SIMULACION SISTEMA DE CLASIFICACION BASADO EN VISIÓN ARTIFICIAL

ERIK GREGORIO VELÁSQUEZ HERNÁNDEZ
Código: 1.110.501.630

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGIA E INGENIERIA
INGENIERIA ELECTRONICA Y CONTROL INDUSTRIAL
IBAGUE
2014

SIMULACION SISTEMA DE CLASIFICACION BASADO EN VISIÓN ARTIFICIAL

ERIK GREGORIO VELÁSQUEZ HERNÁNDEZ
Código: 1.110.501.630

Asesor Metodológico
NOEL JAIR ZAMBRANO

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGIA E INGENIERIA
INGENIERIA ELECTRONICA Y CONTROL INDUSTRIAL
IBAGUE
2014

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

Director del proyecto

_____

Firma del Jurado

_____

Firma del Jurado

Ibagué, mayo, 2014

INFORMACIÓN GENERAL DEL PROYECTO

Fecha: 1 de Mayo del 2014
Título: Simulación sistema de clasificación basado en visión artificial.
Investigador principal: Erik Gregorio Velásquez Hernández. Nivel de capacitación del investigador Principal: muy bueno. E-mail erikgvh@hotmail.com Teléfono: 3142344206 Número de cedula de ciudadanía: 1.110.501.630 Nombre del Grupo de Investigación: Línea de Investigación Automatización y herramientas lógicas. Red de Investigación: innovación tecnológica. Escuela: Escuela de ciencias básicas, tecnología e ingeniería. Programa Académico: Ingeniería electrónica.
Zona UNAD de procedencia del proyecto: CEAD Ibagué Tolima Director zonal: Gloria Isabel Vargas Hurtado Dirección postal: Teléfonos: **(8) 2654385 - 2658107 - 2658337 - 2658380 - 2658287** E-mail: Ciudad: Ibagué. Departamento: Tolima.
Duración del proyecto (meses): 3 meses.
Tipo de proyecto: automatización y control ï Clasificación por visión artificial.
Valor de la Financiación solicitada: 2.495.050
Descriptores palabras claves: Visión, artificial, inteligencia artificial, clasificación, comercialización.

RESUMEN

En el presente proyecto se diseñara e implementara un sistema de clasificación de objetos por medio de visión artificial para realizar su debido control, el sistema está compuesto por dos bloques que son el hardware que mediante una tarjeta de adquisición de datos desarrollada y controlado por el microcontrolador PIC18F4550 con comunicación USB, esta interface se encarga del control del motorreductor de la banda transportadora y el motor paso a paso encargado del movimiento de los contenedores, dicha interface tiene como entrada la señal del sensor óptico de barrera encargo de detectar el paso de los objeto, el siguiente componente importante es el software en el cual mediante las toolbox de MatLab como son Image Acquisition se controlara la cámara y la toolbox Image Processing se encarga de realizar el procesamiento digital de las imágenes, la programación de la interface de adquisición de datos se realizara por medio del software PIC C Compiler para la programación del microcontrolador en lenguaje C.

El funcionamiento básico del sistema es emplear una banda transportadora para realizar el transporte de los objetos de los colores básicos que componen en modelo RGB que son rojo verde o azul, los cuales serán detectados al paso del sensor óptico de barrera y de esta forma capturar la imagen con la cámara, la cual será procesada por el software de ingeniería MatLab en tiempo real y de esta forma clasificarla en alguno de los contenedores según el tipo de clasificación que se desee realizar en el momento ya sea por forma o color.

Los sistemas de visión artificial efectúan tareas repetitivas con precisión y rapidez y permiten trabajar fuera del espectro visible distinguiendo detalles no visibles por el ojo humano y aportando numerosos beneficios, siendo los más inmediatos el incremento de la calidad y del rendimiento de la producción y la reducción de costes de mano de obra.

ABSTRAC

In this project we design and implement a system of classification of objects by means of artificial vision for its proper control , the system is composed of two blocks are the hardware via an acquisition card data developed and controlled by the microcontroller PIC18F4550 USB communication , this interface is responsible for controlling the gear motor and conveyor stepper motor responsible for movement of containers, such interface has as input the signal from the optical sensor to detect custom barrier over the object the next important component is the software in which by the MatLab toolbox like camera Image Acquisition and Image processing toolbox is responsible for performing digital image processing were controlled , the programming interface for data acquisition conduct by the PIC C Compiler software for programming the microcontroller in C

The basic operation of the system is to use a conveyor belt to transport the objects that compose the basic colors in RGB model that are red green or blue, which will be detected by the optical sensor step barrier and thus capture the image with the camera, which will be processed by MatLab software engineering real-time and thus classify it in one of the containers by type of classification is desired at the time either by shape or color.

The machine vision systems perform repetitive tasks with precision and speed, allowing work outside the visible spectrum distinguishing details not visible to the human eye and bringing many benefits, the most immediate increase in the quality and efficiency of production and reduced cost of labor.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

CAPITULO I

1. GENERALIDADES DE LA INVESTIGACION

1.1 Introducción

Según Aboudara C., Nielsen I., Huang JC., Maki K., Miller AJ., Hatcher D. (2009)

> La visión artificial o visión asistida por computadora es uno de los subcampos de la inteligencia artificial, cuyo propósito es realizar la programación de un computador para que este logre ñentenderò las características de una imagen o escena. La visión artificial consiste en la captación de imágenes mediante cámaras con sensores CCD, por sus siglas en inglés: "Charge Coupled Device" ósea "Dispositivo de Carga Acoplada", contiene pequeños circuitos electrónicos que transforman la luz captada por la cámara en electricidad. (p. 468)

Otro tipo de sensor es el CMOS es un tipo de sensor que es normalmente 10 veces menos sensible que sensor de CCD, esto para su posterior tratamiento mediante técnicas de procesamiento digital de imágenes, permitiendo de esta forma intervenir en el proceso para realizar el control de calidad o que se lleven a cabo control para la modificación del producto.

Las grandes ventajas de los sistemas de clasificación por visión artificial es que pueden efectuar tareas repetitivas con rapidez y precisión, permitiendo distinguir detalles que no son visibles por el ojo humano, lo que conlleva rendimiento en la cadena de producción, mejorar la calidad y una de las más importantes a nivel empresarial reducción de costos en la mano de obra, ya que permite el funcionamiento las 24 horas del día, evitando los errores humanos.

Los componentes que hacen parte de un sistema de visión artificial son sistema de iluminación ya sea Led, láser, fluorescente, etc. La cámara encargada de la captura de imágenes, Tarjeta de adquisición de datos, Procesamiento digital de imágenes mediante un software el cual mediante algoritmos se encargue de procesar, filtrar, segmentación y reconocimiento de forma, y de clasificación de los objetos en la imagen.

1.2 Definición del Problema

¿Cómo resolver las perdidas en la industria debido al rechazo de lotes de producto terminado, que se deben a la incorrecta clasificación del producto que no cumple con los estándares establecidos?

Para solucionar esto se plantea la automatización y debido a ya que la automatización de procesos es una tendencia natural del progreso tecnológico, ya sea para disminuir tiempos de producción o para mejorar la calidad del dicho proceso, se plantea una solución mediante un sistema de clasificación de objetos mediante visión artificial que permita la clasificación de objetos según su color o forma (variables), y según esto ubicarlos en distintas posiciones o ubicaciones, en un contendor.

Los problemas a solucionar serían los siguientes:

- Evitar que objetos que no cumplan con los estándares de producto terminado de la empresa lleguen al cliente, con lo cual se reducen los costos que debe asumir la empresa del traslado de dicho producto.

- Crear algoritmos de control que prevean condiciones de peligro y realicen paradas automáticas de la planta en caso de emergencia.

- Reducir los errores humanos.

Una de las consecuencias que se presentan cuando el cliente recibe productos que no corresponden al que solicito o que el producto no cumpla con los estándares de producto terminado, es que el cliente pierde la confianza en adquirir nuevamente un producto con la empresa, aunque se le realice el cambio del mismo ya que para una próxima ocasión se producirá desconfianza para adquirir nuevamente algún producto con la empresa.

1.3 Justificación

Este proyecto es necesario en la industria en una línea de producción continua, para optimizar y evaluar la calidad de la producción separando los productos que no cumplan con los parámetros establecidos de fabricación, debido estricto control y certificación de calidad del exigente mercado nacional o internacional que se solicita actualmente.

Gracias a los avances desarrollados hasta el momento se logrado facilitar el proceso de clasificación de objetos empleando sistemas inteligentes, que realicen este tipo de tareas por el hombre, para que de esta forma se reduzca al mínimo los errores que se presentan al revisar cada uno de los elementos u objetos, que lo hacen un proceso tedioso.

La utilidad que se espera del desarrollo de este proyecto es que permita optimizar la clasificación de objetos según las variables establecidas, y de esta forma emplearlo en diferentes campos en los que se requiera clasificar objetos dependiendo del color o forma, realizando ajusto de alto costo como emplear una cámara de mayores prestaciones, empleando motores adecuados para mover objetos de mayor tamaño.

El sistema está compuesto por dos bloques fundamentales que son el hardware y software.

En esta investigación, se utilizan como herramientas sensores, motores, microcontrolador, el software de ingeniería MATLAB, el software PIC C Compiler para la programación del microcontrolador, cámara y un PC.

La aplicación del proyecto en el campo real seria en la clasificación de frutas como mango, manzanas, naranjas, peras, etc. Clasificación de monedas

1.3 Objetivos

1.3.1 Objetivo general

Implementar un sistema de selección a escala que permita la clasificación dependiendo de sus características.

1.3.2 Objetivos Específicos:

- Diseñar un sistema de control y clasificación de objetos mediante la implementación de un sistema de visión artificial.
- Minimizar los costos de operación del proceso mediante la implementación del sistema.
- Realizar inspecciones de objetos sin contacto físico.
- Realizar inspecciones en procesos donde existe diversidad de piezas con cambios frecuentes de producción.

CAPITULO II
2. DESARROLLO TEÓRICO

2.1 MARCO TEORICO

2.1.1 Visión Artificial o Visión asistida por computadora (concepto específico).

Según Valderrama Gutiérrez, Freddy F. Modulo Académico Robótica. (2010).

> La visión artificial, también conocida como visión por computador (del inglés computer vision) o visión técnica, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen. (p.15)

Los objetivos típicos de la visión artificial incluyen:

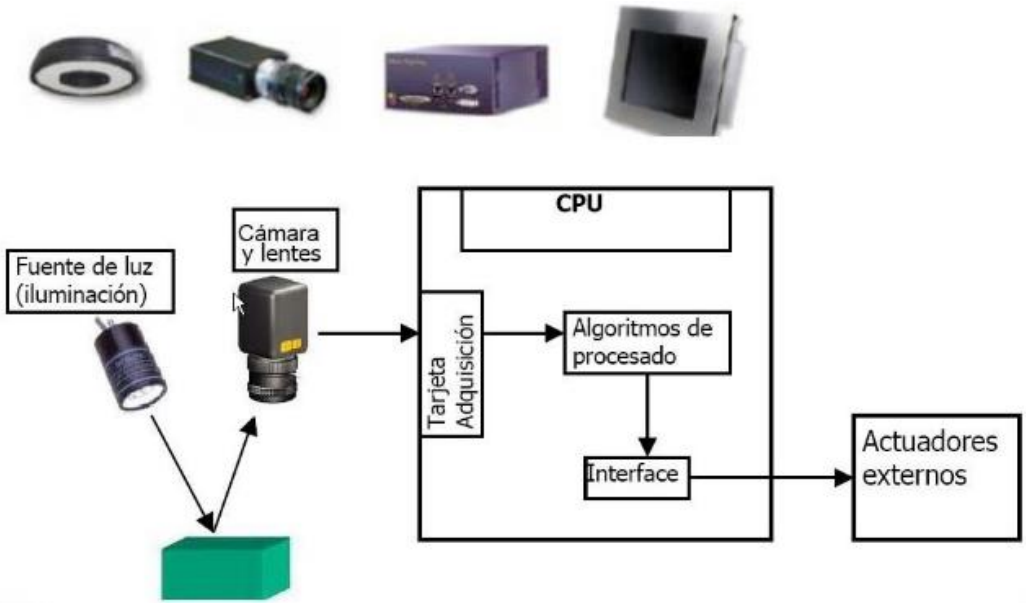
- La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes (por ejemplo, caras humanas).
- La evaluación de los resultados (por ejemplo, segmentación, registro).
- Registro de diferentes imágenes de una misma escena u objeto, es decir, hacer concordar un mismo objeto en diversas imágenes.
- Seguimiento de un objeto en una secuencia de imágenes.
- Mapeo de una escena para generar un modelo tridimensional de la escena; este modelo podría ser usado por un robot para navegar por la escena.
- Estimación de las posturas tridimensionales de humanos.
- Búsqueda de imágenes digitales por su contenido.

2.1.1.1 Características principales en un sistema de visión artificial

- ❖ Analizan luz o color reflejado: Miden nivel de luz
- ❖ Detectan bordes y formas
- ❖ Analizan color
- ❖ Actúan sin contacto: No deforman el material
- ❖ Se puede analizar un objeto en movimiento
- ❖ Son automáticos: Alta velocidad de procesado
- ❖ Flexibles: basados en software
- ❖ Entorno informático

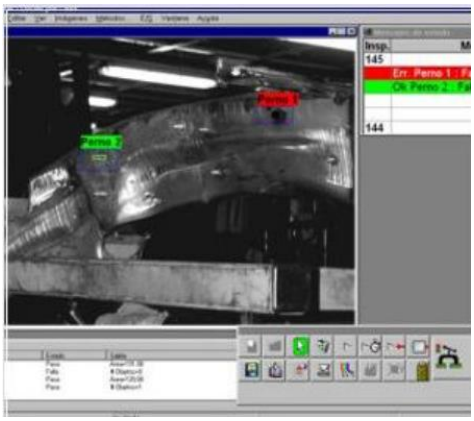
2.1.1.2 Componentes de un sistema de visión artificial

Figura 1. Componentes de un sistema de visión artificial.



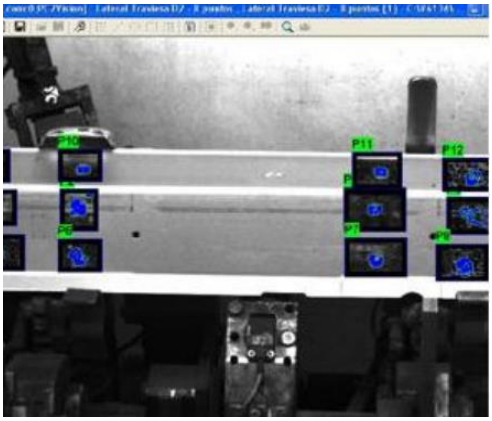
Fuente: Aplicación práctica de la visión artificial en el control de procesos industriales. (2012, Febrero). Recuperado el 21 de Enero de 2014 de la página web: http://www.infoplc.net/files/documentacion/vision_artificial/infoPLC_net_UD_1_DIDAC.pdf Ejemplos de algunas aplicaciones de visión artificial.

