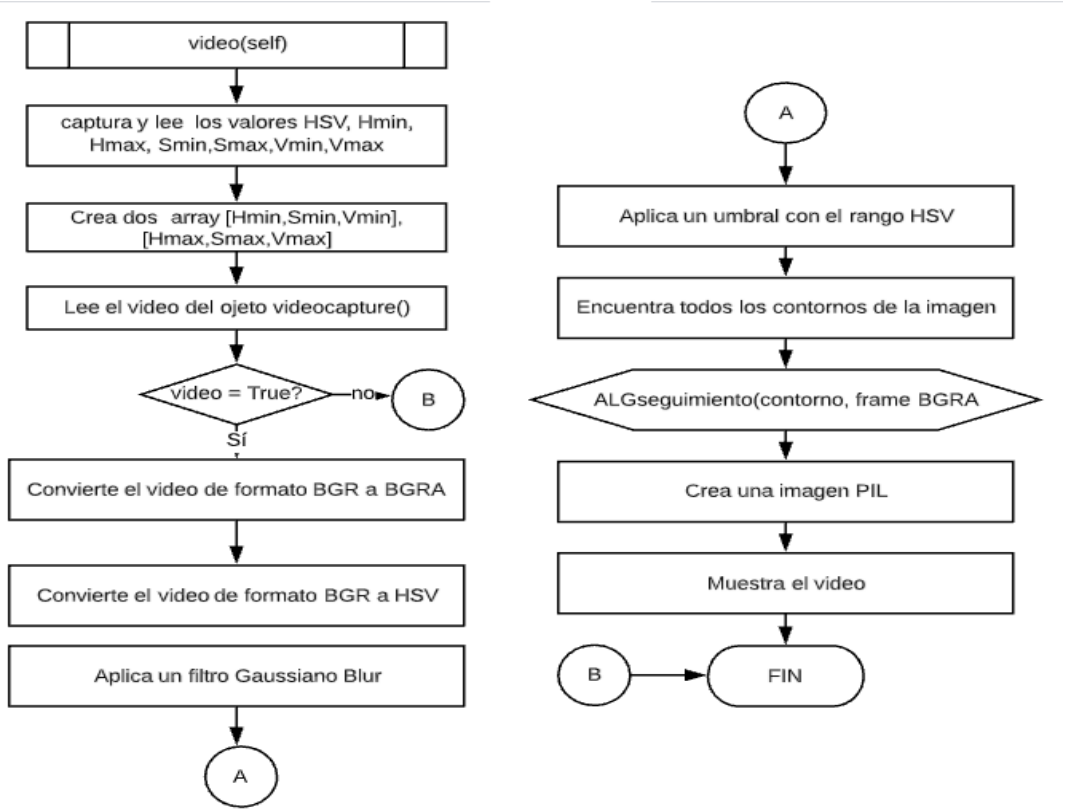
Fuente: elaboración propia, empleando Lucidchart.

Figura 44. **Algoritmo de seguimiento y conteo de objetos parte B**



Fuente: elaboración propia, empleando Lucidchart.

Figura 45. **Diagrama de todas las etapas realizadas en OpenCv**



Fuente: elaboración propia, empleando Lucidchart.

3.2. Diseño de *software* para el control de lubricación en transporte de envases PET

En una línea de envases PET existen varios procesos de envasado en el siguiente orden: soplado de preformas para producir las botellas, etiquetado en vacío, llenado y tapado de envases, formación de *packs* y paletizado. El transporte de envases consta de cadenas y guías laterales. Es importante llevar a cabo una rutina constante de lubricación de las cadenas del transporte para

reducir la fricción entre botellas y la cadena y con ello evitar que los envases PET puedan volcar.

3.2.1. Requisitos del diseño

- Modo de inicio automático: es para inyección continua durante un intervalo de tiempo ajustable. Comúnmente, este modo se utiliza después de limpiar los transportadores, luego cambia a su modo de funcionamiento normal con una dosis cronometrada con un tiempo de encendido y apagado. Normalmente, este tiempo es en segundos y el tiempo de encendido es menor que el de apagado.

- Modo de inicio manual: para estimar el tiempo de encendido y apagado según la demanda para cada línea de transporte, el tiempo debe ser ajustable.

- Electroválvula controlada por tiempo: interruptor de tiempo de encendido y apagado ajustable.

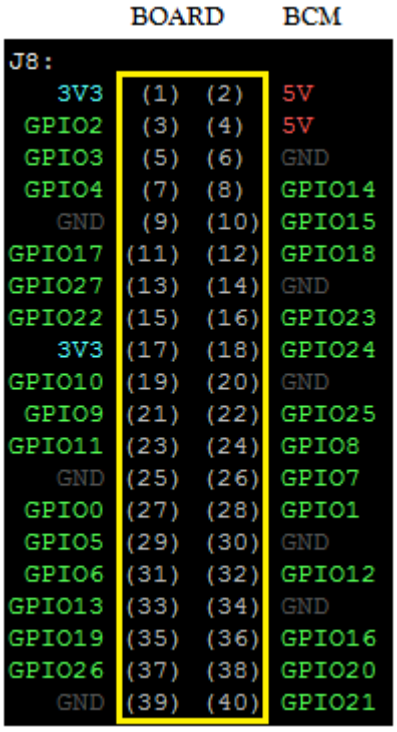
- Bloqueo de la válvula solenoide cuando hay un déficit de producto.

3.2.2. Control de los pines GPIO de la PI

Los GPIO son un arreglo de pines de entrada/salida de propósito general, mediante los cuales es posible activar estados lógicos o interactuar con otras interfaces. La PI envía una señal a 6 salidas de propósito general asignadas a interactuar con la interfaz de potencia. Estas funciones son programadas en un código de lenguaje *Python*, mediante el empleo de *threads*.

Se puede acceder a una referencia útil de los GPIO en la PI empleando el comando *pinout*. Esta herramienta proporciona la biblioteca de *Python* de GPIO *zero*, la cual viene instalada por defecto en la imagen de Raspbian. El resultado de emplear el comando *pinout* se muestra en la figura 46.

Figura 46. **Captura de la salida del comando *pinout***



Fuente: elaboración propia.

Para trabajar con los GPIO en *Python*, es necesario instalar la librería rpio.gpio. Esta biblioteca permite controlar los pines GPIO de la PI. Esta librería se puede instalar empleando el comando *sudo apt-get install python-rpio.gpio.*

Figura 47. **Importando librería rpio. gpio en *Python***

```
import RPi.GPIO as GPIO
```

Fuente: elaboración propia.

Existen dos sistemas de numeración para los pines de la PI, BCM y *BOARD.* En el sistema BCM, a los pines se les da algunos nombres, como se muestra en la figura 46. El pin 10 del PI se llama GPIO 15 y el pin 40 se llama GPIO 21, y así sucesivamente. En el caso de la estructura *BOARD*, los pines están numerados en un patrón de *zig-zag* y pueden usar estos números en los *scripts* de Python, tal como se observa en la figura 46, encerrados en un cuadro amarillo.

Para indicarle a las librerías RPi-gpio que use cualquiera de los dos sistemas de enumeración de pines se utiliza la instrucción *setmode,* tal como se observa en la figura 48.

Figura 48. **Numeración GPIO (BCM) y pin (*BOARD*)**

```
GPIO.setmode(GPIO.BCM)
GPIO.setmode(GPIO.BOARD)
```

Fuente: elaboración propia.

Los GPIO se utilizaron para el control de la interfaz de potencia. Es necesario configurarlos como pines GPIO de salida, para ello se debe utilizar la línea de código *GPIO.setup* como se muestra en la figura 49.

Figura 49. **Configurar un pin como salida**

```
GPIO.setup(pin, GPIO.OUT)
```

Fuente: elaboración propia.

Para controlar el estado de salida de un GPIO es necesario emplear las líneas de código GPIO.LOW y GPIO.HIGH, tal como se muestra en la figura 50.