Figura 2. Control de pernos



Fuente: Aplicación práctica de la visión artificial en el control de procesos industriales. (2012, Febrero). Recuperado el 21 de Enero de 2014 de la página web: http://www.infoplc.net/files/documentacion/vision_artificial/infoPLC_net_UD_1_DIDAC.pdf

Figura 3. Presencia de orificios

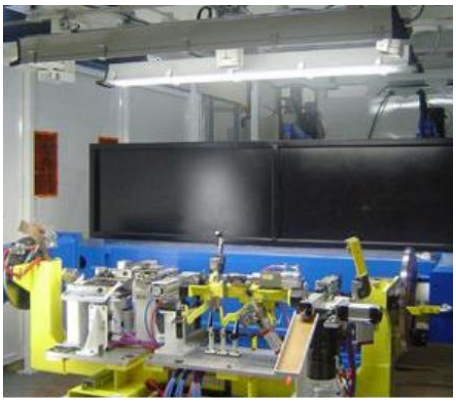


Fuente: Aplicación práctica de la visión artificial en el control de procesos industriales. (2012, Febrero). Recuperado el 21 de Enero de 2014 de la página web: http://www.infoplc.net/files/documentacion/vision_artificial/infoPLC_net_UD_1_DIDAC.pdf

Figura 4. Control de rosca

Fuente: Aplicación práctica de la visión artificial en el control de procesos

Figura 5. Control de utillajes



Fuente: Aplicación práctica de la visión artificial en el control de procesos industriales. (2012, Febrero). Recuperado

industriales. (2012, Febrero). Recuperado el 21 de Enero de 2014 de la página web: http://www.infoplc.net/files/documentacion/vision_artificial/infoPLC_net_UD_1_DIDAC.pdf

2.1.2 Sistema de Pick-Up & Place (Guiado de Robots)

Localización de la posición de un objeto detectando las coordenadas del mismo en el espacio para recogerlo y desplazarlo al lugar deseado.

Guiado de robots y máquinas:

- Localización de centro y orientación para ensamblar piezas
- Manipulado y posicionamiento de piezas
- Recorrido guiado de objetos
- Seguimiento

Figura 6-7. Pick-up & Place 2D.

Aplicación práctica de la visión artificial en el control de procesos industriales. (2012, Febrero). Recuperado el 21 de Enero de 2014 de la página web: http://www.infoplc.net/files/documentacion/vision_artificial/infoPLC_net_UD_1_DIDAC.pdf

2.1.3 Matlab

Según Cuevas Jiménez E. J, Zaldivar Navarro D. (s/f).

> Cuenta una gran cantidad de programas de apoyo especializados, que se denominan Toolboxes**,** que aumentan el potencial en el programa principal. Esos Toolboxes cubren casi todas las áreas principales de la ingeniería y la simulación, unas de las ñtoolboxesò que se destacan son las de adquisición y procesamiento de imágenes y el MatLab Guide que nos permitirá realizar la interface gráfica. (p.32)

El Toolbox de adquisición de imágenes cuenta con una gran cantidad de funciones que permite, adquirir imágenes de distintos dispositivos (desde cámara especializadas compatibles con MatLab hasta las webcams USB), que permiten visualizar videos, adquisición de imágenes, hasta llevar los datos al entorno de trabajo de MatLab, filtrado, etc.

Según Barranco (2011). ñEl Toolbox de procesamiento de Imágenes de MatLab proporciona una gran cantidad de prestaciones que incrementa las capacidades para desarrollar aplicaciones y algoritmos para el campo del análisis y procesado de imágenesò. (p.16)

Ya que MatLab basa su funcionamiento en matrices lo convierte en una excelente herramienta de trabajo gracias a su entorno de trabajo matemático y de creación ya que las imágenes principalmente son matrices con esto la toolbox de MatLab cuanta con las siguientes prestaciones:

- Análisis y estadística de imágenes.
- Diseño de filtros.
- Operaciones morfológicas, geométricas y de color.
- Transformaciones 2D, etc.
- Mejora y retocado de imágenes.

2.1.3.1 Conceptos básicos de Imágenes

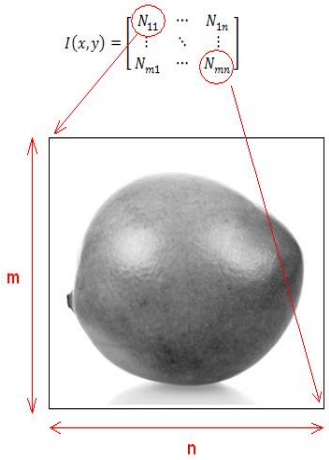
Según Hanselman, y Littlefield, B. (2005). Manual Matlab 7.0

> En MatLab una imagen a escala de grises es representada por medio de una matriz bidimensional de m x n elementos en donde n representa el número de píxeles de

> ancho y m el número de píxeles de largo. El elemento N11 corresponde al elemento de la esquina superior izquierda (ver figura 18), donde cada elemento de la matriz de la imagen tiene un valor de 0 (negro) a 255 (blanco). (p.21)

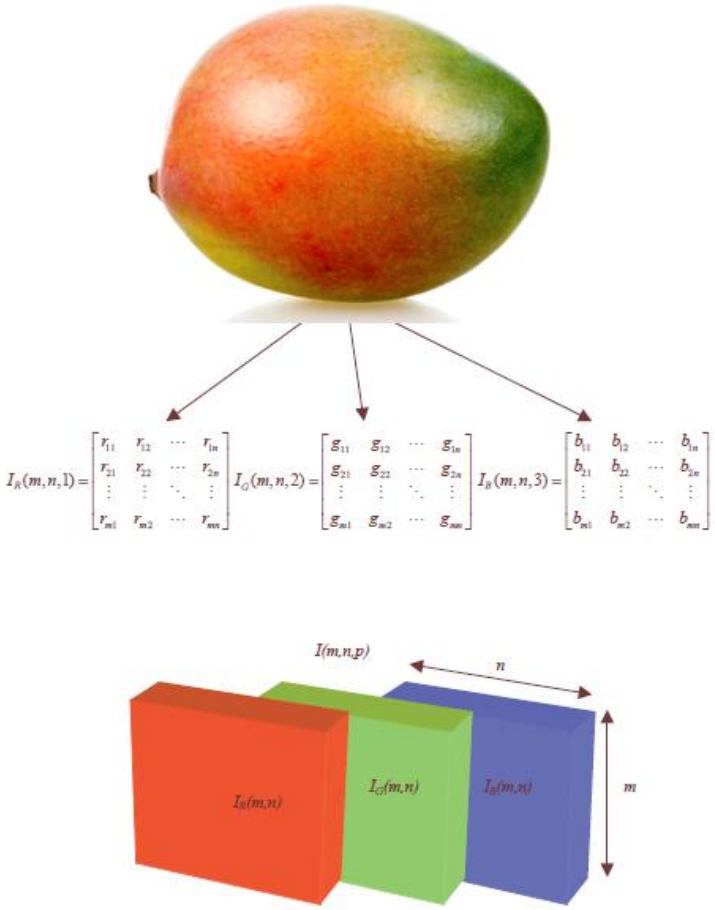
Por otro lado una imagen de color RGB (la más usada para la visión computacional, además de ser para MatLab la opción default) es representada por una matriz tridimensional m x n x p, donde m y n tienen la misma significación que para el caso de las imágenes de escala de grises mientras p representa el plano, que para RGB que puede ser 1 para el rojo, 2 para el verde y 3 para el azul. La figura 19 muestra detalles de estos conceptos.

Figura 8 Imagen en escala de grises



Fuente: Espin, R (2011) ñCORTINA DE AGUA PROGRAMABLEò Universitat Politècnica de Catalunya (UPC) (PP.21-25) (modificado por el autor)

Figura 9. Imagen a color representada en RGB.



Fuente: Espin, R (2011) ñCORTINA DE AGUA PROGRAMABLEò Universitat Politècnica de Catalunya (UPC) (PP.21-25) (modificado por el autor)

Según Espin, R (2011)

Los formatos de imágenes soportados por MatLab son:

- JPEG: La extensión de este formato es .jpg
- TIFF: La extensión de este formato es .tiff
- GIF: La extensión de este formato es .gif
- BMP: La extensión de este formato es .bmp
- XWD: La extensión de este formato es .xwd
- PNG: La extensión de este formato es .png (p.23)

2.1.3.2 Procesamiento de Imágenes

Es el conjunto de técnicas que se aplican sobre imágenes digitales con el fin de mejorar la calidad o facilitar la búsqueda de información dentro de las mismas. Algunas de las técnicas que podremos encontrar son el Filtrado, la Transformación, la Ecualización, y Segmentación digital.

En este trabajo aplicamos las siguientes técnicas:

- Conversión a escala de grises mediante la función rgb2gray(): Roncagliolo, p. (2012) "Las imágenes en escala de grises al contrario de las RGB son un arreglo unidimensional de 8 bits de profundidad. Esta escala tiene un total de 256 intensidades de grises donde 1 corresponde al Negro y 255 corresponde al Blancoò. (p.1)

- Detección de bordes mediante los algoritmos sobel y canny: Según Lego. (2013 ñEn primera instancia se aplicó el algoritmo sobel mediante la siguiente función edge(im_gray,'sobel'), esta función se encarga de invocar distintos algoritmos de detección de bordes como el canny, sobel y prewittò.

Según Departamento de Ingeniería electrónica, Telecomunicación y Automática. (2005)

> Algoritmo Sobel es utilizado en procesamiento de imágenes, especialmente en algoritmos de detección de bordes. Técnicamente es un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen. Para cada punto de la imagen a procesar, el resultado del operador Sobel es tanto el vector gradiente correspondiente como la norma de éste vector. (p.2)

En MatLab este algoritmo se debe aplicar solo en una imagen en escala de grises.

- Dilatación de imagen: Mediante la función imdilate(im_edge,SE) la cual nos permite complementar el contorno de la imagen debido a que por el movimiento o características de la imagen el objeto no presenta borde definidos o separados entre sí.

- Filtrado de la imagen: Mediante la función imfill(im_dilate,'holes'), se aplica un filtro el cual nos permite rellenar agujeros o ruido en la imagen, pero si esto no es suficiente aplicamos la siguiente operación find([propied.Area]<1000) la cual nos permite buscar áreas menores a mil pixeles que luego se eliminaran mediante un ciclo for combinado con la función round.

- Corte de imagen: En el cual mediante la obtención de las características de la imagen y luego de aplicar un cuadro delimitador del objeto, la realizamos mediante la función imcrop(im,[CAJA]).

- Separamos la imagen recortada en los tres planos RGB: Mediante la siguiente operación

```
R = RECO(:,:,1);
G = RECO(:,:,2);
B = RECO(:,:,3);
```

Continua a se aplica la función sum para cada uno de los planos de la imagen para de esta forma conocer cuál es el color al que corresponde el objeto a la imagen, dependiendo de cuál de los planos tenga el mayor valor.

Según Hanselman, D. C., & Littlefield, B. (2005).

> Para la comunicación USB con la cual cuenta la tarjeta de adquisición de datos, la empresa Microchip cuanta en su página web con los drivers y los archivos que son necesarios para permitir establecer una comunicación mediante el puerto USB con la familia de microcontroladores 18FXXXX, estos microcontroladores permiten efectuar transferencias de paquetes de hasta 64 bytes cada milisegundo por cada túnel abierto, para la programación del microcontrolador se emplea el software PIC C Compiler, los descriptores o drivers utilizados para la comunicación USB son los que se encuentran en el compilador PIC C Compiler. (p.12)

CAPITULOIII

3. ASPECTOS METODOLOGICOS

3.1 Tipo de investigación: Descriptiva

Técnica:

- Análisis y observación de variables
- Procesamiento y análisis de datos

3.2 Fase 1: Investigación Teórica:

La investigación teórica es aquella que se realiza para determinar cómo funciona el proceso y cómo podríamos llevarlo a cabo mediante el empleo de un sistema de visión artificial o visión asistida por computadora, es toda aquella documentación acerca del proceso de clasificación de objetos mediante la visión artificial, se evidencian en el marco teórico de este anteproyecto, lo mismo que la investigación sobre el procesamiento de la imagen, sensores, cámara, clasificación de objetos.

3.3 Fase 2: Realización de pruebas:

Se realizaran pruebas del sistema de visión artificial, inicialmente se debe verificar el funcionamiento de cada una de las piezas del sistema lo mismo que las variables a simular de la siguiente manera:

Pruebas de la cámara web:

- Prueba de encendido, conectividad al PC de programación, cableado y comunicación con el software de control para el procesamiento digital de la imagen.

- Pruebas de interface de entradas y salidas con el software de programación, simulaciones de funcionamiento.

- Prueba y revisión de las imágenes capturadas.

Pruebas del sistema de detección del objeto:

- Led a chorro blanco apuntado a la fotorresistencia, fotorresistencia no percibe luz, detecto que el objeto ya paso y se encuentra en una posición favorable para tomar la imagen.

Pruebas de comunicación USB:

- Se revisa la correcta comunicación USB entre el PC y el microcontrolador.

Pruebas microcontrolador:

- Verificación de funcionamiento de entradas digitales y salidas digitales.

- Verificación de la entrada analógica correspondiente al sensor óptico de barrera.

- Programación básica de arranque, compilación y ejecución de órdenes básicas de trabajo.

Se ubicara cada uno de los objetos en la banda transportadora para su posterior clasificación de acuerdo a la variable seleccionada por el operador ya sea forma o color para verificar la correcta ubicación del objeto en el contenedor que corresponda al mismo.

3.4 Fase 3: Implementación de la Simulación:

Se realiza el proceso de clasificación de objetos de producto terminado de acuerdo a su forma y color, con un sistema de visión artificial mediante la detección del objeto tras el paso por el sensor óptico de barrera, captura del mismo mediante la cámara, y el posterior procesamiento de la imagen en el software de ingeniería MATLAB, y comunicación vía USB con el microcontrolador para la correcta ubicación de los objetos mediante el motor (en este caso motor paso a paso) que girara para ubicarlo en el contenedor adecuada para dicho objeto, por obvias razones el sistema solo tiene operación automática.

Se ubicaran distintos objetos para de esta forma modificar las variables del sistema que son forma y color del mismo.

Se requieren los siguientes elementos:

- Computador: En el cual se encontrara la interface gráfica para la operación del sistema, por parte del operador.

- Software de ingeniería MATLAB: El cual se encargara del procesamiento digital de las imágenes y en el cual se desarrollara la interface gráfica.

- Software de programación del microcontrolador PIC C Compiler: En el cual se desarrollara la programación del microcontrolador PIC18F4550 que se empleara para el desarrollo de la tarjeta de adquisición de datos o interface de entradas y salidas del sistema.

- Programador de microcontroladores Pickit: El cual como su nombre lo indica nos permitirá realizar la programación del microcontrolador PIC18F4550 que se empleara en la tarjeta de adquisición de datos o interface de entradas y salidas del sistema.

- Cable de comunicación USB: el cual permitirá realizar la comunicación entre la tarjeta de adquisición de datos con el computador.

- Sensor de óptico barrera compuesto por un diodo led a chorro, y una fotorresistencia: que se empleara para la detección de los objetos para enviar la orden de tomar la fotografía en el momento indicado.

- Cámara Logitech C170: Encargada de realizar la toma de imágenes en el momento en que se le dé la orden.

- Driver MCHPFSUSB v2.2: Proporcionado por la empresa Microchip el cual permite el reconocimiento por parte del PC de la tarjeta de adquisición de dato o interface de entradas y salidas.

CAPITULO IV

4. DESARROLLO METODOLOGICO

4.1 Resultados Esperados

El fin de este proyecto es desarrollar un sistema de clasificación de objetos empleando la visión artificial, mediante el diseño e implementación de un algoritmo de control en el software de ingeniería MatLab, que permita al computador entender e interpretar una imagen tomada por una cámara mediante la toolbox de MatLab Image Processing, la cámara también será controlada por dicho software mediante la toolbox Image Acquisition, con lo cual también se diseñara e implementara una tarjeta de adquisición de datos como interface de entradas y salidas con comunicación vía USB mediante el microcontrolador PIC 18F4550, la cual se encarga de controlar los actuadores del sistema que son el motorreductor de la banda transportadora y el motor para el control del movimiento de los contenedores que corresponde a un motor paso a paso unipolar, la tarjeta de adquisición de datos tendrá como entrada un sensor óptico de barrera, el cual determinara el momento indicado para realizar la fotografía del objeto.

Diseñar e implementar una interface gráfica en MatLab que permita al usuario u operador observar en video en tiempo real lo objetos que son detectados por la cámara, como también la indicación en la parte inferior de qué tipo de objetos es y el color del mismo, también que el operador pueda realizar el control del sistema, mediante botones con los cuales el usuario pueda activar o desactivar la cámara, encender el motorreductor de la banda transportadora o apagarlo en caso de emergencia, dicha interface contara con un panel de selección de la variable por la cual se realizara la clasificación del objeto ya sea por forma o color.

Figura 10. Activamos la cámara y el motor de la banda transportadora en la interface

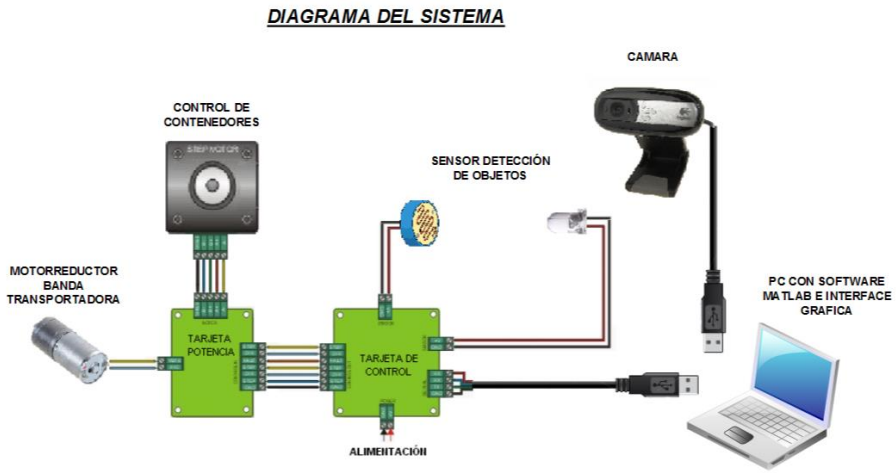
Fuente: El autor

FIGURA 11. Interface Gráfica en MatLab

Fuente: El autor

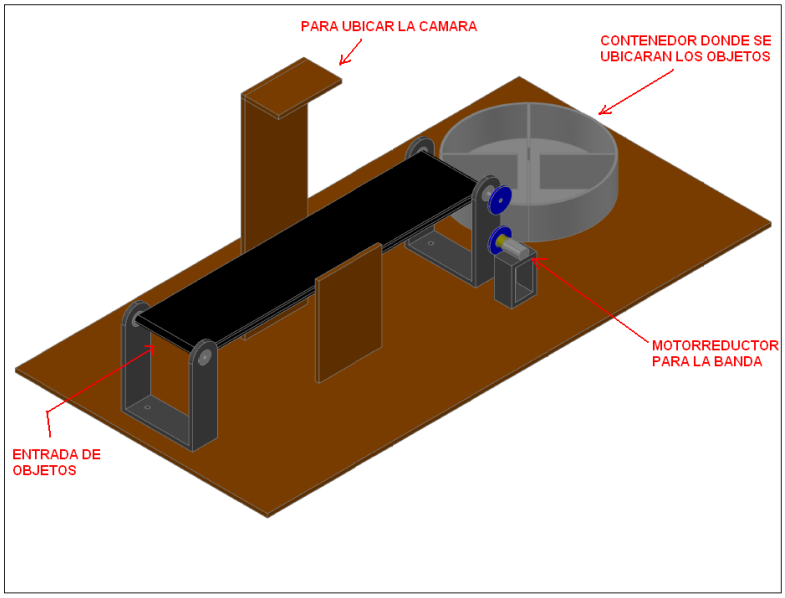
Como resultado se necesita que el sistema clasifique los objetos según su color y forma, para el color se empleara los colores rojo, verde y azul que componen el modelo de colores básicos RGB, para la forma se empleara las figuras triangulo, circulo, cuadrado y hexágono, según estos parámetros los objetos serán ubicados en distintos contenedores.

Figura 12. Diagrama del sistema



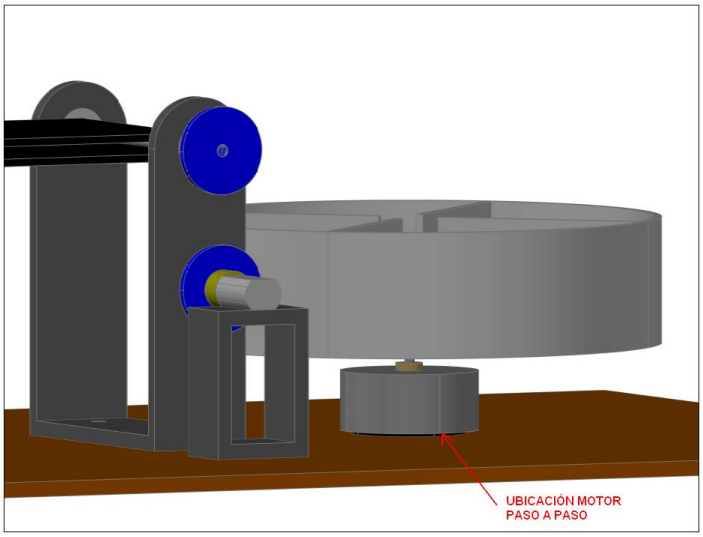
Fuente: El autor

Figura 13. Maqueta Propuesta elaborada en AutoCad 2008



Fuente: El autor

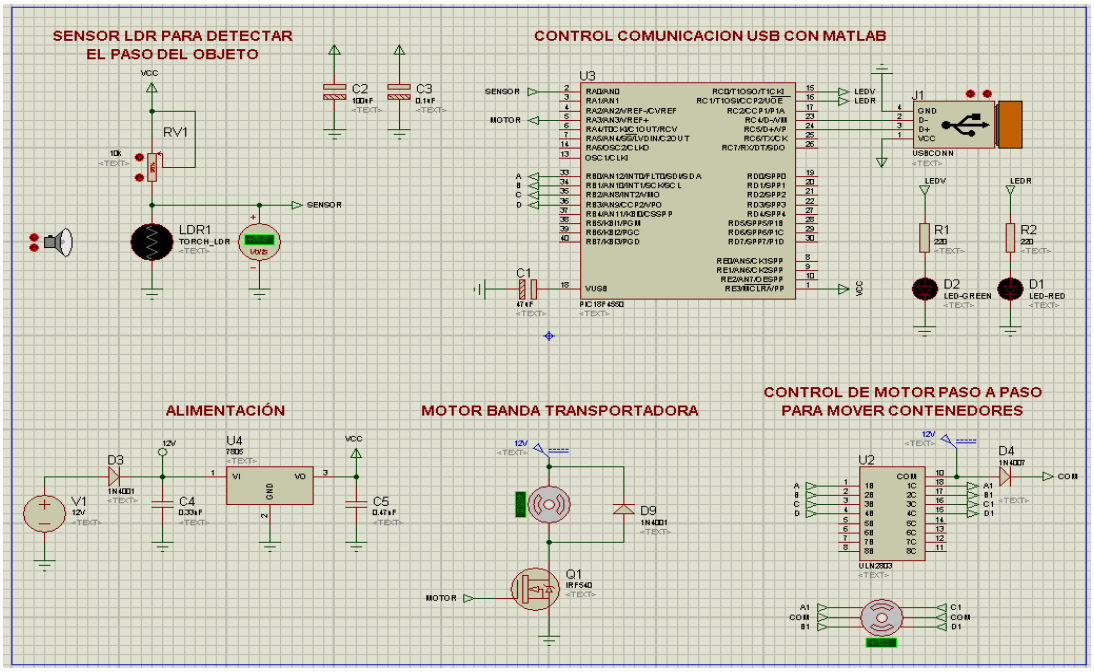
Figura 14. Detalle ubicación del motor paso a paso



Fuente: El autor

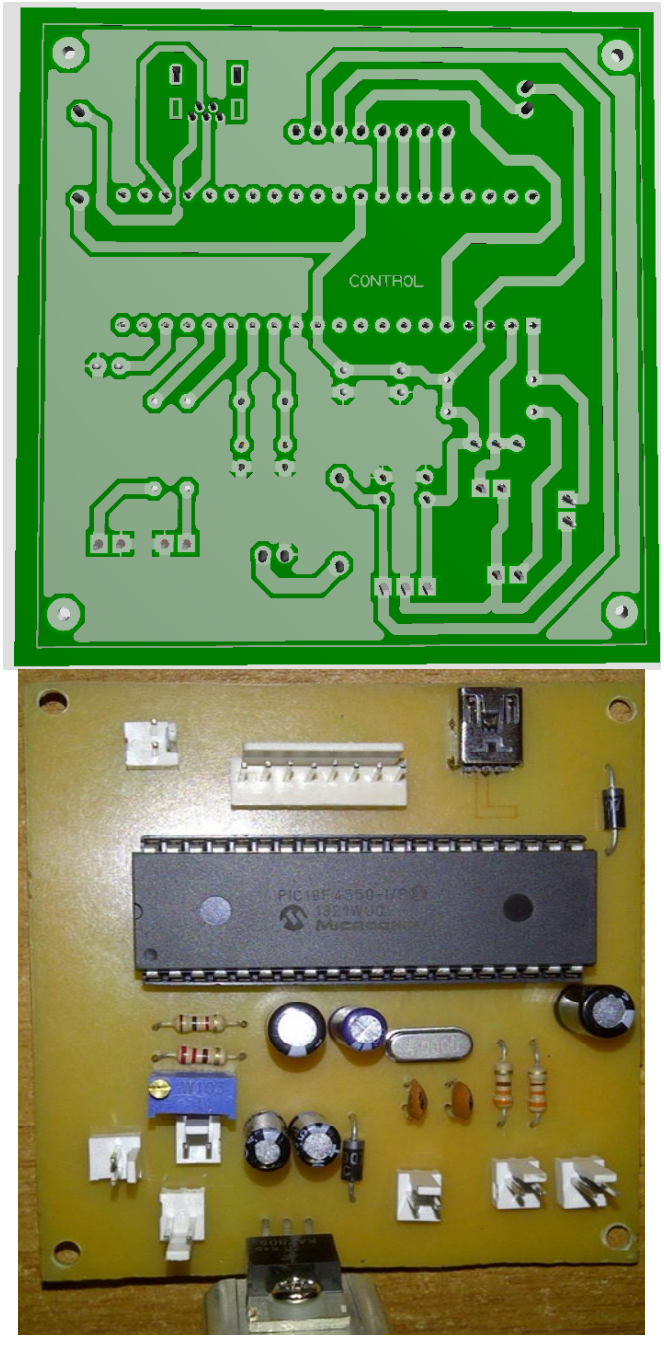
4.2 **Diseño e Implementación:** Plano

Figura 15. Plano Electrónico



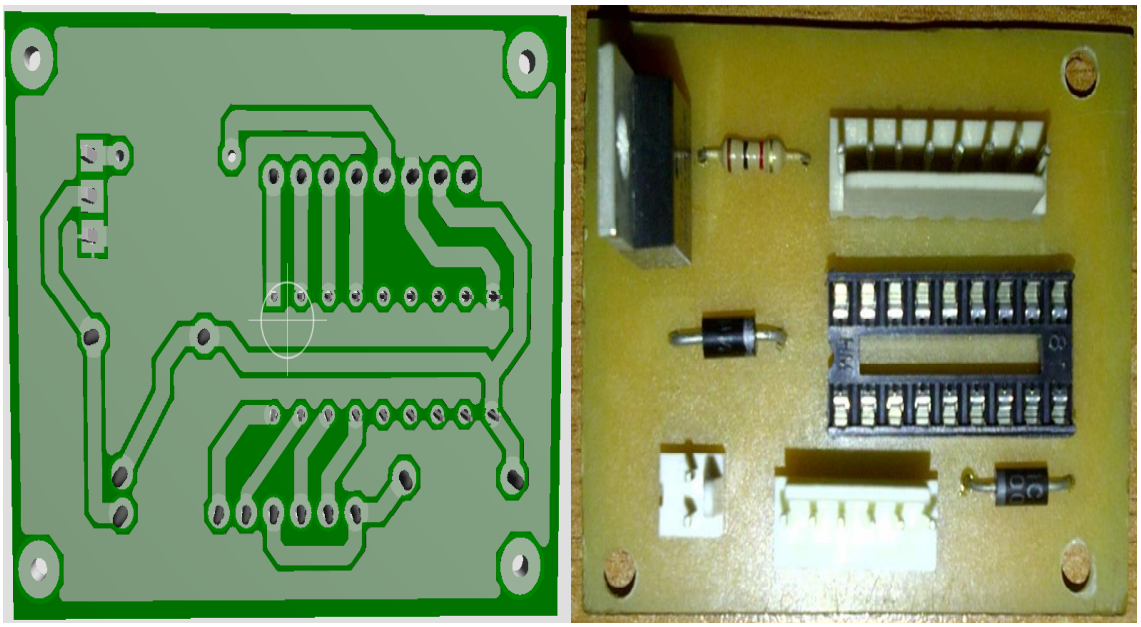
Fuente: El autor

Figura 16. Tarjeta de Control



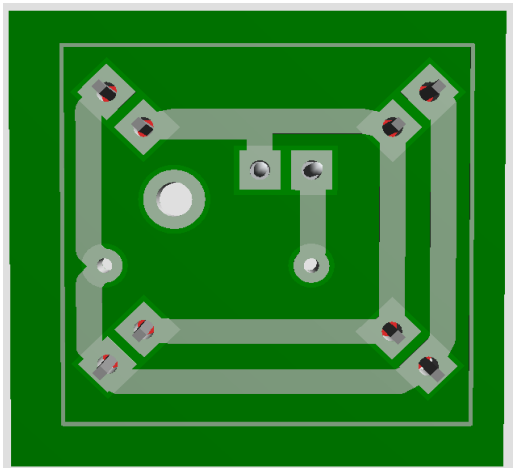
Fuente: El autor

Figura 17. Tarjeta de potencia



Fuente: El autor

Figura 18. Tarjeta de iluminación

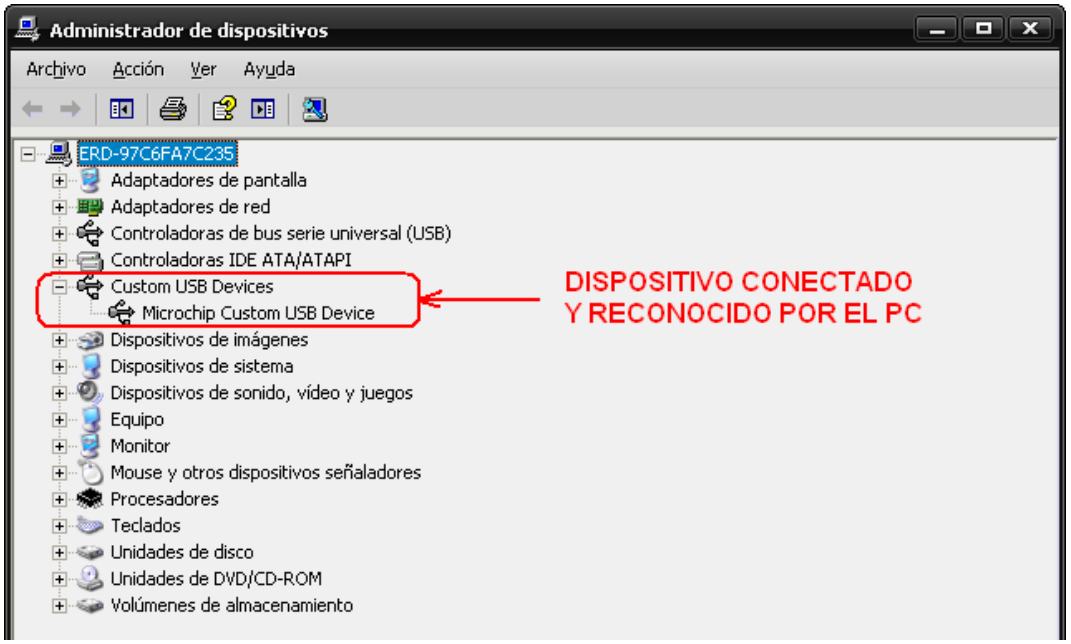


Fuente: El autor

Figura 19 Administrador de dispositivos sin conectar la tarjeta

Fuente: El autor

Figura 20. Administrador de dispositivos después de conectar la tarjeta al PC



Fuente: El autor

Al conectarse por primera vez pide la instalación del controlador de nuestro dispositivo.