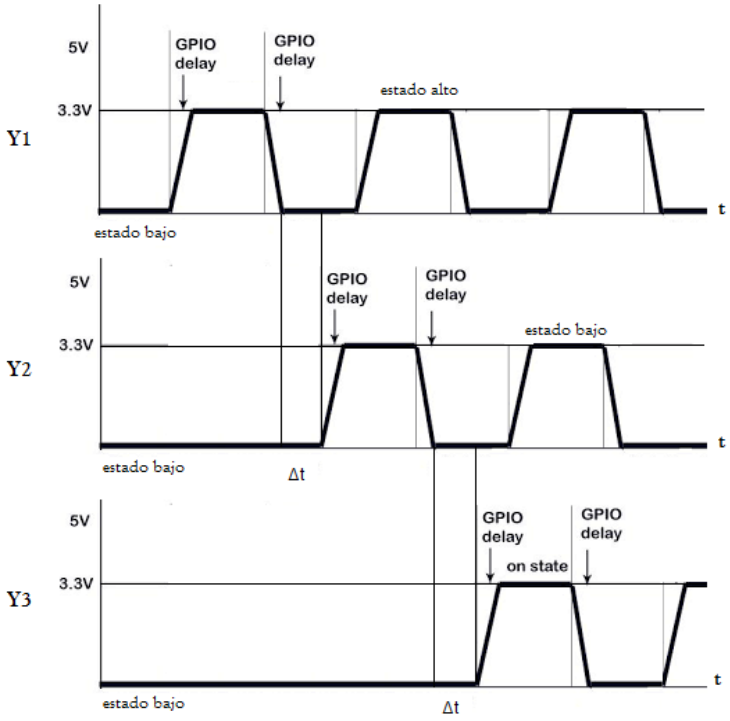
Figura 50. **Control del estado de salida de un GPIO**

```
GPIO.output(pin, GPIO.HIGH)
GPIO.output(pin, GPIO.LOW)
```

Fuente: elaboración propia.

En el diseño del prototipo se consideraron 6 electroválvulas; es decir, por cada electroválvula existe una salida GPIO de la PI. Sin importar el modo de funcionamiento, al iniciar el programa enciende la electroválvula y1, Δt segundos después enciende la electroválvula y2, Δt segundos después enciende la electroválvula y3 y así sucesivamente, hasta que cada electroválvula quede funcionando independientemente. Δt es modificable por el usuario. La lógica del arranque de las electroválvulas se describe en la figura 51.

Figura 51. **Arranque de electroválvulas en secuencia**



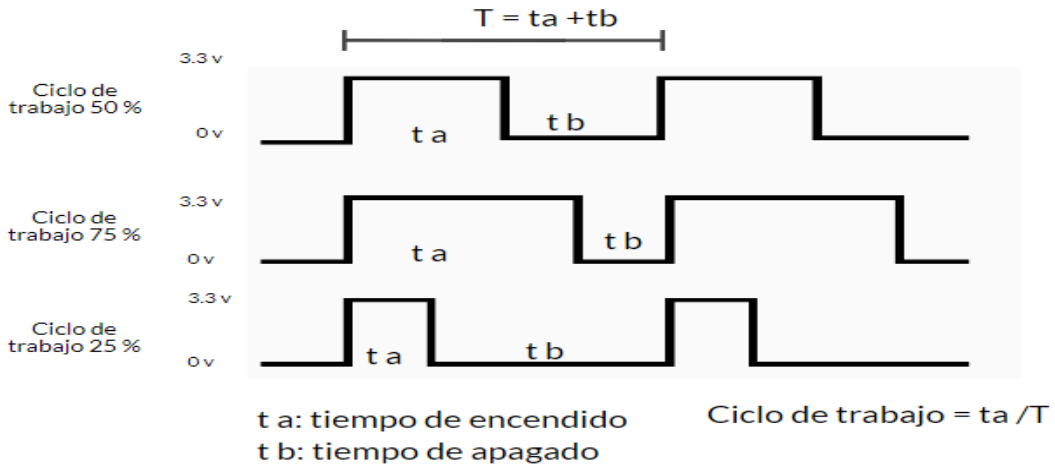
Fuente: elaboración propia, empleando Pixilart.

El tiempo de encendido y apagado es programado por el usuario desde la interfaz gráfica. En la medida que el algoritmo detecte un aumento o decremento de producto, automáticamente se regulará el ciclo de trabajo. De acuerdo con esto, el ciclo de trabajo puede ser representado como:

$$D = \frac{t_a}{T}, \qquad T = t_a + t_b \tag{3,9}$$

En la figura 52 se muestra una gráfica del ciclo de trabajo para cada salida GPIO.

Figura 52. **Ciclo de trabajo de las salidas GPIO**

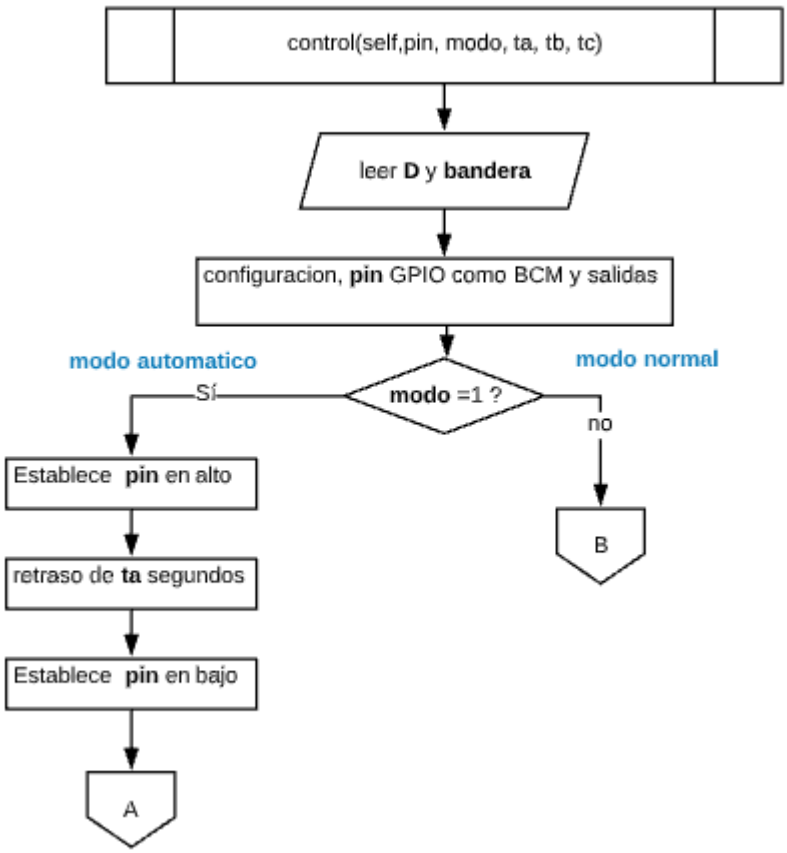


Fuente: elaboración propia, empleando Pixilart.

3.2.3. Diagramas de flujo

Los diagramas para el *software* de control de lubricación se utilizaron símbolos de referencia en otra página para indicar que hay una referencia cruzada en otra página de este mismo proceso.

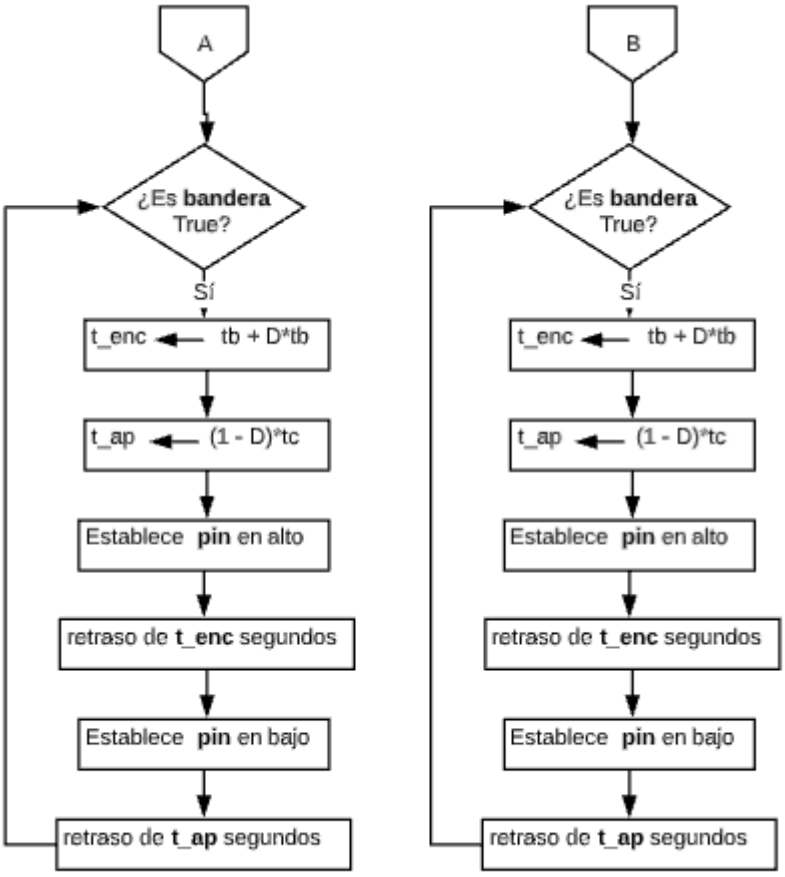
Figura 53. **Algoritmo de control de las electroválvulas parte A**



Variable global D	Esta variable es la encargada de regular el ciclo de trabajo para las electroválvulas, los valores de D se asignan en el algoritmo de seguimiento.
Variable global bandera	Es una variable booleana, encargada de controlar el ciclo *while*, variable inicializada como *True*, al cerrar la ventana principal cambia a *False*

Fuente: elaboración propia, empleando Lucidchart.