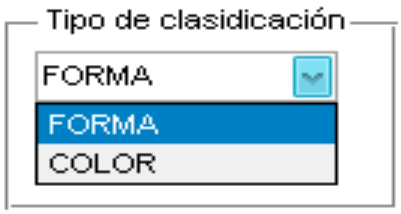
Figura 21 Interface Gráfica en MatLab

Fuente: El autor

Figura 22 Activamos la cámara y el motor de la banda transportadora en la interface

Fuente: El autor

Figura 23 Tipos de clasificación



Fuente: El autor

Figura 24. Prueba con el círculo azul y clasificación por color, se bloque la luz hacia el LDR para que tomara la foto en ese instante.

Fuente: El autor

Figura 25 Prueba para el círculo rojo

Fuente: El autor

Figura 26 Prueba para el Hexágono verde

Fuente: El autor

Cambiamos a clasificación por forma y probamos con el cuadrado azul

Figura 27 Prueba para el Cuadrado azul

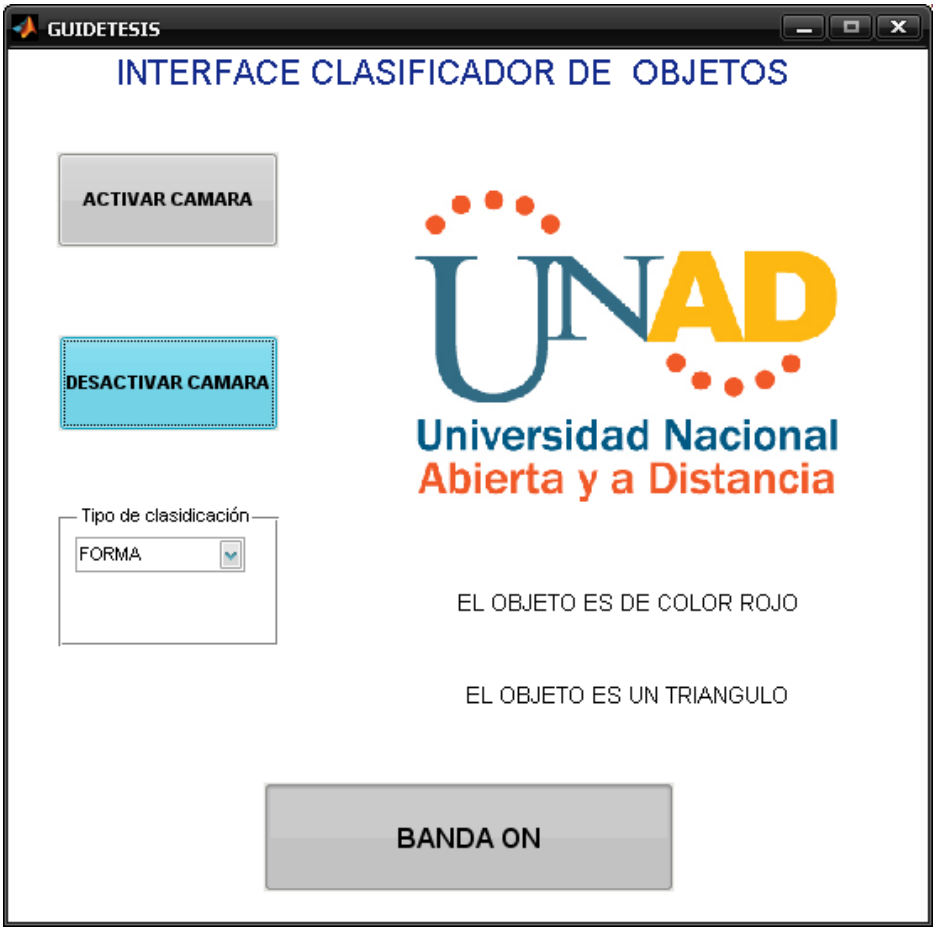
Fuente: El autor

Figura 28 Prueba con el triángulo rojo

Fuente: El autor

Para terminar desactivamos la cámara y apagamos el motor de lavanda transportadora.

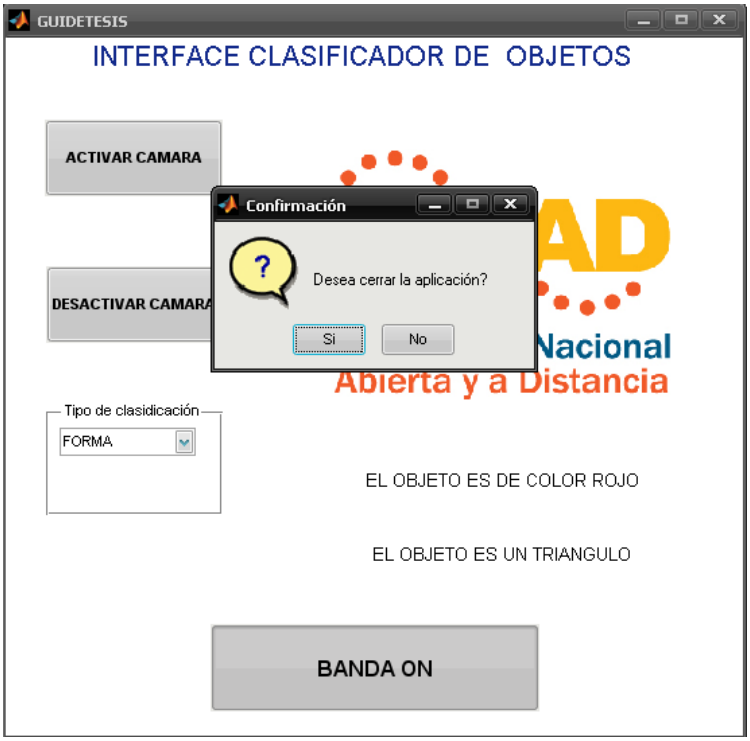
Figura 29 Preparando el cierre de la aplicación



Fuente: El autor

Cerramos la aplicación pero antes nos muestra la siguiente ventana emergente de confirmación.

Figura 30 Confirmación del cierre



Fuente: El autor

Cámara Empleada

Figura 31 Logitech C170

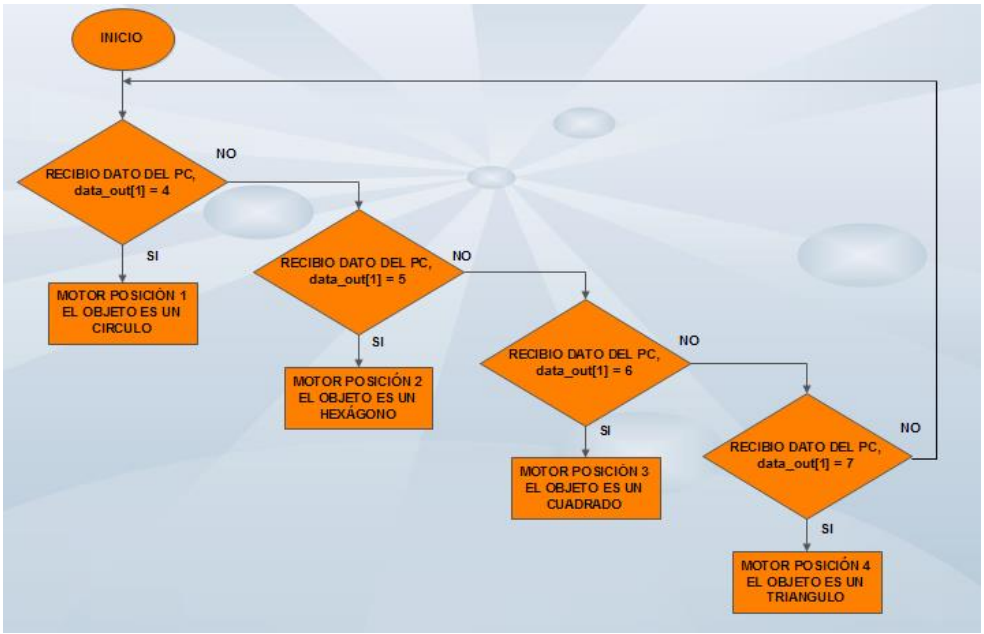
Fuente: El autor

Figura 32 Diagrama de flujo detección de objeto

Figura 33 Diagrama de flujo captura de imagen

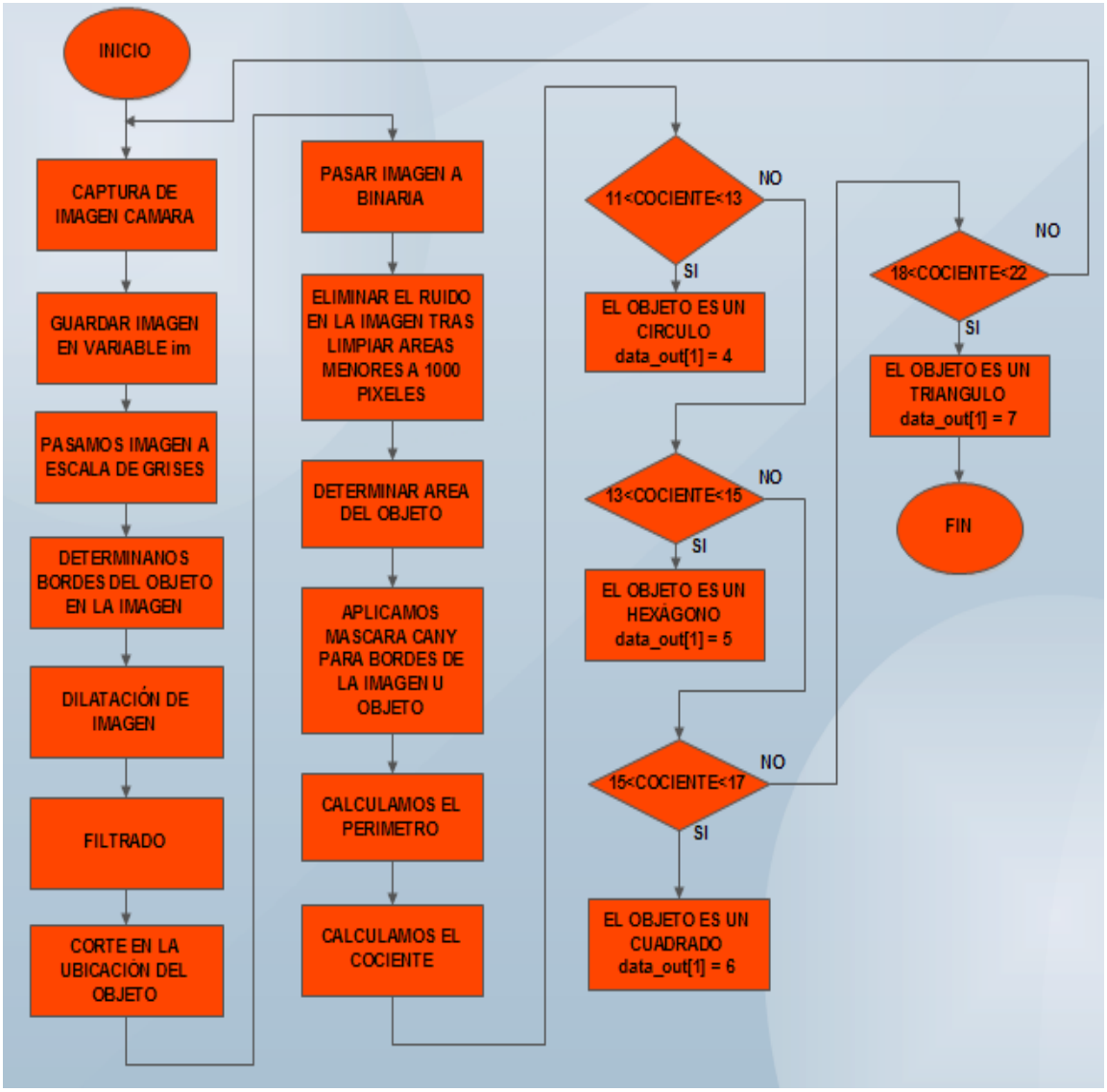
Fuente: El autor

Figura 34. Diagrama de flujo control de la ubicación del contenedor



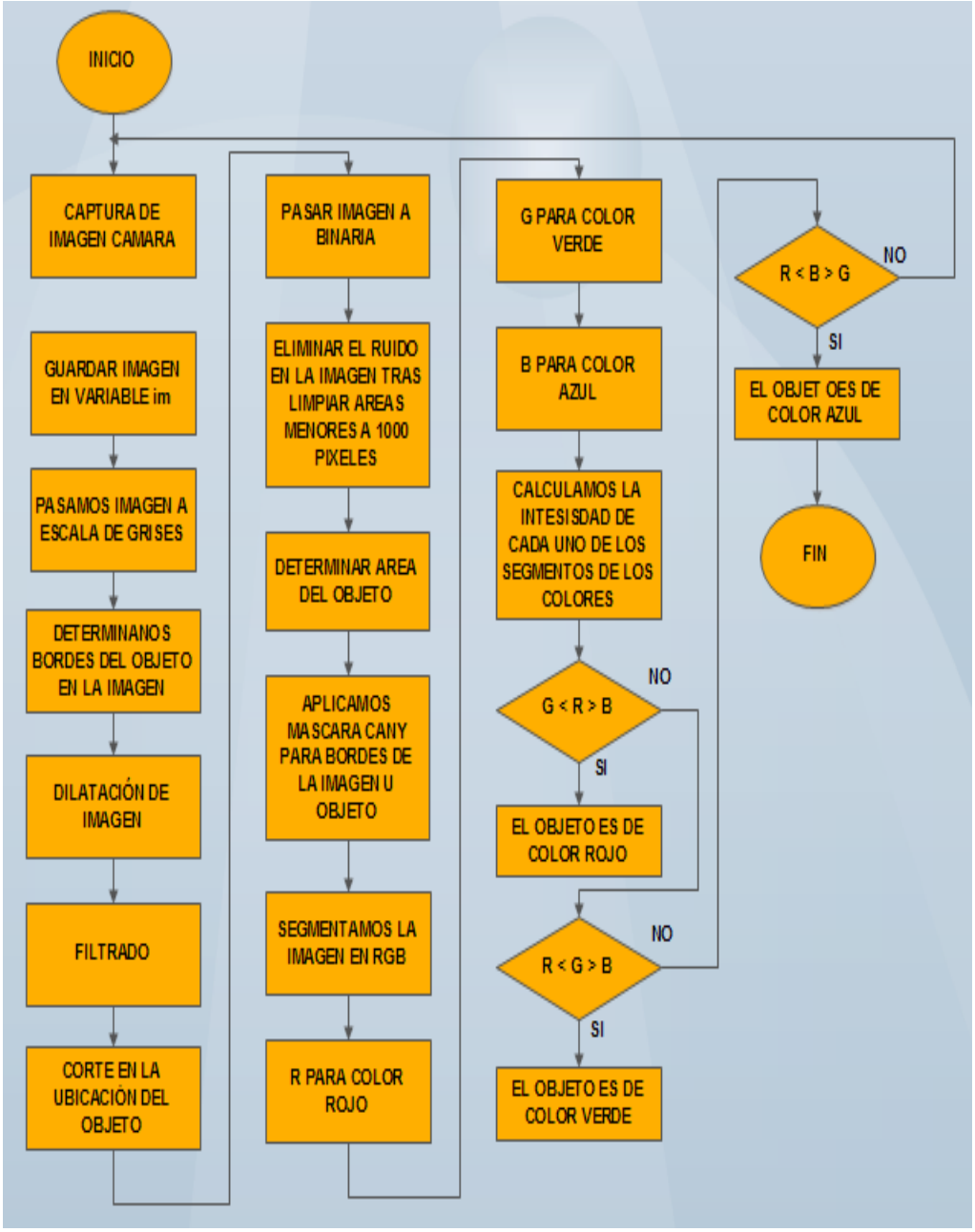
Fuente: El autor

Figura 35. Diagrama de flujo para determinar forma del objeto, si el operador determino que la clasificación es para forma.



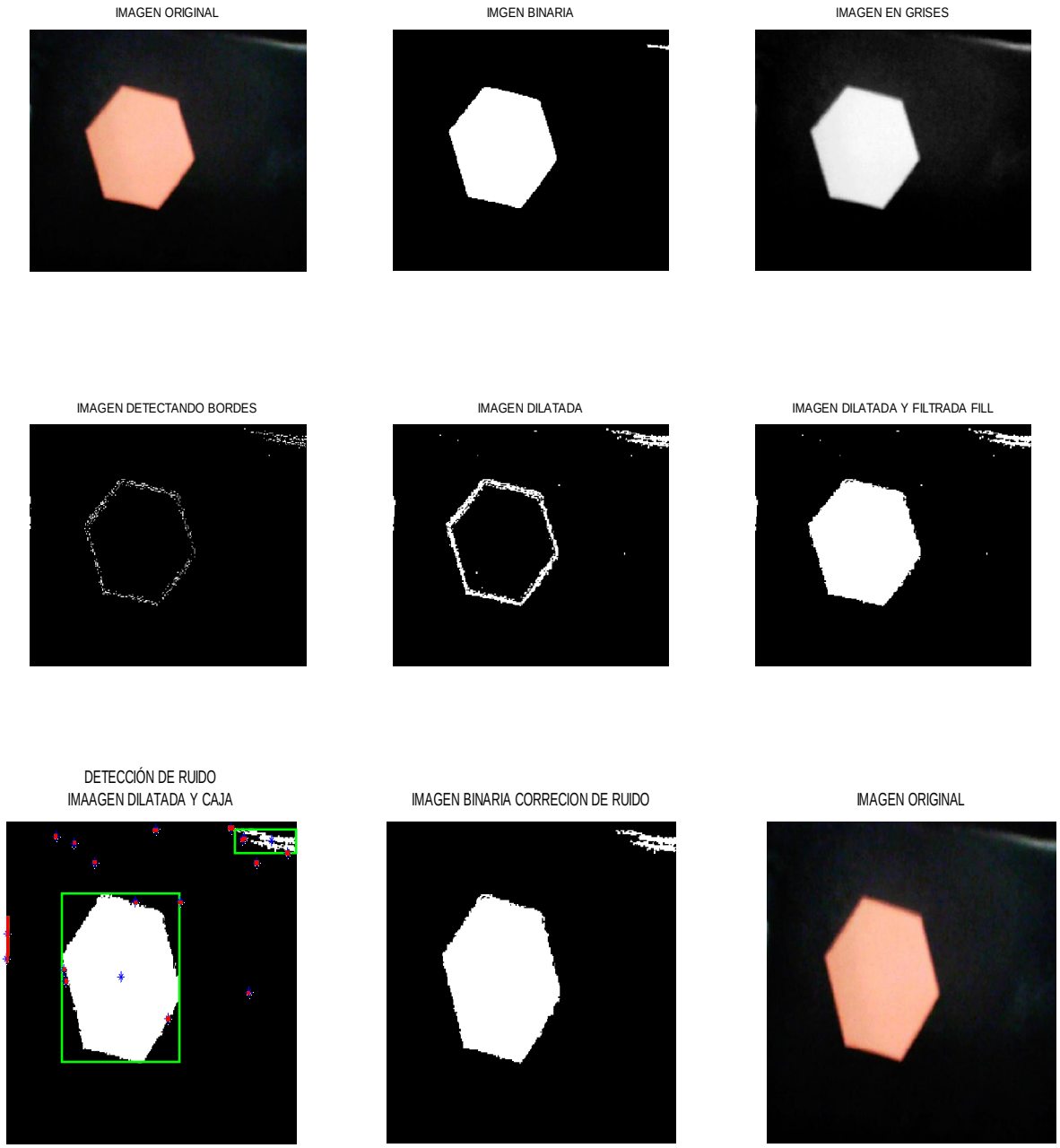
Fuente: El autor

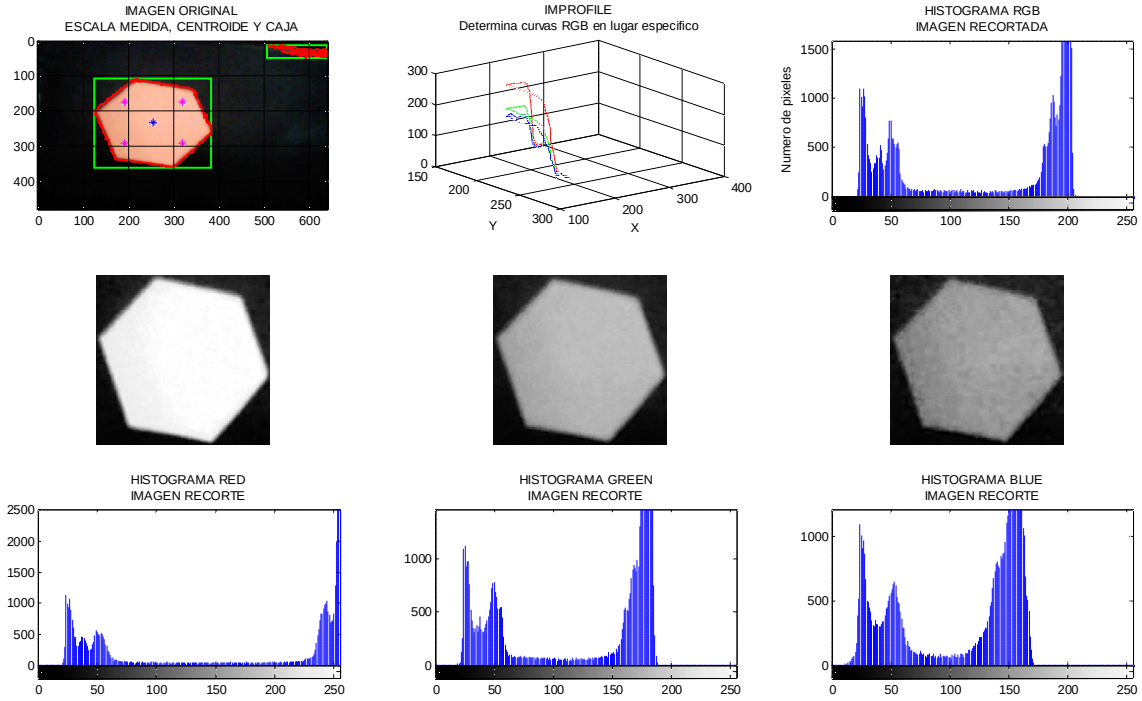
Figura 36. Diagrama de flujo para determinar el color del objeto si el operador determino que la clasificación es por color.



Fuente: El autor

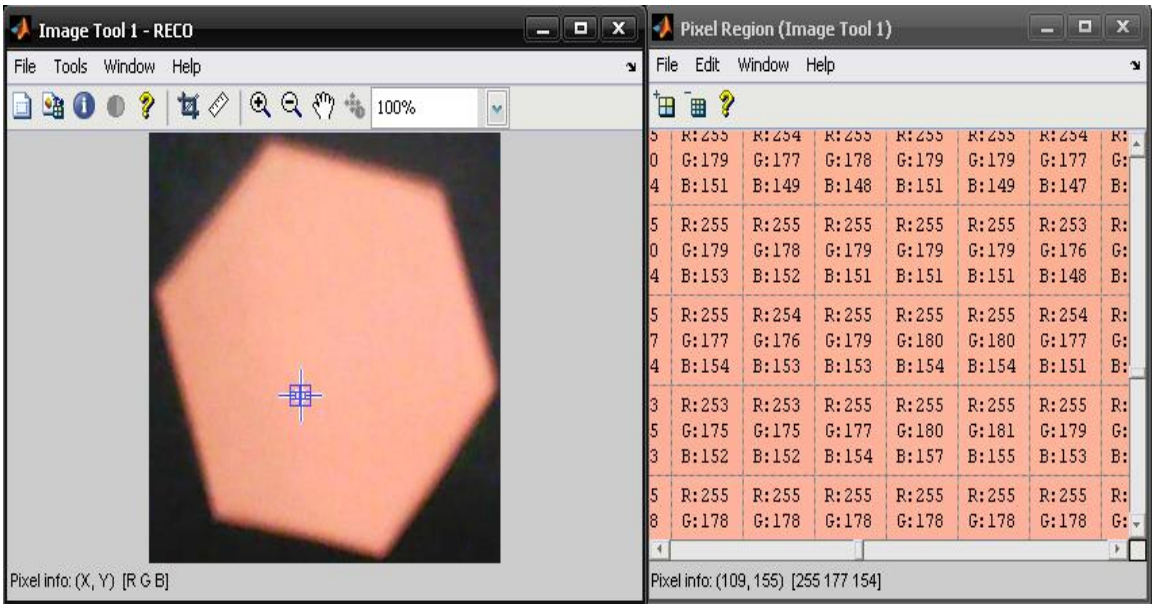
Figura 37. Procesamiento de la imagen que es llevado a cabo





Fuente: El autor

Figura 38. Mapa De Pixeles RGB del objeto



Fuente: El autor

Desarrollo Matemático para Clasificación por Formas

Área del cuadrado $A = L^2$

Perímetro $P = 4L$

Cociente (C) $C = \frac{P^2}{A} = 16$

$$C = \frac{(4L)^2}{L^2} = \frac{16\cancel{L^2}}{\cancel{L^2}} = 16$$

Área del circulo $A = \pi r^2$

Perímetro $P = 2\pi r$

Cociente (C) $C = \frac{P^2}{A} = 12.566$

$$C = \frac{(2\pi r)^2}{\pi r^2} = \frac{4\pi^2\cancel{r^2}}{\cancel{\pi r^2}} = 4\pi = 12.566$$

Área del triangulo $A = \frac{\sqrt{3}}{4} l^2$

Perímetro $P = 3l$

Cociente (C) $C = \frac{P^2}{A} = 20.784$

Desarrollo matemático para el hexágono

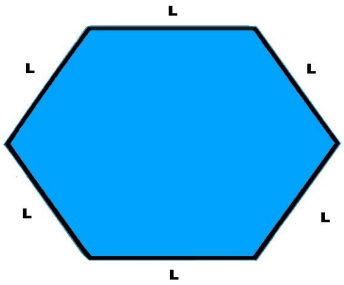
Área del hexágono $A = \frac{3\sqrt{3}}{2}l^2$

Perímetro $P = 6l$

Cociente (C) $C = \frac{P^2}{A} = 13.856$

$$C = \frac{(6l)^2}{\frac{3\sqrt{3}}{2}l^2} = \frac{\cancel{36l^2}}{\frac{\cancel{3}\sqrt{3}}{2}\cancel{l^2}} = \frac{12}{\frac{\sqrt{3}}{2}} = 13.856$$

Figura 39. Hexágono



Los intervalos para que el computador tenga conocimiento que forma está procesando son los siguientes:

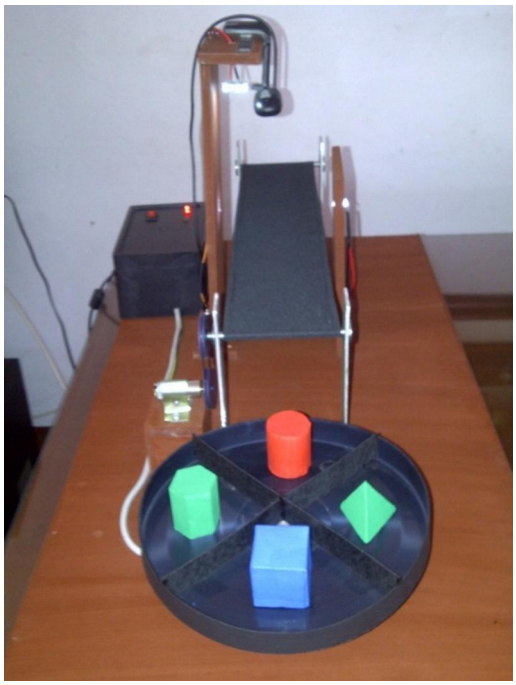
Para el círculo: 11 < cociente C < 13

Para el hexágono: 13 < cociente C < 15

Para el cuadrado: 15 < cociente C < 17

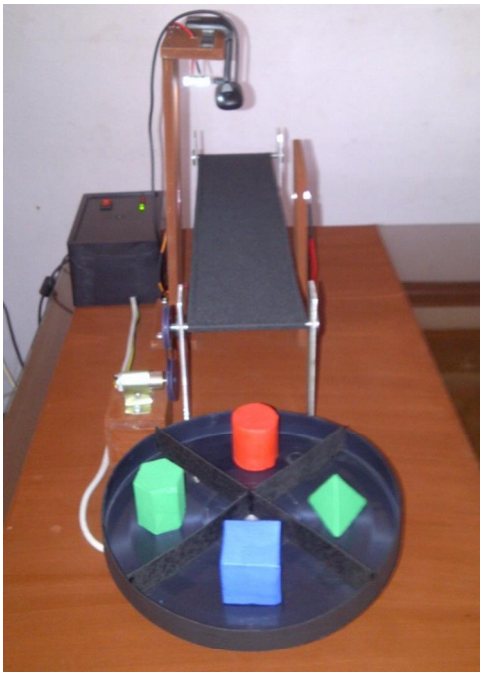
Para el triángulo: 17 < cociente C < 22

Figura 40. Prototipo sin ser conectado al computador, led rojo encendido



Fuente: El autor

Figura 41. Prototipo tras se conectado al computador, led verde encendido.