Figura 54. **Algoritmo de control de las electroválvulas parte B**



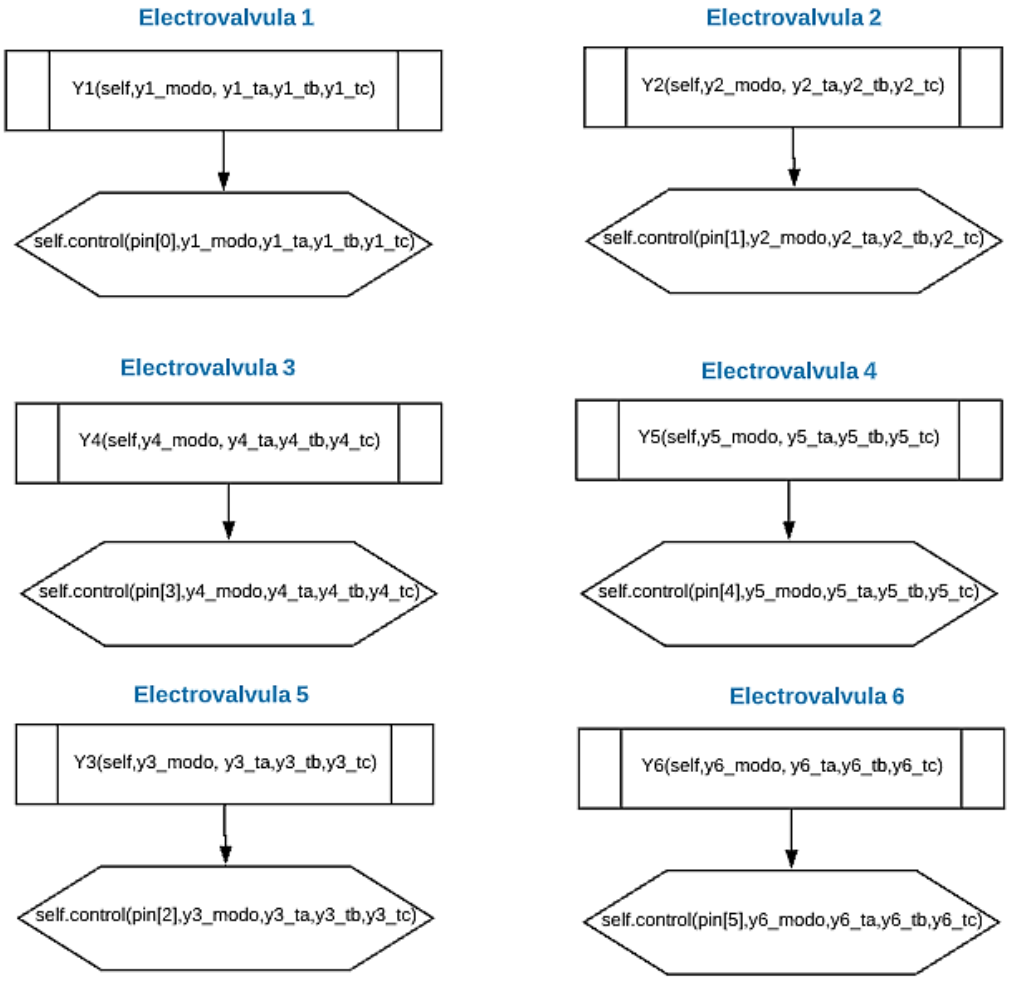
Fuente: elaboración propia, empleando Lucidchart.

Tabla XIV. **Parámetros y variables del algoritmo de control**

Parámetro pin GPIO	pin	Variable local entera
Parámetro modo de operación	modo	Variable local entera
Parámetro tiempo de inyección	ta	Variable local flotante
Parámetro tiempo de encendido	tb	Variable local flotante
Parámetro tiempo de apagado	tc	Variable local flotante
Ciclo de trabajo	D	Variable global flotante,
Activador del bucle *while*	bandera	Variable global booleana

Fuente: elaboración propia.

Figura 55. **Uso del algoritmo de control para cada electroválvula**



Fuente: elaboración propia, empleando Lucidchart.

Tabla XV. **Parámetros para cada función Yn**

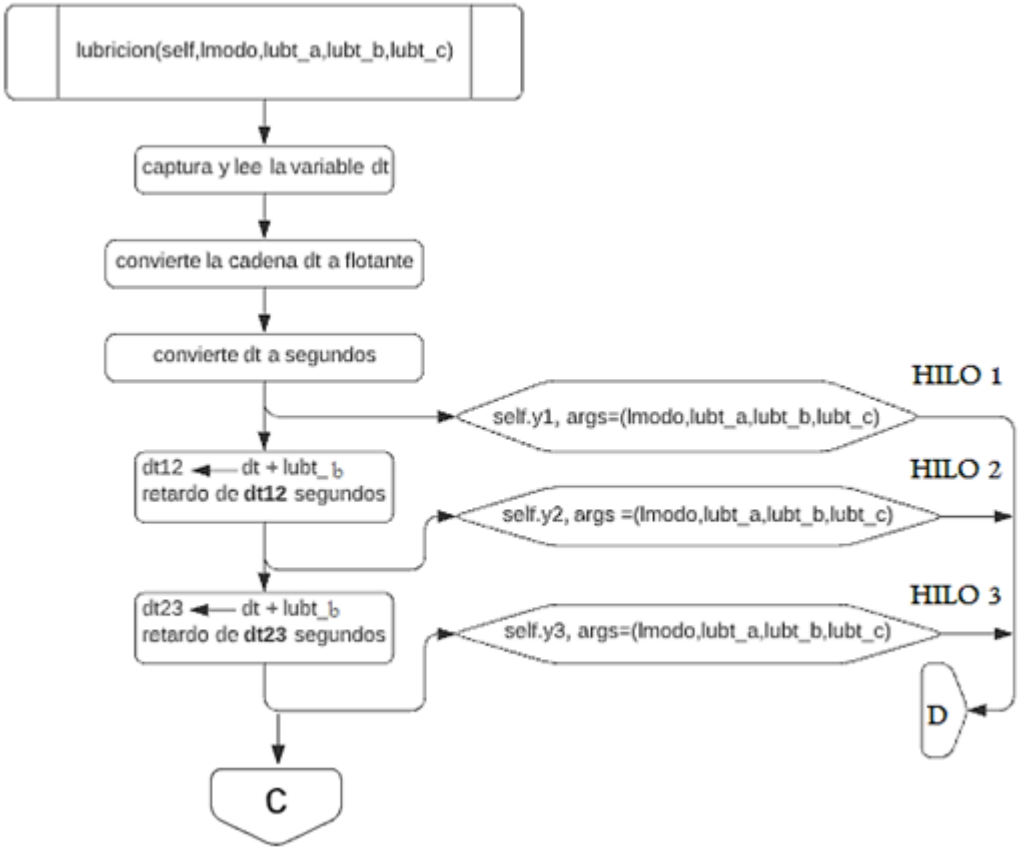
Parámetro modo de operación	**yn_modo**	Variable local entera
Parámetro tiempo de inyección	**yn_ta**	Variable local flotante
Parámetro tiempo de encendido	**yn_tb**	Variable local flotante
Parámetro tiempo de apagado	**yn_tc**	Variable local flotante

Fuente: elaboración propia.