Fuente: El autor

Programación Pic C Compiler Para El Microcontrolador

```
////////////////////////////////////////////////////////////////////////
//// Codigo Fuente Proyecto de Grado para el titulo de Ingenieira    ////
//// Electrónica de la UNAD.                                         ////
////                                                                 ////
//// Proyecto Clasificación de Objetos geometrico por visión         ////
//// artificial, la comunicación entre la tarjeta de adquicición de  ////
//// datos se realiza mediante el puerto USB del PIC 18F2550 el cual ////
//// recibira y enviara datos al PC, enviandole la señal al PC       ////
//// para que capture la imagen del objeto tras el paso por el       ////
//// sensor por medio de la webcam, para que esta sea procesada por  ////
//// software de ingenieria MatLab, este se encargara de procesar    ////
//// la imagen imagen y determinar que es ya sea un cuadrado,        ////
//// circulo o triangulo para luego enviar al PIC en que ubicación   ////
//// del contenedor debe ubicarlo, los contenedores son modificado   ////
//// mediante un motor paso a paso, el cual recibira las ordenes del ////
//// PIC para ubicar el objeto en la posición a la cual corresponda  ////
////                                                                 ////
//// Los drivers que se requiere para éste dispositivo son los que   ////
//// proporciona Microchip en su pagina web                          ////
//// http://ww1.microchip.com/downloads/en/DeviceDoc/....            ////
//// Microchip%20MCHPFSUSB%20v2.2%20Installer.zip                    ////
////                                                                 ////
//// Cuando el dispositivo sea conectado al PC, saldrá el asistente  ////
//// para la instalación del driver. Se instala el driver            ////
//// que nos proporciona Microchip.                                  ////
////                                                                 ////
//// Al aplicar energia al PIC se enciende el LED en pin RB6 y       ////
//// hasta que el dispositivo halla sido configurado por la PC       ////
//// via puerto USB se encende el Led en RB7.                        ////
////                                                                 ////
////////////////////////////////////////////////////////////////////////

// Por favor, complete los 4 pasos siguientes ...
// Paso 1:
// Cambie la siguiente instrucción de acuerdo al PIC que utilice PIC18F2455/2550/4455/4550
#include <18F4550.h>
#fuses XTPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL1,CPUDIV1,VREGEN,MCLR,NOPBADEN
// Paso 2: Ajuste el fusible del PLL de acuerdo al Xtal que utilice
// No olvide que PLL1 = Para un Xtal de 4Mhz
//               PLL2 = Para un Xtal de 8Mhz
//               PLL3 = Para un Xtal de 12Mhz
//               PLL4 = Para un Xtal de 20Mhz , etc.
#device ADC=10                //La resolución para el conversor analogo es de 10 bit osea que va de 0 a 1024
#use delay(clock=48000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

////////////////////////////////////////////////////////////////////////////
//
// CCS Library dynamic defines.  For dynamic configuration of the CCS Library
// for your application several defines need to be made.  See the comments
// at usb.h for more information
//
////////////////////////////////////////////////////////////////////////////

#define USB_HID_DEVICE     FALSE              //deshabilitamos el uso de las directivas HID
#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK    //turn on EP1(EndPoint1) for IN bulk/interrupt transfers
#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK    //turn on EP1(EndPoint1) for OUT bulk/interrupt transfers
#define USB_EP1_TX_SIZE    64                 //size to allocate for the tx endpoint 1 buffer
#define USB_EP1_RX_SIZE    64                 //size to allocate for the rx endpoint 1 buffer


#include <pic18_usb.h>             //Microchip PIC18Fxx5x Hardware layer for CCS's PIC USB driver
#include <usb_desc_scope_mod.h>    //Descriptors del Pic USB
#include <usb.c>                   //handles usb setup tokens and get descriptor reports

////////////////////////////////////////////////////////////////////////////
/*
Paso 3: Ábra el archivo C:\Archivos de programa\PICC\Drivers\usb_desc_scope.h
        ( donde se instaló el compilador de CCS ) que es el descriptor del USB
        ubicado en su PC, avance hasta la sección start device descriptors
        (aprox en la linea 132) y reemplace los valores del vendor id,
        el product id y el device release number como sigue ( puede copiar
        las tres líneas siguiente y pegar en el archivo del descriptor <<usb_desc_scope.h>> ) :

         0xD8,0x04,           //vendor id (0x04D8 is Microchip)
         0x0B,0x00,           //product id
         0x01,0x00,           //device release number

ESTO ES IMPORTANTE HACERLO CORRECTAMENTE DE LO CONTRARIO, EL DISPOSITIVO NO SERA RECONOCIDO POR EL DRIVER.


*/
////////////////////////////////////////////////////////////////////////////

#define  MOTOR_BANDA PIN_A3
#define  LEDV    PIN_C0
#define  LEDR    PIN_C1
#define  LED_ON  output_high
#define  LED_OFF output_low

// Direcciones de memoria válidas para PIC18F2455/2550/4455/4550 (no olvide que son la misma familia)
// Esto es con el fin de poder escribir directamente en ellos sin usar instrucciones como intermediarios.
#BYTE TRISA  = 0x0F92                 // Registro de control de E/S del puerto A
#BYTE TRISB  = 0x0F93                 // Registro de control de E/S del puerto B
#BYTE TRISC  = 0x0F94                 // Registro de control de E/S del puerto C
#BYTE TRISD  = 0x0F95                 // Registro de control de E/S del puerto C
#BYTE PORTA  = 0x0F80                 // Registro del puerto A
#BYTE PORTB  = 0x0F81                 // Registro del puerto B
#BYTE PORTC  = 0x0F82                 // Registro del puerto C
#BYTE PORTD  = 0x0F83                 // Registro del puerto C
#BYTE ADCON0  = 0x0FC2                // Registro de control del ADC
#BYTE ADCON1 = 0x0FC1                 // Registro de control del ADC
#BYTE CMCON  = 0x0FB4                 // Registro del modulo comparador

// Variables globales
int dato [64];                        // Variable para almacenar los datos enviados desde el PC a el PIC
int enviado [64];                     // Variable para almacenar los datos enviados desde el PIC al PC
```

```
int adelante[6] = {0x0E,0x0D,0x0B,0x07};        // Variable del tipo vector empleada para indicarle al motor paso a paso que avance hacia adelante   (0x03,0x
int atras[6]    = {0x07,0x0B,0x0D,0x0E};        // Variable del tipo vector para indicarle al motor paso a paso que avance hacia atrás               (0x09,0x
int pos = 0;                                    // Variable para indicar la posición a la que debe avanzar el motor paso a paso o contenedor
int velocidad = 40;                             // Variable empleada para determinar la velocidad del motor paso a paso o contenedor
float  ubicacion = 1;                           // Variable empleada para indicarle al motor paso a paso o contenedor cual era la ubicación anterior
float mover=0;                                  // Variable empleada para indicarle al motor paso a paso o contenedor que avance
unsigned int16 tempo;                           // Variable empleada para almacenar el valor leido por el ADC del PIC valor en hexadecimal del voltaje en el
float sensor;                                   // Variable empleada para obtener el voltaje real en el sensor LDR

// Funciones
void configuracion (void);                      // Prototipo de la función configuración
void contenedor (int pos);                      // Prototipo de la función contenedor


void main(void) {                               // Función principal

   int flag = 0;                                // Variable local bandera empleada para la carga inicial del Timer 1
   configuracion();                             // Función de configuración del pic

   while (TRUE)                                 // Ciclo while infinito
   {
      if(usb_enumerated())                      // Si el Pic está configurado via USB
      {
         if (usb_kbhit(1))                      // Si el endpoint de salida contiene datos del host
         {
               usb_get_packet(1, dato, 64);     // Tomamos el paquete de tamaño 8bytes del EP1 y almacenamos en la variable dato
               enviado[0] =0x00;                // Limpiamos el valor que tenemos en la variable enviado para envitar problemas con valores cargados
                                                // con anterioridad

               switch(dato[0])                  // Ciclo switch el cual nos permite indicar que acciones tomar segun los datos enviados del PC al PIC
               {
                 case 2:                                        // Si se pulsa el boton de parada de la banda transportadora en la interface grafica

                      output_low(MOTOR_BANDA);                  // Se apaga del motor de la banda transportadora
                      break;

                 case 3:                                        // Si se desactiva el boton de parada de la interface grafica
                      output_high(MOTOR_BANDA);                 // Se enciende del motor de la banda transportadora
                      break;                                    // Timer 1 y con esto no permitir la banda transportadora se encienda al conectar la tarjet


                 case 4:                                        // El objeto es un circulo
                      pos = 1;                                  // Guardamos en la variable pos la posición a la cual debe ir el contedor siendo 1 la del c
                      contenedor(pos);                          // Llamamos a la funcion contenedor que se encargara de ubicar el objeto en el contenedor d
                      break;                                    // circulo que es la posición 1

                 case 5:                                        // El objeto es un hexagono
                      pos = 2;                                  // Guardamos en la variable pos la posición a la cual debe ir el contedor siendo 2 la del h
                      contenedor(pos);                          // Llamamos a la funcion contenedor que se encargara de ubicar el objeto en el contenedor d
                      break;                                    // hexágono que es la posición 2

                 case 6:                                        // El objeto es un cuadrado
                      pos = 3;                                  // Guardamos en la variable pos la posición a la cual debe ir el contedor siendo 3 la del c
                      contenedor(pos);                          // Llamamos a la funcion contenedor que se encargara de ubicar el objeto en el contenedor d
                      break;                                    // cuadrado que es la posición 3

                 case 7:                                        // El objeto es un triangulo
                      pos = 4;                                  // Guardamos en la variable pos la posición a la cual debe ir el contedor siendo 4 la del t
                      contenedor(pos);                          // Llamamos a la funcion contenedor que se encargara de ubicar el objeto en el contenedor d
                      break;                                    // triangulo que es la posición 4

               }

               tempo=read_adc();                                //Leemos el valor de voltaje en el pin RA0 y se lo asignamos a la varialbe tempo

               sensor=(tempo*0.004887585);                      //Conversión del valor leido en hexadecimal a voltios del LDR tempo*(5/1024)

              if(sensor>4.3)  //4.3V que seria cuando pasa el objeto y no llega luz al LDR
               {
                  enviado[0] = 0x02;                            // Le asignamos el valor de dos a la variable enviado en la posision cero
                  usb_put_packet(1, enviado, 64, USB_DTS_TOGGLE); // Enviamos el dato en un paquete de 64 bytes del EP1 al PC para que el PC se encargue de t
               }
               else
               {
                  enviado[0] = 0x00;                            // Le asignamos el valor de cero a la variable enviado en la posision cero
                  usb_put_packet(1, enviado, 64, USB_DTS_TOGGLE); // y de esta forma no se toma la foto
               }

         }
      }
   }
}

void configuracion (void)               //Funcion de configuración del microcontrolador
{

   TRISA  = 0xF7;                               // Se declara el puerto A como entradas
   TRISB  = 0x00;                               // Se declara el puerto B como salidas
   TRISC  = 0x00;                               // Se declara el puerto C como entradas
   TRISD  = 0x00;
   ADCON1 = 0x0F;                               // Se configura al ADC para entradas digitales (apagar el ADC)
   CMCON  = 0x07;                               // Se configuran los comparadores para entradas digitales (apagar los comparadores)
   PORTA  = 0x00;                               // Limpiamos el puerto A
   PORTB  = 0x00;                               // Limpiamos el puerto B
   PORTD  = 0x00;
   LED_ON(LEDR);                                // Encendemos led en RC1 para indicar presencia de energia
   LED_OFF(LEDV);

   usb_init();                                  // Inicializamos el USB
   usb_task();                                  // Habilita periferico usb e interrupciones
   usb_wait_for_enumeration();                  // Esperamos infinata hasta que el PicUSB sea configurado por el host


   LED_OFF(LEDR);
   LED_ON(LEDV);                                // Encendemos led en RC0 al establecer contacto o comunicación con la PC

   //setup_timer_1 (T1_INTERNAL | T1_DIV_BY_8);
   //set_timer1(3036);
   //enable_interrupts(INT_TIMER1);

   setup_comparator(NC_NC_NC_NC);
```

```
   setup_adc_ports(AN0);                        // Se selecciona AN0 como entrada analógica y las demás como digitales
   setup_adc( VSS_VDD );                        // Se indica el rango de voltaje que tendrá la entrada análoga
   set_adc_channel( 0 );                        // Indica de que pin se hará la conversion
   setup_adc( ADC_CLOCK_div_16 );               // Indica la frecuencia que se usará el reloj del ADC ADC_CLOCK_DIV_16

   enable_interrupts(GLOBAL);                    // Habilitamos las interrupciones globales

}

void contenedor (int pos)                        // Funcion Contenedor que se encarga del control de la ubicacion de los objetos en los contenedores in
{
   int i, j, k;                                  // Variables empleadas en los cilco for que se encunetra en esta función

   mover = pos - ubicacion;                      // Operación que nos permite conecer cual es la posición actual del contedor para luego indica si debe
                                                 // o retroceder y que tande debe hacerlo para cada caso
   if(mover>0)                                   // Preguntamos si mover es mayor a cero para indicar que avance hacia adelante al contendor
   {
      if(mover!=3)                               // Se excluye a 3 para evitar que se realice un mayor recorrido en las posiciones del contenedor
      {
         for(k=0;k<mover;k++)                    // Ciclo for que nos permite indicar que tantas posiciones debe avanzar el contenedor para quedar en l
         {                                       // adeacuada para dejar el objeto
            for(i=0;i<3;i++)
            {
               for(j=0;j<6;j++)                  // Ciclo for que nos permite avanzar en el motor paso a paso
               {
                  output_b(adelante[j]);//PORTB = (adelante[j]);          // Asiganamos el valor que se encuentra en la posisión de la variable adela
                  delay_ms(velocidad);           // Retardo para controlar la velocidad en que se debe dar cada paso del motor
               }
            }
         }
      }
      else                                       // Si es 3 se realiza el recorrido hacia atras debido a que es la forma mover el contenedor mas rapida
      {
         for(k=0;k<1;k++)                        // Es el mismos preoceso que en el anterior solo que se emplea una
         {                                       // variable para que en contenedor retroceda

            for(i=0;i<3;i++)
            {
                for(j=0; j<6; j++)
                {
                    output_b(atras[j]);//PORTB = (atras[j]);
                    delay_ms(velocidad);
                }

            }
         }
      }
   }

   else                                          // Si mover da un valor inferior a 0 quiere decir que el contenedor debe retroceder
   {
      if(mover == -1)                            // Si da menos uno debe retroceder una posición
      {
         mover=1;
      }

      else if(mover == -2)                       // Si da menos dos debe retroceder dos posiciones
      {
         mover=2;
      }

       else if(mover == -3)                      // Si da menos tres debe retroceder tres posiciones
      {
         mover=3;
      }

      if(mover!=3)                               // Se excluye a 3 para evitar que se realice un mayor recorrido en las posiciones del contenedor
      {
         for(k=0;k<mover;k++)                    // Es el mismos preoceso que en el anterior solo que se emplea una
         {                                       // variable para que en contenedor retroceda

            for(i=0;i<3;i++)
            {
                for(j=0; j<6; j++)
                {
                    output_b(atras[j]);//PORTB = (atras[j]);
                    delay_ms(velocidad);
                }

            }
         }
      }

      else                                       // Si es 3 se realiza el recorrido hacia atras debido a que es la forma mover el contenedor mas rapida
      {
         for(k=0;k<1;k++)                        // Ciclo for que nos permite indicar que tantas posiciones debe avanzar el contenedor para quedar en la
         {                                       // adeacuada para dejar el objeto
            for(i=0;i<3;i++)
            {
               for(j=0;j<6;j++)                  // Ciclo for que nos permite avanzar en el motor paso a paso
               {
                  output_b(adelante[j]);//PORTB = (adelante[j]);          // Asiganamos el valor que se encuentra en la posisión de la variable adela
                  delay_ms(velocidad);           // Retardo para controlar la velocidad en que se debe dar cada paso del motor
               }
            }
         }
      }
   }
   ubicacion = pos;                              // Le asignamos el valor que se encuentra en la variable pos a la variable ubicación
                                                 // para saber cual es la posición en la que se encuentra actual mente el contenedor
}
```

La programación en PIC C Compiler nos permite programar el microcontrolador 18F4550 la cual nos permite crear una interface de adquisición de datos entre MatLab y el Proceso a controlar.

Programación Matlab

```
function varargout = GUIDETESIS(varargin)
% GUIDETESIS M-file for GUIDETESIS.fig
%      GUIDETESIS, by itself, creates a new GUIDETESIS or raises the existing
%      singleton*.
%
%      H = GUIDETESIS returns the handle to a new GUIDETESIS or the handle to
%      the existing singleton*.
%
%      GUIDETESIS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUIDETESIS.M with the given input arguments.
%
%      GUIDETESIS('Property','Value',...) creates a new GUIDETESIS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUIDETESIS_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUIDETESIS_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUIDETESIS

% Last Modified by GUIDE v2.5 05-Mar-2014 12:19:55

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUIDETESIS_OpeningFcn, ...
                   'gui_OutputFcn',  @GUIDETESIS_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin(1))
    gui_State.gui_Callback = str2func(varargin(1));
end

if nargout
    [varargout(1:nargout)] = gui_mainfcn(gui_State, varargin(:));
else
    gui_mainfcn(gui_State, varargin(:));
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUIDETESIS is made visible.
function GUIDETESIS_OpeningFcn(hObject, eventdata, handles, varargin)

global t data_in data_out vid_pid_norm out_pipe in_pipe conectado my_in_pipe my_out_pipe handles1 sel

handles1 = handles;

ime=imread('logo.jpg');             %Leemos la imagen logo que es el logo de la UNAD
axes(handles1.axes1);               %Indicamos que se cargue la imagen en el axes1
imshow(ime);                        %Mostramos la imgen

% Choose default command line output for GUIDETESIS
handles.output = hObject;

guidata(hObject, handles);

conectado = 0;                      %Inicializamos la variable conectado que indicara si se detecto la tarjeta

data_in = eye(1,64,'uint8');        % Se declara el vector de datos de entrada (el que se recibe del PIC)
data_out = eye(1,64,'uint8');       % Se declara el vector de datos de salida (el que se envia al PIC)
                                    % TODOS LOS DATOS SE DECLARAN COMO
                                    % UINT8 de lo contrario no hay
                                    % comunicación.

%Indicamos a MatLab el Vendor ID y el producto ID de nuestra tarjeta
vid_pid_norm = libpointer('int8Ptr',[uint8('vid_04d8&pid_000b') 0]);
out_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);
in_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);


set(handles.figure1,'CloseRequestFcn',@closeGUI); %Declaramos la rutina que se debe ejecutar al cerrar la apliación
%Creamos el objeto Timer y se arranca
t = timer('TimerFcn',@tiempo, 'ExecutionMode', 'fixedSpacing' ,'BusyMode', 'queue' ,'Period', 0.25);
start (t);

loadlibrary mpusbapi _mpusbapi.h alias libreria
% Los archivos _mpusbapi.c y mpusbapi.dll deben de estar en la misma
% carpeta de trabajo y se obtienen de la descarga del driver en la
% pagina de microchip ("Microchip MCHPFSUSB v2.2 Installer.zip"),
% al instalarse queda ubicado en X:\Microchip Solutions\USB
% Tools\MCHPUSB Custom Driver\Mpusbapi\Dll\Borland_C, en caso de descargar
% una version de driver más reciente, reemplace éstos archivos por los más
% nuevos.

%libisloaded libreria               % Confirma que la librería ha sido
                                    % cargada
%libfunctions('libreria', '-full')  % Muestra en la línea de comandos las
                                    % funciones de la librería
%libfunctionsview libreria          % Muestra en un cuadro lo mismo que la
                                    % instrucción anterior
```

```
t = timer('TimerFcn',@tiempo, 'ExecutionMode', 'fixedSpacing' ,'BusyMode', 'queue' ,'Period', 0.25);
start (t);

loadlibrary mpusbapi _mpusbapi.h alias libreria
% Los archivos _mpusbapi.c y mpusbapi.dll deben de estar en la misma
% carpeta de trabajo y se obtienen de la descarga del driver en la
% pagina de microchip ("Microchip MCHPFSUSB v2.2 Installer.zip"),
% al instalarse queda ubicado en X:\Microchip Solutions\USB
% Tools\MCHPUSB Custom Driver\Mpusbapi\Dll\Borland_C, en caso de descargar
% una version de driver más reciente, reemplace éstos archivos por los más
% nuevos.

%libisloaded libreria               % Confirma que la librería ha sido
                                    % cargada
%libfunctions('libreria', '-full')  % Muestra en la línea de comandos las
                                    % funciones de la librería
%libfunctionsview libreria          % Muestra en un cuadro lo mismo que la
                                    % instrucción anterior

%calllib('libreria','MPUSBGetDLLVersion');

while (conectado == 0)  %Si al iniciar nuestra aplicación no se encuentra conectada la aplicación
    [conectado] = calllib('libreria','MPUSBGetDeviceCount',vid_pid_norm);   %Nos permite conocer si nuestra tarjeta esta conectada
    if (conectado ==0)  %Si no esta conectada mostramos el siguiente mensaje en una ventana emergente
        selection = questdlg('La tarjeta de evaluación no se encuentra',...
                        'Tarjeta no encontrada',...
                        'Buscar de nuevo','Cancelar','Cancelar');
        switch selection,       %Le damos al usuario la opción de cancelar o buscar de nuevo si se conecto la tarjeta
            case 'Cancelar',
                stop (t);       %Si desea cancelar detenemos la funcion de temporización
                return
            case 'Buscar de nuevo'  %Si desea buscar nuevamente limpiamos el estado de la variable
                conectado = 0;      %conectado
        end
    end
end
if conectado == 1   % Es importante seguir ésta secuencia para comunicarse con el PIC:
                    % 1. Abrir tuneles, 2. Enviar/Recibir dato
                    % 3. Cerrar tuneles
while (conectado == 0)  %Si al iniciar nuestra aplicación no se encuentra conectada la aplicación
    [conectado] = calllib('libreria','MPUSBGetDeviceCount',vid_pid_norm);   %Nos permite conocer si nuestra tarjeta esta conectada
    if (conectado ==0)  %Si no esta conectada mostramos el siguiente mensaje en una ventana emergente
        selection = questdlg('La tarjeta de evaluación no se encuentra',...
                        'Tarjeta no encontrada',...
                        'Buscar de nuevo','Cancelar','Cancelar');
        switch selection,       %Le damos al usuario la opción de cancelar o buscar de nuevo si se conecto la tarjeta
            case 'Cancelar',
                stop (t);       %Si desea cancelar detenemos la funcion de temporización
                return
            case 'Buscar de nuevo'  %Si desea buscar nuevamente limpiamos el estado de la variable
                conectado = 0;      %conectado
        end
    end
end
if conectado == 1   % Es importante seguir ésta secuencia para comunicarse con el PIC:
                    % 1. Abrir tuneles, 2. Enviar/Recibir dato
                    % 3. Cerrar tuneles

    [my_out_pipe] = calllib('libreria', 'MPUSBOpen',uint8 (0), vid_pid_norm, out_pipe, uint8(0), uint8 (0)); % Se abre el tunel de envio
    [my_in_pipe] = calllib('libreria', 'MPUSBOpen',uint8 (0), vid_pid_norm, in_pipe, uint8 (1), uint8 (0)); % Se abre el tunel de recepción
end



% --- Outputs from this function are returned to the command line.
function varargout = GUIDETESIS_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;


%Funcion que se ejecuta al usuario indicar que desea cerrar la aplicación
function closeGUI(source,eventdata)
```

```
global t my_in_pipe my_out_pipe conectado

%Abrimos una ventana la cual nos pediara confirmar el cierre de la apliación
selection = questdlg('Desea cerrar la aplicación?',...
                     'Confirmación',...
                     'Si','No','Si');
switch selection,
   case 'Si',
     stop (t);
     if conectado == 1
        calllib('libreria', 'MPUSBClose', my_in_pipe);  % Se cierra el tunel de recepción
        calllib('libreria', 'MPUSBClose', my_out_pipe); % Se cierra el tunel de envio
     end
     unloadlibrary libreria      %Descargamos la librería de memoria, para que no se generen errores
     delete(gcf)
     close all
     clear all
     %clc
   case 'No'                    %Si no desea cerrarlar regresamos
     return
end
```

```
%Función de temporización que nos permite estar revisando constantemente
%como se encuentra la tarjeta
function tiempo(hObject, eventdata, handles) %(source,eventdata)

global conectado my_out_pipe my_in_pipe data_in data_out handles1 im sel vid
int16 data;

%sel = handles.sel;

if conectado == 1       %Si permanece conectada la tarjeta
    calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out, uint8(64), uint8(64), uint8(10)); % Se envia el dato al PIC
    [aa,bb,data_in,dd] = calllib('libreria', 'MPUSBRead',my_in_pipe, data_in, uint8(64), uint8(64), uint8(10)); % Se recibe el dato que envia el PIC
    %data_out (1) = data_out (1)+ 1;     % Se incrementa el primer dato del vector que se envia

end

    if(data_in(1)==2)               %Si el sensor detect oel paso del objeto data_in(1) es igual a 2
        disp('paso objeto')
        axes(handles1.axes1);
        im=getsnapshot(vid);                    % Funcion para tomar la foto y asiganrla a la variable im
        imwrite(im,'objeto.jpg');               % Guardamos la imagen que tenemos en la variable im con el nombre de objeto y extención .jpg
        %im=imread('objeto.jpg');               % Cargamos la imagen para hacer pruebas al bo tener webcam
        %imshow(im);
        %subplot(221); image(im);

        V=3;
        SE=strel('square',V);                   % Nos permite crear un elemento estructurante de forma cuadrado de tamaño V para posteriormete realiz
        im_gray=rgb2gray(im);                   % Convertimos la imagen a escala de grises
        im_edge=edge(im_gray,'sobel');          % Aplicamos la mascara sobel para delimitar bordes de nuestro objeto
        im_dilate=imdilate(im_edge,SE);         % Aplicamos el proceso de dilatación a la imagen
        fill=imfill(im_dilate,'holes');         % Aplicamos el filtro fill el cual nos permite rellenar los agujeros o ruido en la imagen
        im_bin2=(fill);                         % Asignamos esta nuva imagen a la variable im_bin2 como imagen binaria 2
        [l ne]=bwlabel(im_bin2);                % Nos permite etiquetar la imagen de la siguiente forma 0 para el fondo y 1 para el objeto yde esta f
        propied=regionprops(l);                 % Nos permite obtener las propiedades de la imagen como area, perimetro, etc.
        %Busca areas menores a 500
        s=find([propied.Area]<1000);            % Buscamos areas menores a 1000

        for n=1:size(s,2)                               % Nos permite aplicar el cuadro delimitador para eliminar las areas
            d=round(propied(s(n)).BoundingBox);         % menores a 1000 de nuestra imagen contenida en la variables s
            im_bin2(d(2):d(2)+d(4),d(1):d(1)+d(3))=0;
        end

        [l ne]=bwlabel(im_bin2);    % Nuevamente realizamos el etiquetado de la imagen despues de eliminar el ruido de las areas menores a 1000
        propied2=regionprops(l);    % Obtenemos nuevamente las propiedades de la imagen
        CAJA=propied2.BoundingBox;  % Obtenemos el cuadro delimitador o caja donde se encuentra nuestro objeto y no tomar encuenta el resto de la imagen
        RECO = imcrop(im,[CAJA]);   % Realizamos el recorte de la imagen para solo tomar en cuenta el objeto dentro de la caja o cuadro delimitador

        umbral = graythresh(RECO);  % Obtenemos el umbral de nuestra imagen para mejorar el proceso de binarizado de imagen
        R = RECO(:,:,1);            % Separamos la imagen en los tres plano RGB
        G = RECO(:,:,2);            % por sus siglas en ingles Red, Green, Blue
        B = RECO(:,:,3);


        mR=sum(sum(R));             % Nos permite obtener el valor decimal del plano rojo o color rojo
        mG=sum(sum(G));             % Nos permite obtener el valor decimal del plano verde o color verde
        mB=sum(sum(B));             % Nos permite obtener el valor decimal del plano azul o color azul



            if((mR>mG) & (mR>mB))           % Realizamos la comparación si es mayor mR que ambos el color del objeto es rojo
               set(handles1.text3,'String','EL OBJETO ES DE COLOR ROJO');   % Indicamos que el objeto es rojo en el GUIDE
               ByN=im2bw(R,umbral);             % Pasamos a binario la imagen
               if(sel==2)                % Si la clasificación es por color
                  data_out (1) = 4;     % Le asiganamos el valor de 4 a la variable data_out para que el microcontrolador ubique el objeto en la posición 1
               end                       % para el color rojo

            elseif((mG>mR) & (mG>mB))          % Si es mayor mG que ambos el color del objeto es verde
                set(handles1.text3,'String','EL OBJETO ES DE COLOR VERDE'); % Indicamos que el objeto es verde en el GUIDE
                ByN=im2bw(G,umbral);            % Pasamos a binario la imagen
               if(sel==2)
                  data_out (1) = 5;     % Le asiganamos el valor de 5 a la variable data_out para que el microcontrolador ubique el objeto en la posición 2
               end                       % para color verde

            elseif((mB>mR) & (mB>mG))         % Si es mayor mB que ambos el color del objeto es azul
                set(handles1.text3,'String','EL OBJETO ES DE COLOR AZUL');% Indicamos que el objeto es azul en el GUIDE
                ByN=im2bw(B,umbral);            % Pasamos a binario la imagen
               if(sel==2)
                  data_out (1) = 6;     % Le asiganamos el valor de 6 a la variable data_out para que el microcontrolador ubique el objeto en la posición 3
```

```
                end                         % para color azul

            else
                ByN=im2bw(im,umbral);           % Si no se cumple ninguna quiere decir que el objeto puede ser blanco asi que
            end                             % Tomamos la imagen sin segmentar y la convertimos a binario para luego ser procesada
            %axes(handles1.axes1);
            %imshow(im);
            %ByN=im2bw(im);
            ByN=bwareaopen(ByN,1000); % Nos permite eliminar de la imagen los objetos que tengan menos de 1000 pixeles asi eliminamos ruido en la imagen
            AREA=bwarea(ByN);           % Estima la superficie o area de los objetos en la imagen binaria es un escalar cuyo valor corresponde aproximadamente c
            g = edge(ByN,'canny'); % Esta funcion nos permite invocar el algoritmo de detección de borde canny
            PERIMETRO=bwarea(g);    % Calculamos el perimetro
            P=PERIMETRO^2;          % Elevamos el permiteo al cuadrado para emplearlo en la formula para hallar el cociente del objeto
            COCIENTE=P/AREA         % Calculamos el conciente

             if ((10<COCIENTE) & (COCIENTE<13)) % Realizamos la comparación del valor de cociente para determinar que tipo de objeto es
                  set(handles1.text2,'String','EL OBJETO ES UN CIRCULO'); % Mostramos en el GUIDE que el objet oes un circulo