En la figura 53 y 54 se muestra el algoritmo encargado de controlar cada electroválvula de forma independiente. Para ello se empleó *threading* o hilos. Un objeto *thread* representa una determinada operación que se ejecuta como un subproceso independiente.

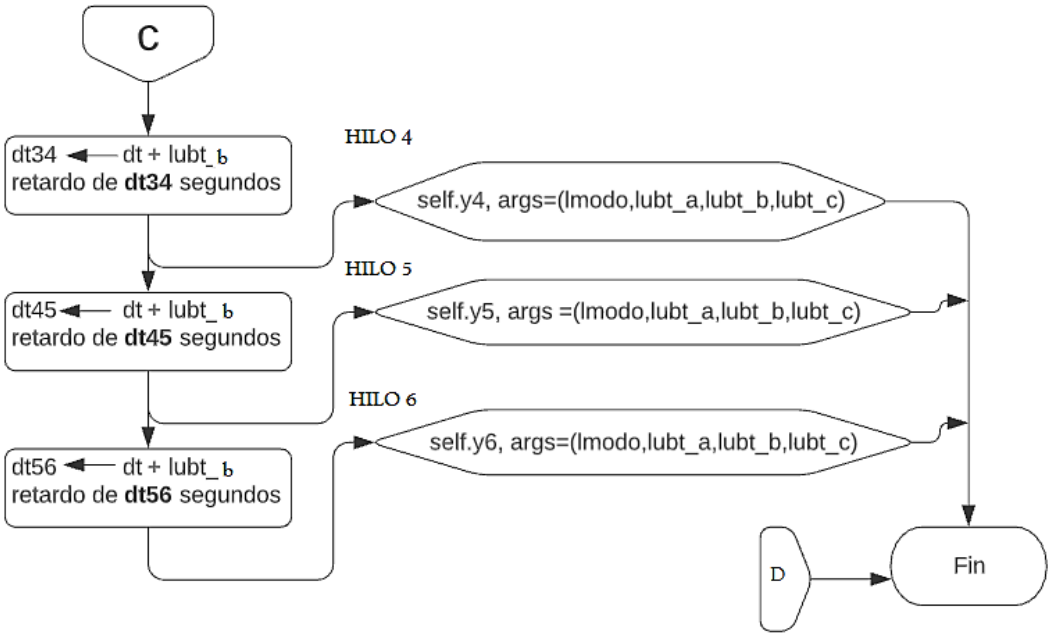
Como se observa en la figura 51, el arranque de las electroválvulas es en secuencia, con Δt segundos después del primer *falling Edge* de cada electroválvula. Esto se consigue inicializando cada *threading* Δt + tb segundos después, hasta que cada electroválvula funcione independientemente.

Figura 56. **Diagrama de flujo de la función lubricación parte A**



Fuente: elaboración propia, empleando Lucidchart.

Figura 57. **Diagrama de flujo de la función lubricación parte B**



Fuente: elaboración propia, empleando Lucidchart.

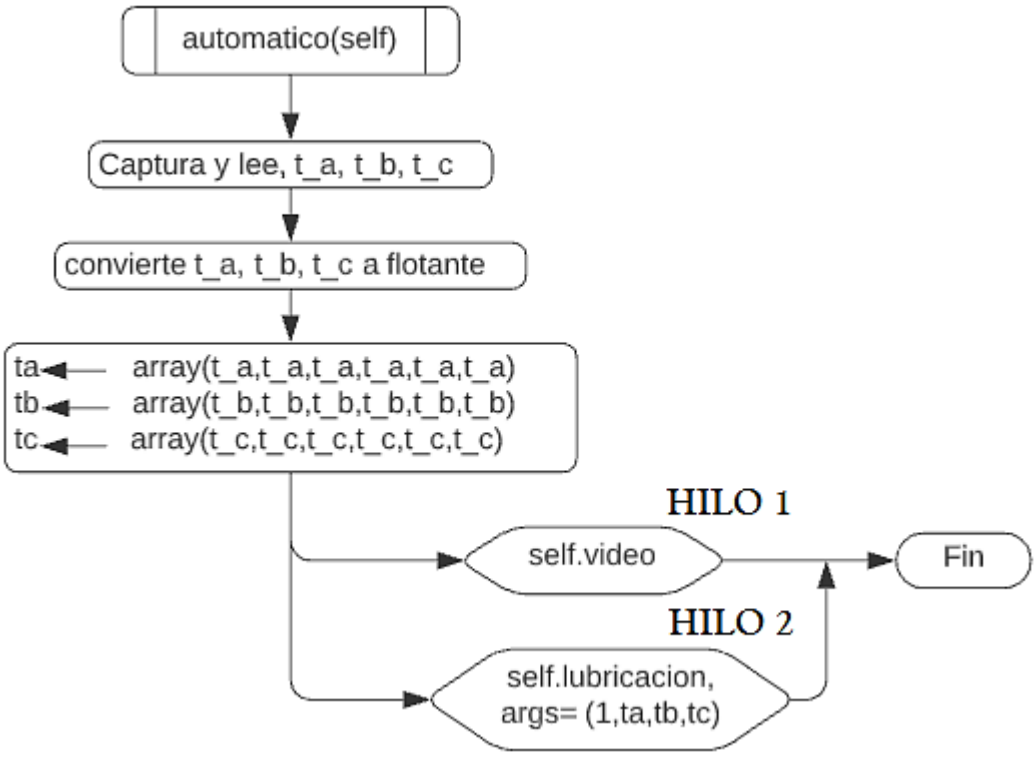
Tabla XVI. **Parámetros para la función lubricación**

Parámetro modo de operación	**lmodo**	Variable local entera
Parámetro tiempo de inyección	**lubt_a**	Variable local flotante
Parámetro tiempo de encendido	**lubt_b**	Variable local flotante
Parámetro tiempo de apagado	**lubt_c**	Variable local flotante
Tiempo entre válvulas para el arranque en secuencia, ej. dt12 (tiempo entre y1 y y2)	**dt12**	Variable local flotante
	dt23	
	dt34	
	dt45	
	dt56	

Fuente: elaboración propia.

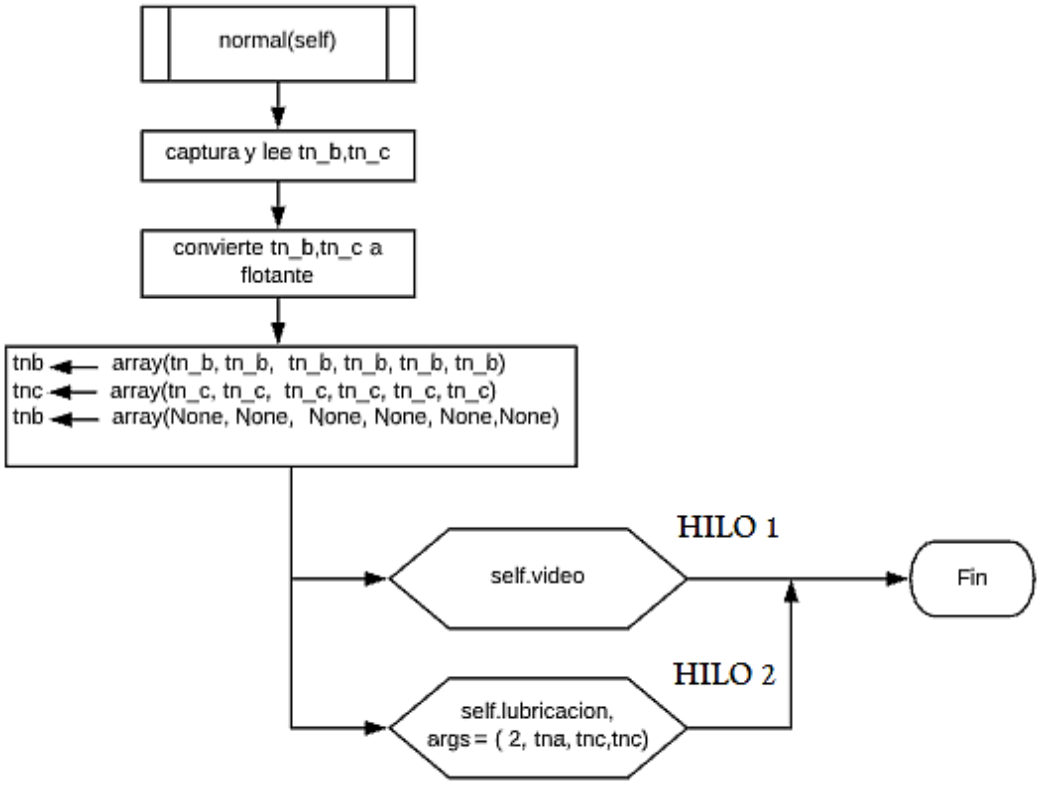
La función modo automático es llamada desde el menú de la interfaz gráfica. Luego de introducir los parámetros desde la interfaz, el usuario debe escoger uno de los tres modos de operación: automático, manual y normal. Los modos de operación habilitan dos *threading*, la función lubricación y el algoritmo de detección, el cual es el encargado de realizar todo el procesamiento de imágenes.