                    if(sel==1)  % Si el operador selecciono clasificar por forma
                        data_out (1) = 4;% Le asiganamos el valor de 4 a la variable data_out para que el microcontrolador ubique el objeto en la posición 1
                    end

             elseif ((13<COCIENTE) & (COCIENTE<15))
                  set(handles1.text2,'String','EL OBJETO ES UN HEXAGONO');% Mostramos en el GUIDE que el objeto es un hexágono
                  if(sel==1)
                        data out (1) = 5;% Le asiganamos el valor de 4 a la variable data out para que el microcontrolador ubique el objeto en la posición 2
                  end

             elseif ((15<COCIENTE) & (COCIENTE<17))
                   set(handles1.text2,'String','EL OBJETO ES UN CUADRADO');% Mostramos en el GUIDE que el objeto es un cuadrado

                    if(sel==1)
                        data_out (1) = 6;% Le asiganamos el valor de 4 a la variable data_out para que el microcontrolador ubique el objeto en la posición 3
                    end

             elseif ((18<COCIENTE) & (COCIENTE<22))
                   set(handles1.text2,'String','EL OBJETO ES UN TRIANGULO');% Mostramos en el GUIDE que el objeto es un triangulo

                   if(sel==1)
                        data_out (1) = 7;% Le asiganamos el valor de 4 a la variable data_out para que el microcontrolador ubique el objeto en la posición 4
                   end

             end



        %else
           % set(handles1.text2,'String',' ');
        %end
        else
            disp('no paso objeto')
        end



    % --- Executes during object creation, after setting all properties.
    function text2_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to text2 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    empty - handles not created until after all CreateFcns called


    %Función para el botón que activara el motor de la banda transportadora
    %este boton es del tipo toggle
    function Motor_Banda_Callback(hObject, eventdata, handles)

    global conectado my_out_pipe my_in_pipe data_in data_out handles1

    botonbanda=get(hObject,'Value'); %Obtenemos el estado del boton de para el encendido de la banda que es del tipo toggle
    handles.botonbanda=botonbanda;
    if handles.botonbanda==1        %Si es 1 quiere decir que no fue pulsado
        data_out (1) = 2;           %Se le envia el dato al PIC para que no encienda el motor de la banda transportadora
        set(handles.Motor_Banda,'String','BANDA ON'); %Se cambia el texto del boton para que cuando lo pulse el operador encienda la banda transportadora
    else
        data_out (1) = 3;           %Se le envia el dato al PIC para que encienda el motor de la banda transportadora
        set(handles.Motor_Banda,'String','BANDA OFF'); %Se cambia el texto del boton para que cuando lo pulse el operador se apague la banda transportadora
    end


    % Funcion que se ejecuta al oprimir el botón para activar la camara.
    function Activar_camara_Callback(hObject, eventdata, handles)

    global conectado my_out_pipe my_in_pipe data_in data_out handles1 vid src
    vid = videoinput('winvideo', 1, 'YUY2_640x480'); % Elegimos la camara a emplear ('winvideo', 1) y el formato esto se lo asignamos al objeto vid
    src = getselectedsource(vid); % Asigna src los parametros del video
    get(src);
    set(src, 'Saturation',45.160);          % Controla la saturacion de la foto 45,160
    set(vid, 'ReturnedColorSpace', 'RGB');  % Configuramos la imagen para que esta regrese en el espacio RGB y no en YUY2
    %vid.FramesPerTrigger = 1;
    %vid.ReturnedColorspace = 'rgb';
    %triggerconfig(vid, 'manual');
    vidRes = get(vid, 'VideoResolution');   % Obtenemos la altura y el ancho de la imagen
    imWidth = vidRes(1);                    % Ancho de la imagen
    imHeight = vidRes(2);                   % Altura de la imagen
    nBands = get(vid, 'NumberOfBands');     % Número de bandas de nuestra imagen (debe ser de 3 porque es RGB)
    hImagen = image(zeros(imHeight, imWidth, nBands), 'parent', handles1.axes1) % crear un espacio vacio para la imagen y la muestra en axes1
    preview(vid, hImagen);                  % Inicia la vista previa
    start(vid);                             % Comienza a grabar el video


    % Función que se ejecuta al oprimir el botón para desactivar la camara
    function Desactivar_camara_Callback(hObject, eventdata, handles)
    % hObject    handle to Desactivar_camara (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
global conectado my_out_pipe my_in_pipe data_in data_out handles1 vid src
stoppreview(vid);                   % Detenemos la grabación del video
delete(vid);                        % Elemina el video
ime=imread('logo.jpg');             % Leemos la imagen logo que es el logo de la UNAD
axes(handles1.axes1);               % Indicamos que se cargue la imagen en el axes1
imshow(ime);                        % Mostramos la imgen

% Función que nos permite obtener el estado de el tipo de clasificación
% para forma y color
function selector_Callback(hObject, eventdata, handles)

%sel=get(handles.selector,'Value');
global sel
sel=get(hObject,'Value');   %Ontenemos el valor seleccionado en cuanto a la variable por la cual va a clasificar
%handles1.sel = sel;        %el objeto el operador siendo 1 para Forma y 2 para Color
%guidata(hObject, handles1);


% --- Executes during object creation, after setting all properties.
function selector_CreateFcn(hObject, eventdata, handles)
% hObject    handle to selector (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

La programación en MatLab es la encargada de controlar la cámara mediante la toolbox Image Acquisition y realizar el procesamiento de las imágenes tomadas por la cámara mediante la toolbox Image Processing, también se encarga de recibir y enviar datos a la interface de adquisición de datos mediante el protocolo de comunicación USB, y generar la interface para la operación del sistema.

4.2 Análisis de los Resultados

Se cumplió con lo planteado desde el inicio del proyecto, solo se realizaron las siguientes modificaciones.

Emplear la función imcrop de MatLab que nos permite realizar el recorte de la imagen para solo tomar en cuenta el objeto dentro de la caja o cuadro delimitador y no tener problemas de brillo o ruido en la imagen, ya que al inicio se presentaba este problema por tomar toda el área de la imagen para realizar el procesado de la misma y determinar el tipo de objeto que era.

Se realizó el cambio de la interface de adquisición de datos USB debido a que al realizar el circuito impreso en el software Proteus se encontró un problema debido a que dicho software invierte los pines de conexión del conector mini USB.

Tras una clasificación de 100 objetos se produjo un error para un margen de error del 1%

Se desarrolló una interface visual de fácil manipulación y entendimiento para la operación del sistema.

4.3 Impacto Social

El sistema de clasificación de objetos es una plataforma que funcionara como prototipo para mejorar una línea de producción en la cual se realicen cambios constantes de productos.

También será empleada como plataforma didáctica para ser empleada en la universidad para que se pueda perfeccionar este tipo de sistema, el cual permitirá que los distintos estudiantes lo tomen como punto de partida y puedan realizarle modificaciones que le permitan mejorar o perfeccionarse el sistema.

4.4 Cronograma de Actividades

Tabla 1. Cronograma de actividades

PROYECTO SIMULACIÓN SISTEMA DE CLASIFICACIÓN DE OBJETOS BASADO EN VISIÓN ARTIFICIAL.	Enero / Marzo	Febrero / Marzo	Marzo / Abril	Mayo
Desarrollo de anteproyecto de trabajo de grado y entrega con concepto favorable del asesor de la coord. académica	01 de enero al 03 de marzo	XXXXXXXX	XXXXXXXX	XXXXXXXX
Desarrollo del proyecto de ingenieríaï programación de en el software MatLab, Generación de evidencias	XXXXXXXX	01 de febrero al 15 de marzo	XXXXXXXX	XXXXXXXX
Instalación de equipos conexión y cableado instalación los componentes Pruebas de comunicación documentación	XXXXXXXX	XXXXXXXX	Del 15 de marzo al 11 de abril	XXXXXXXX
Entrega final de ingeniería documentación, pruebas finales y entrega para sustentación del proyecto	XXXXXXXX	XXXXXXXX	XXXXXXXX	18 de abril a 12 de mayo

Fuente: El autor

4.5 Propuesta Económica

Tabla 2. Costo del proyecto

PROPUESTA ECONOMICA DEL PROYECTO DE VISIÓN ARTIFICIAL			
DESCRIPCIÓN DEL EQUIPO	**REFERENCIA**	**CANTIDAD**	**PRECIO**
PC de escritorio	Compaq Presario SR5020LA	1	600.000
Led Chorro blanco	-	1	1.700
Led rojo difuso	-	1	150
Led verde difuso	-	1	150
LDR o fotorresistencia	VT43N1	1	1.700
Cámara Web	Logitech C170	1	55.900
Moto reductor	Moto reductor 12V	1	80.000
Motor Paso a Paso	-	1	38.000
Microcontrolador	PIC18F4550	1	17.000
Driver motor paso a paso	ULN2803A	1	1.400
Case 18 pines	-	1	150
Base 40 pines	-	1	300
Resistencia 1KÝ	-	2	200
Resistencia 330Ý	-	2	200
Resistencia 220Ý	-	1	100
Resistencia 72Ý	-	1	200

Diodo rectificador	1N4007	4	400
Condensador cerámico 22pF	-	2	400
Condensador Electrolítico 0.1µF	-	1	200
Condensador Electrolítico 0.33µF	-	1	200
Condensador Electrolítico 0.47µF	-	1	200
Condensador Electrolítico 47µF	-	1	200
Condensador Electrolítico 100µF	-	1	200
Trimer 10KÝ	-	1	1.000
Regulador de 5V	LM7805	1	1.000
Conector mini USB	548190519	1	650
Cable mini USB 5 Pines	-	1	1.550
Cristal de cuarzo 4MHz	-	1	400
Transistor Mosfet	IRF540N	1	1.700
Multímetro Digital	TOP-SPEC TS628T	1	80.000
Programador microcontrolador USB	Pic-kit 2	1	60.000
Cableado	Varios	1	50.000
Desarrollo de software MatLab + Desarrollo software microcontrolador+ interface de comunicación PC a microcontrolador+ desarrollo de comunicaciones + puesta en marcha		1	1.500.000
VALOR TOTAL DE PROYECTO			**2.495.050**

Fuente: El autor

CAPITULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Se aplicaron los conocimientos adquiridos a lo largo de la carrera, específicamente en el procesamiento digital de señales, mediante el procesamiento de las imágenes tomadas mediante la cámara, sistemas embebidos, microcontroladores y microprocesadores mediante el desarrollo de la interface de adquisición de datos con comunicación USB, CAD Avanzado para electrónica mediante el desarrollo de la interface de operación y control del sistema en el software de ingeniería MatLab.

- La construcción de este prototipo tuvo como principal ventaja contar con componentes fáciles de adquirir en el mercado nacional y de bajo costo.

- La construcción de este prototipo tuvo como principal ventaja contar con componentes fáciles de adquirir en el mercado nacional y de bajo costo.

- Se realizó la implementación del sistema diseñado, Se obtuvieron los resultados esperados y se pudo comprobar el correcto funcionamiento del prototipo.

- El protocolo de comunicación USB empleada en el proyecto para la tarjeta de adquisición de datos presenta excelente resultados a nivel de velocidad de transmisión de datos.

- Se pueden aplicar mejoras en la programación que se elaboró para la interfaz de operación, dependiendo de los requerimientos que necesiten las personas que adquieran el sistema, ya que la programación de la interface es muy flexible.

5.2 Recomendaciones

- Se puede implementar sistemas de filtrado que permitan hacer la tarjeta de adquisición de datos inmune al ruido que se producen a nivel industrial.

- Emplear una cámara con mejores prestaciones que permita realizar más tomas por segundo y de esta forma hacer que el sistema sea mucho más rápido.

- Realizar el cambio de motorreductor de la banda transportadora por uno que se emplee a nivel industrial.

- Se podría mejorar el sistema de ubicación de los objetos en distintos contenedores mediante cilindros neumáticos lo cual mejoraría el sistema

5.3 Limitaciones

- El sistema está limitado a mover objetos de bajo peso debido al tipo de banda empleada en el prototipo.

- La cámara empleada en el proyecto está limitada a tomar diez cuadros por segundo.

REFERENCIAS

Aboudara C., Nielsen I., Huang JC., Maki K., Miller AJ., Hatcher D. (2009) Comparison of airway space with conventional lateral headfilms and 3-dimensional reconstruction from cone-beam computed tomography. Am J Orthod Dentofacial Orthop; Vol. 135 (pp.468-79).

Aplicación práctica de la visión artificial en el control de procesos industriales. (2012, Febrero). Recuperado el 21 de Enero de 2014 de la página web: http://www.infoplc.net/files/documentacion/vision_artificial/infoPLC_net_UD_1_DIDAC.pdf

Barranco García, C. A. (2011). Diseño e implementación de algoritmos de tratamiento de imágenes para ayudas técnicas visuales usando HMDS.

Bruned Mercè. (2008, 29 de Noviembre). Visión artificial en procesos industriales. Recuperado el 20 de Enero de 2014 de la página web: http://www.measurecontrol.com/vision-artificial-en-procesos-industriales-parte-i/

Cuevas Jiménez E. J, Zaldivar Navarro D. (s/f). Visión por Computador utilizando MatLAB Y el Toolbox de Procesamiento Digital de Imágenes. Recuperado el 10 de Marzo de 2014 de la página web: http://es.scribd.com/doc/176884488/Vision-por-computador-utilizando-MATLAB-y-el-Toolbox-de-PDI

Departamento de Ingeniería electrónica, Telecomunicación y Automática. (2005) Práctica 3ª. Detección de bordes en una imagen. Segmentación. Universidad de Jaen. (p.2) http://www4.ujaen.es/~satorres/practicas/practica3_vc.pdf

Espin, R (2011) ñCORTINA DE AGUA PROGRAMABLEò Universitat Politècnica de Catalunya (UPC) (PP.21-25) http://upcommons.upc.edu/pfc/bitstream/2099.1/11374/1/Memoria_PFC2.pdf

Grant, M., Boyd, S., & Ye, Y. (2008). CVX: Matlab software for disciplined convex programming. (p.32)

Hanselman, D. C., & Littlefield, B. (2005). *Mastering matlab 7*. Pearson/Prentice Hall.

Lego. (2013, 11 de Marzo). Operador Sobel. Recuperado el 10 de Marzo de 2014 de la página web: http://es.wikipedia.org/wiki/Operador_Sobel

Olivier Faugeras (1993). Three-Dimensional Computer Vision, A Geometric Viewpoint. MIT Press. ISBN 0-262-06158-9.

Procesamiento digital de imágenes. (2013, 9 de Marzo). Recuperado el 10 de Marzo de 2014 de la página web:http://es.wikipedia.org/wiki/Procesamiento_digital_de_im%C3%A1genes

Roncagliolo, p. (2012) Ejemplo práctico de procesamiento de imágenes en color: efecto publicitario ñrojo sobre grisesò (p.1) http://www2.elo.utfsm.cl/~elo328/PDI14_EjemploPocesamientoColor.pdf

Salg Eduardo. (2013, 17 de Noviembre). Visión artificial. http://es.wikipedia.org/wiki/Visi%C3%B3n_artificial

San Andrés Lascano X, Franco Pombo V. J. (2011). Tesis Seminario. Recuperado el 10 de Marzo de 2014 de la página web: http://www.dspace.espol.edu.ec/bitstream/123456789/20040/2/TESIS%20VERSION%20FINAL.pdf

Valderrama Gutiérrez, Freddy F. Modulo Académico Robótica. (2010). Universidad Nacional Abierta y a Distancia UNAD. Recuperado el 20 de Enero de 2014 de la página web: http://www.unad.learnmate.co/file.php/187/Contenido_en_linea/Robotica/modulo.html

Visión Artificial. (s/f). Recuperado el 11 de Marzo de 2014 de la página web: http://proyectojefer.wikispaces.com/file/view/Visi%C3%B3n%20artificial.pdf/160082401/Visi%C3%B3n%20artificial.pdf