Figura 58. **Función modo automático**



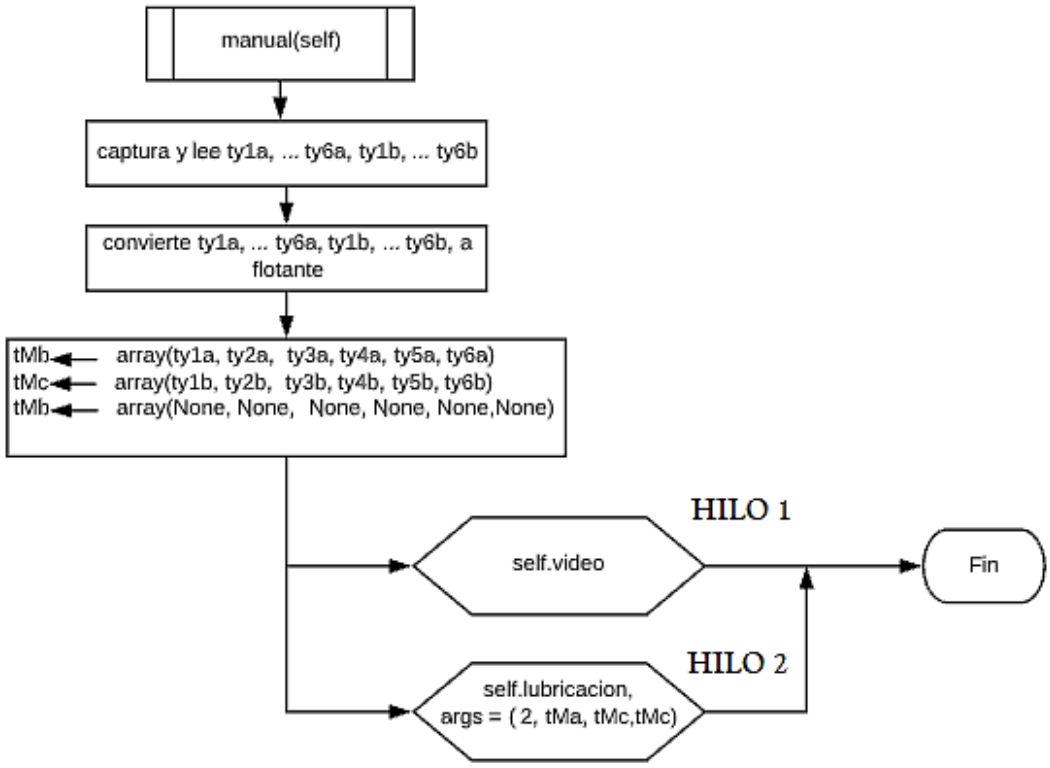
Fuente: elaboración propia, empleando Lucidchart.

Figura 59. **Función modo normal**



Fuente: elaboración propia, empleando Lucidchart.

Figura 60. **Función modo manual**



Fuente: elaboración propia, empleando Lucidchart.

Tabla XVII. **Parámetros de las funciones automático, normal y manual**

Parámetro	Automático	Normal	Manual
1	1	2	2
2	Tiempo de inyección	*None*	*None*
3	Tiempo encendido	Tiempo encendido	tiempo encendido
4	Tiempo apagado	Tiempo apagado	Tiempo apagado

Fuente: elaboración propia.

3.3. Etapa de interfaz de usuario

El módulo Tkinter y la biblioteca de imágenes de Python (PIL), junto con la librería OpenCv, han sido utilizadas en el diseño de la interfaz de usuario. En las figuras 61 a la 64 se muestran la interfaz del usuario.

La interfaz gráfica del sistema está conformada por lo siguiente:

- Cuadro de video: la captura de video en tiempo real con el etiquetado de los elementos detectados.

- Cuadros de texto: en la pantalla principal se muestran 8 cuadros de texto, los cuales se describen de la siguiente forma:

    - Tiempo continuo modo automático.
    - Tiempo encendido modo automático.
    - Tiempo apagado modo automático.
    - Tiempo entre electroválvulas.
    - Tiempo de encendido modo normal.
    - Tiempo de apagado modo normal.
    - Define el límite inferior y superior del área de los envases PET.
    - N1, N2, N3 y N4: por cada N existe un valor predefinido de ciclo de trabajo, dependiendo de la cantidad de elementos que se detecten, N_n representan la cantidad de objetos esperados a detectar en una sección determinada del transporte.

- Botones: en la pantalla principal se muestran 4 botones, un botón para cada modo de operación, modo automático, normal y manual. Para este último se abre una nueva ventana, en ella hay 2 cuadros de texto por cada

electroválvula. Para ingresar los valores de tiempo en milisegundos de forma manual hay un cuarto botón llamado HSV. Este abre una ventana con una imagen de la variación de color HSV para emplearla de guía al momento de definir las máscaras de detección por colores.

Sliders: en la pantalla principal se muestran 6 *sliders*, para configurar los rangos de colores a detectar.

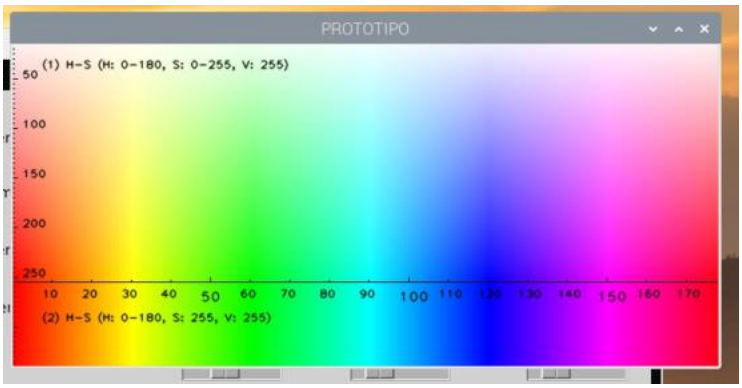
Figura 61. **Interfaz de usuario**

Fuente: elaboración propia.

Figura 62. **Ventana secundaria para el modo manual**

Fuente: elaboración propia.

Figura 63. **Ventana auxiliar de variación de colores**



Fuente: elaboración propia.

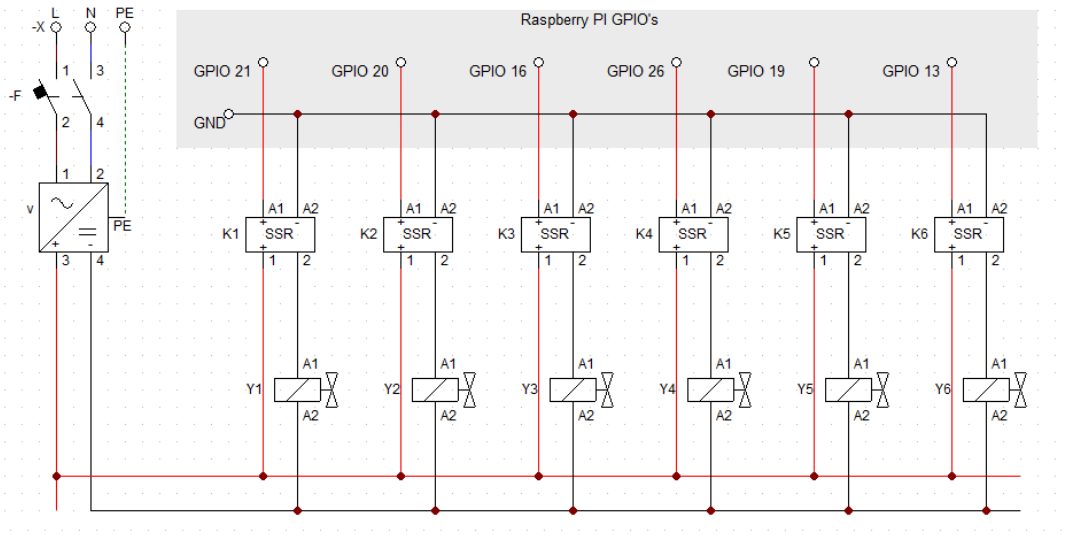
Figura 64. **Ventana principal con captura de video**

Fuente: elaboración propia.

3.4. Diseño de la interfaz de potencia

La interfaz de potencia está constituida por un conjunto de seis relevadores de estado sólido gobernados por cada una de las seis salidas GPIO de la PI. Por cada relé existe una electroválvula encargada de la dosificación de lubricante. El diagrama eléctrico se muestra en la figura 65.

Figura 65. **Diagrama eléctrico de la interfaz de potencia**



Fuente: elaboración propia, empleando CADe SIMU.

3.4.1. Relés de estado sólido

Los relés de estado sólido deben presentar las características que se muestran en la tabla XVIII.

Tabla XVIII. **Especificaciones técnicas requeridas para los relés**

Voltaje de entrada	DC 3 a 32 v
Salida de voltaje	DC 5 a 60 v, AC 24 a 280 V
Corriente de salida	3 a 40 A
Voltaje Min. ON/OFF	ON > 2.4V, OFF < 1.0V

Fuente: elaboración propia.

3.5. Diseño del tablero eléctrico

Los materiales y componentes empleados en el tablero eléctrico se detallan en la tabla XIX.

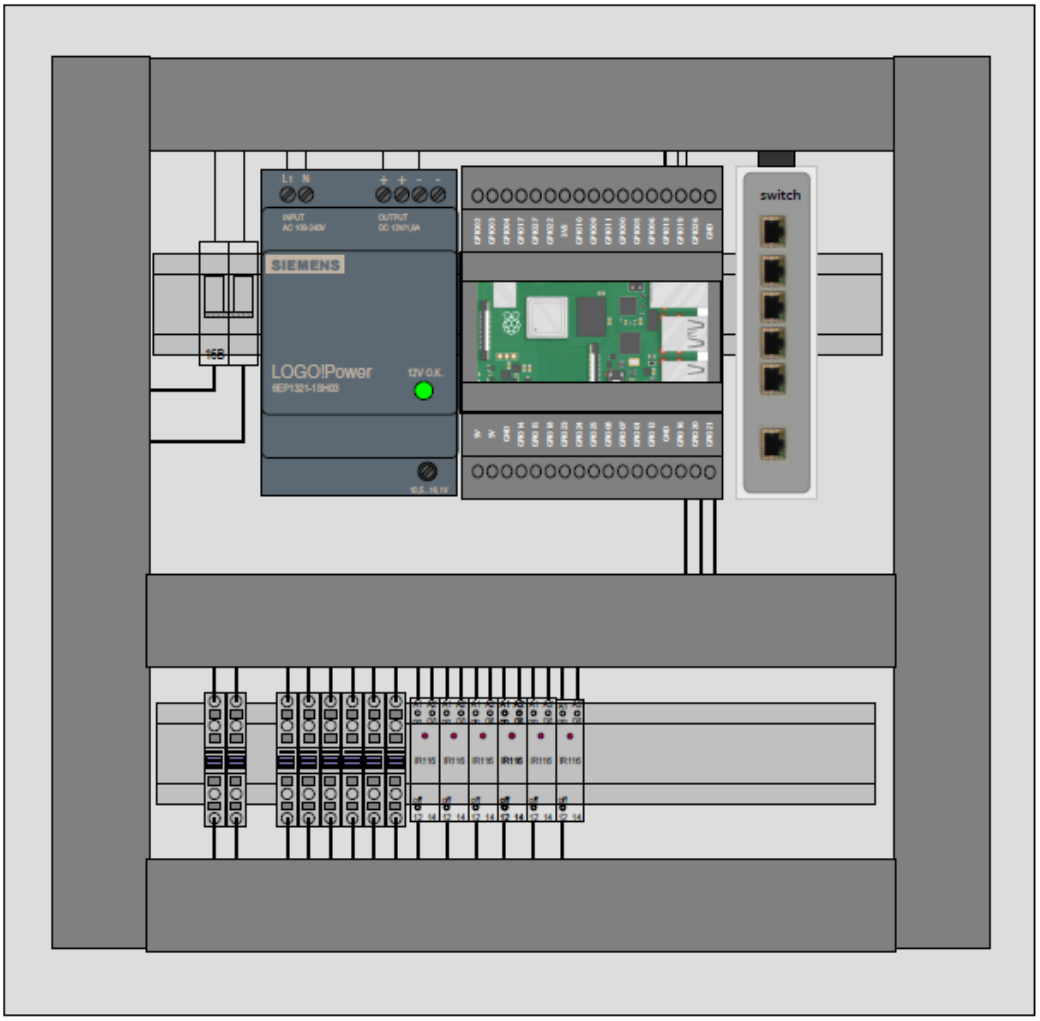
Tabla XIX. **Materiales y componentes en el tablero eléctrico**

Cantidad	Material/ componente
1	Disyuntor bipolar
1	Fuente de alimentación AC ~ DC
8	Borneras de conexiones
6	Relés de estado sólido SSR
1	*Switch* (dispositivo conmutador de red)
1	Riel metálico, 300mm
1	Canaleta de cableado, ranurada de PVC, sección útil 149mm^2
1	Prensa cable
1	Gabinete eléctrico
1	Cable eléctrico color azul, marrón, amarillo-verde
1	Cable de red

Fuente: elaboración propia.

Para el diseño del tablero eléctrico se empleó el *software* ProfiCAD. En la figura 66 se muestra cómo se vería el tablero eléctrico del prototipo.

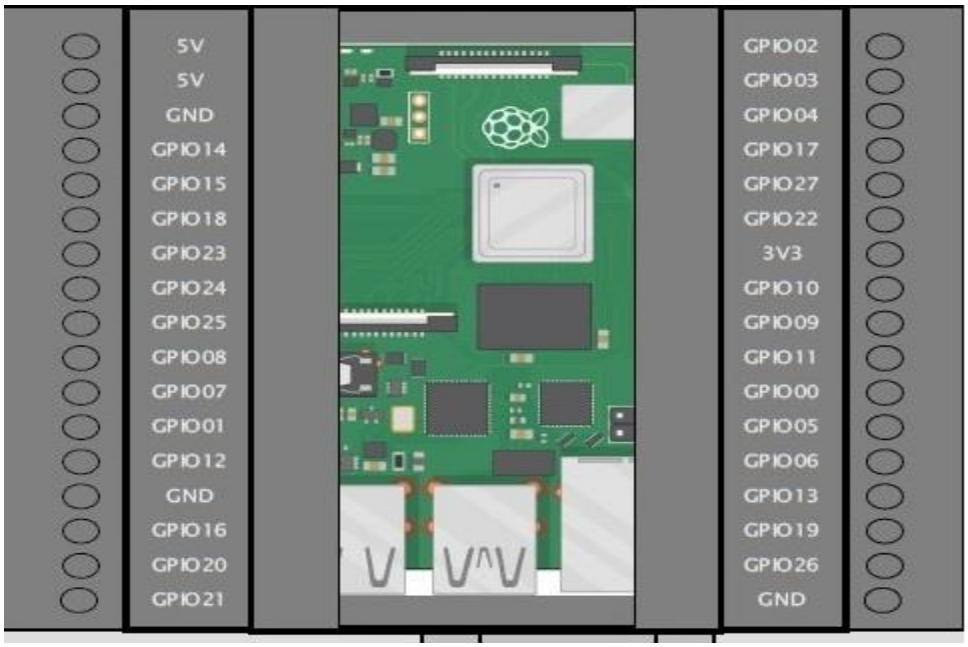
Figura 66. **Tablero eléctrico del prototipo**



Fuente: elaboración propia, empleando ProfiCAD.

Puesto que el diseño del prototipo tiene enfoque industrial, es necesario que la PI esté protegida por un encapsulado industrial, como se muestra en la figura 66. En la figura 67 se muestra únicamente la PI con su encapsulado industrial.

Figura 67. **Encapsulado industrial para la PI**



Fuente: elaboración propia, empleando ProfiCAD.

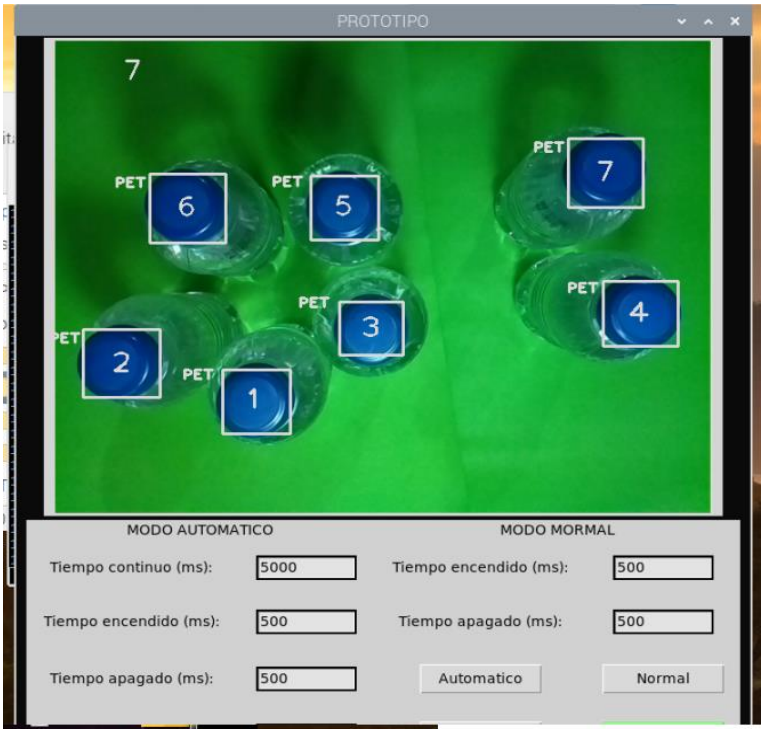
4. ANÁLISIS DE RESULTADOS

Durante el diseño del prototipo se realizaron varias pruebas en las diferentes etapas que componen el prototipo.

4.1. Resultados obtenidos al emplear el algoritmo de detección

Para realizar pruebas del algoritmo de detección se utilizaron siete envases PET.

Figura 68. **Prueba de detección de envases**



Fuente: elaboración propia.

Los parámetros de área mayor que y menor que se obtienen con una prueba de detección de área. Esto se consigue al seleccionar el rango de colores de detección deseado, luego se selecciona cualquier modo de operación. De inmediato en un archivo de texto se imprimen todos los valores de área en pixeles detectados, se inicia nuevamente la aplicación y se introduce el área mínima y máxima obtenida. En la siguiente tabla se muestran los parámetros y resultados obtenidos.

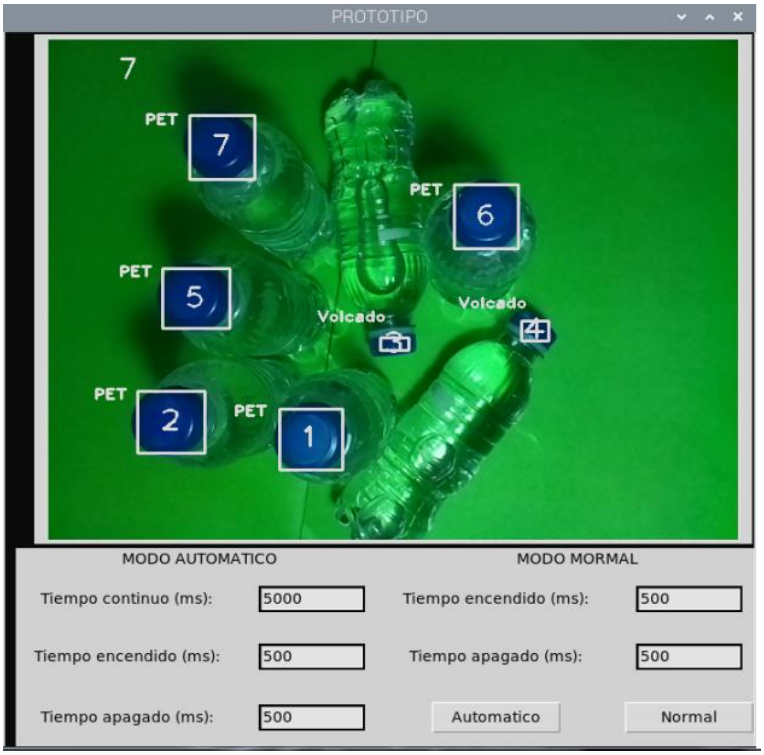
Tabla XX. **Resultados obtenidos en la detección de objetos**

<table>
<tr><th colspan="2">Obtenida del archivo de texto</th><th colspan="2">Definido por el usuario</th></tr>
<tr><th colspan="2">Área (pixeles)</th><th colspan="2">Área (pixeles)</th></tr>
<tr><td>Min</td><td>Max</td><td>Min</td><td>Max</td></tr>
<tr><td>2 200</td><td>3 500</td><td>2 200</td><td>3 500</td></tr>
<tr><th colspan="2">Definido por el usuario</th><th colspan="2">Color detectado</th></tr>
<tr><th colspan="2">Color</th><td colspan="2" rowspan="2">Azul</td></tr>
<tr><td colspan="2">Azul</td></tr>
<tr><th colspan="2">No. envases detectados</th><th colspan="2">No. envases volcados</th></tr>
<tr><td colspan="2">7</td><td colspan="2">2</td></tr>
</table>

Fuente: elaboración propia.

En la siguiente figura se muestra siete envases PET detectados, dos de ellos están volcados.

Figura 69. **Prueba de detección de envases volcados**



Fuente: elaboración propia.

4.2. Resultados obtenidos al emplear el algoritmo de lubricación

En el algoritmo de lubricación, la función control obtiene del algoritmo de detección la variable ciclo de trabajo. Esta es la encargada de modificar los intervalos de tiempo predefinidos por el usuario. En la siguiente tabla se muestran los intervalos de tiempo modificados en función de la cantidad de envases detectados.

Tabla XXI. **Intervalos de tiempo obtenidos**

No.	**Definidos por el usuario**		**Variable D**	**Modificados por D**	
	Tiempo de encendido (ms)	Tiempo apagado (ms)	Ciclo de trabajo (%)	Tiempo encendido (ms)	Tiempo apagado (ms)
1	500	500	20	600	400
2	500	500	20	600	400
3	500	500	20	600	400
4	500	500	20	700	300
5	500	500	20	700	300
6	500	500	60	800	200
7	500	500	60	800	200

Definidos por el usuario		**Valores asignados por el programa**	
Envases esperados a detectar)			**D %**
N1: 0	N2: 3	Mayor que N1 y menor que N2	20
N3: 5	N4: 7	Mayore o iguale que N2 y menor que N3	40
		Mayor o igual que N3 y menor o igual a N4	60

Fuente: elaboración propia.

Para realizar pruebas del arranque entre electroválvulas no hay otra forma más que medir el tiempo de ejecución de cada hilo. En la siguiente tabla se muestra el tiempo de ejecución.

Tabla XXII. **Resultado del tiempo de arranque entre electroválvulas**

Definidos por el usuario						
Tiempo inyección (ms)		Δt (ms)	**Tiempo encendido (ms)**	**Tiempo apagado (ms)**		
4 000		5 000	500	500		
		Tiempo de arranque (s)		**Definidos por el usuario**		
Automático	Y1	t = 0			**Encendido (ms)**	**Apagado (ms)**
	Y2	t = 9,5		Y1	100	150
	Y3	t = 15,0		Y2	200	250
	Y4	t = 20,5		Y3	300	350
	Y5	t= 26,0		Y4	400	450
	Y6	t= 31,5		Y5	500	550
Normal	Y1	t = 0		Y6	600	650
	Y2	t = 4,5				
	Y3	t = 9,0				
	Y4	t = 13,5				
	Y5	t = 18,0				
	Y6	t= 22,5				
Manual	Y1	t = 0				
	Y2	t = 4,1				
	Y3	t = 8,3				
	Y4	t = 12,6				
	Y5	t = 17,0				
	Y6	t = 21,5				

Fuente: elaboración propia.

5. ANÁLISIS FINANCIERO

El análisis financiero de la inversión del prototipo será estimado con la fórmula para el cálculo de coste total de propiedad de Eduardo Pierdant.

Tabla XXIII. **Costo de adquisición de hardware**

Concepto	**Cantidad**	**Costo unitario**	**Costo total**
Raspberrypi	1	Q. 595,00	Q. 595,00
Cámara	1	Q. 200,00	Q. 200,00
		Total, *hardware*	**Q. 795,00**

Fuente: elaboración propia.

En la siguiente tabla se muestra los costos de los componentes utilizados en el tablero eléctrico.

Tabla XXIV. **Costo de adquisición de materiales para el gabinete eléctrico**

Concepto	**Cantidad**	**Costo unitario**	**Costo total**
Disyuntor bipolar	1	Q. 186,00	Q. 186,00
Fuente de alimentación	1	Q. 280,00	Q. 280,00
Borneras	8	Q. 14,00	Q. 112,00
Relés de estado solido	6	Q. 65,00	Q. 390,00
Gabinete	1	Q. 300,00	Q. 300,00
Riel Din metálico			Q. 160,00
Carcasa Raspberry Pi para montaje a Riel Din	1	Q. 289,00	Q. 289,00
Otros			Q. 150,00
		Total, costo del tablero	**Q. 1 867,00**

Fuente: elaboración propia.

En la siguiente tabla se muestra un estimado del costo del software.

Tabla XXV. **Costo de adquisición de software**

Concepto	Cantidad	Costo unitario	Costo total
Diseño de la ampliación	1	Q. 6 000,00	Q. 6 000,00
Mantenimiento de la aplicación	1	Q. 1 500,00	Q. 1 500,00
		Total, *software*	**Q. 7 500,00**

Fuente: elaboración propia.

En la siguiente tabla se muestra los costos iniciales para la instalación del prototipo. Se considera la mano de obra en horas de tres técnicos.

Tabla XXVI. **Costos iniciales para instalación**

Costo por hora para servicios de instalación	Q. 23,00
Numero de técnicos	3
Tiempo aproximado e instalación	48 horas
Total, de costos de instalación	**Q. 3 312,00**

Fuente: elaboración propia.

Tabla XXVII. **Costos de configuración**

Costo por hora para servicios de configuración	Q. 23,00
Numero de técnicos	1
Tiempo aproximado e instalación	24 horas
Total, de costos de configuración	**Q. 552,00**

Fuente: elaboración propia.

En la siguiente tabla se desglosan los costos iniciales del prototipo.

Tabla XXVIII. **Costos iniciales**

Costo de adquisición de hardware	Q. 795,00
Costo de adquisición de materiales para el gabinete eléctrico	Q. 1 867,00
Costo de adquisición de software	Q. 7 500,00
Costos iniciales de instalación	Q. 3 212,00
Costos de configuración	Q. 552,00
Total, costos iniciales	**Q. 13 926,00**

Fuente: elaboración propia.

En la siguiente tabla se muestra los costos de administración, que involucran costos anuales del personal.

Tabla XXIX. **Costo de administración**

Costo anual de un ingeniero	Q. 13 8240,00
Número de ingenieros	1
Cantidad de años en funcionamiento	2
Tiempo dedicado a la solución en porcentaje	20 %
Total, costos de administración	**Q. 55 296,00**

Fuente: elaboración propia.

En la siguiente tabla se muestran los costos de operación por mantenimiento y reparación anual.

Tabla XXX. **Costo de operación**

Número de incidentes promedio por año	10
Cantidad de años en funcionamiento	2 años
Costo por hora de servicio	Q. 60,00
Tiempo aproximado para reparar falla	5 horas
Total, de costos de operación	**Q. 600,00**

Fuente: elaboración propia.

En la siguiente tabla se desglosan los costos de servicio a usuario, debido al soporte a los usuarios finales.

Tabla XXXI. **Costo de servicio a usuario**

Número de incidentes promedio por año	48
Número de años de la solución en funcionamiento	2
Costo por hora de soporte	Q. 36,00
Tiempo aproximado para dar el soporte requerido	2 horas
Total, costos de servicio a usuario	**Q. 6 912,00**

Fuente: elaboración propia.

Tabla XXXII. **Costo de la solución**

Costo inicial	Q.13 637,00
Costo de administración	Q. 55 296,00
Costo de operación	Q. 600,00
Costo de servicio a usuario	Q. 6 912,00
Total de costos de la solución	**Q. 76 445,00**

Fuente: elaboración propia.

CONCLUSIONES

1. Se desarrollaron los temas fundamentales de una faja transportadora, tipos de lubricación y tipos de envases para bebidas.

2. Como consecuencia de lo expuesto en este trabajo de graduación, se presentó a la computadora de tamaño reducido Raspberry Pi como una alternativa para la automatización industrial.

3. Dentro del análisis de resultados, se logró diseñar un algoritmo eficiente para el procesamiento de imágenes en tiempo real en la detección de envases PET.

4. Se diseñó un algoritmo en el lenguaje de programación Python para la toma de decisiones en el control de lubricación.

5. El análisis financiero pone de manifiesto la rentabilidad económica que supone el prototipo, puesto que los costes son realmente mínimos. Además de los bajos costes, el prototipo pretende reducir gastos de operación y pérdidas de producción.

RECOMENDACIONES

1. El manejo de la interfaz de usuario lo puede realizar un técnico con conocimientos de informática y computación; sin embargo, se recomienda que la interfaz sea manipulada únicamente por un ingeniero con conocimientos afines a la ingeniería electrónica.

2. Para mejorar la calidad de las imágenes en el procesamiento se debe utilizar correcta iluminación frontal, adecuada al ambiente en el que se encuentra la cámara.

3. Para que los distintos parámetros que se introducen desde la interfaz de usuario sean óptimos, se recomienda realizar diferentes pruebas de monitoreo hasta conseguir los valores esperados.

4. La temperatura de la Raspberry Pi es un factor importante por considerar. Se recomienda mantener el gabinete en un cuarto frío o implementar una adecuada ventilación para mantener condiciones óptimas de temperatura y, con ello, prolongar la vida útil de la plataforma.

5. Para implementar el gabinete eléctrico del prototipo es preciso regirse por la norma NEMA 250.

BIBLIOGRAFÍA

1. CERVERA FANTONI, Angel Luis. *Envase y embalaje.* Madrid, España : ESIC, 2003. ISBN: 84-7356-339-5. 293 p.

2. FLUSSER, Jan, SUK, Tomáš y ZITOVÀ, Barbara. *Moments and Moment Invariants in Pattern Recognition.* Reino Unido : WILEY, 2009. ISBN: 978-0470-69987-4. 312 p.

3. GAY, Warren. *Raspberry Pi Hardware Reference.* Estados Unidos: APRESS, 2014. ISBN: 978-1-4842-0799-4. 213 p.

4. GERVASO, Mario Salinero. *Diseño de una banda transportadora mediante guide Matlab.* España : Universidad Carlos III de Madrid, 2013. 199 p.

5. ALFACREE, Gareth. *Raspberry Pi beginner´s guide.* Reino Unido : Raspberry Pi PRESS, 2018. ISBN: 9781912047680. 241 p.

6. INTRALOX. *Manual de Ingenieria Bandas transportadores modulares plastico*. [en línea]. <https://www.intralox.com/>. [Consulta: 12 de febrero de 2020]. 386 p.

7. KURNIAWAN, Agus. *Raspbian OS Programming with the Raspberry Pi: IoT Projects with Wolfram Mathematica and Scratch*. Indonesia : Apress, 2018. ISBN: 9798-1-4842-4212-4. 196 p.

8. MASKEPACK. *Catalogo transportadores cintas*. [en línea] <http://www.maskepack.com/>. [Consulta: 12 de febrero de 2020].

9. MEDIN, Roxana y MEDIN, Silvina. *Alimentos introducción técnica y seguridad.* Argentina : AKADIA, 2011. ISBN: 978-9-87-947373-3. 198 p.

10. MONK, Simon. *Raspberry Pi Software and Hardware Problems and Solutions.* Estados Unidos : O'Reilly, 2016. ISBN: 978-1-449-36522-6. 510 p.

11. OSPINA ARIAS, Juan Carlos. *Fundamentos de Envases y Embalajes.* Colombia: Educosta, 2015. ISBN: 978-958-8921-21-1. 174 p.

12. PAJANKAR, Ashwin. *Raspberry Pi Image Processing Programming.* India: APRESS, 2017. ISBN: 978-1-4842-2731-2. 145 p.

13. YANG, Luren y ALBREGSTEN, Fritz. *Fast and Exact Computation of Moments Using Discrete Green's Theorem*. [en línea] <http://hdl.handle.net/10852/9540>. [Consulta: 03 de marzo de 2020].