ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DISEÑO DE UN SISTEMA DE VISIÓN ARTIFICIAL EN EL PROCESO DE LLENADO DE LA ESTACIÓN DE CONTROL IPA3 LUCAS NÜLLE"

INFORME DE MATERIA INTEGRADORA

Previo a la obtención del Título de:

INGENIERO EN ELECTRICIDAD, ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

MARK JOAN ANDRADE CASTILLO

TONNY WESLEY TOSCANO QUIROGA

GUAYAQUIL – ECUADOR

AÑO: 2018

DEDICATORIA

A Dios por guiarme y no dejarme caer en momentos difíciles y darme fuerzas para superar todo tipo de adversidades.

A mis padres Emilio Andrade y Elisa Castillo, por educarme de la mejor manera, apoyarme en todo momento, y gracias a ello poder cumplir este logro. A mi hermano Anthony por contagiarme de las ganas de sobresalir y ser responsable en todo momento.

A mis tíos Nerie Castillo y Myriam León, por abrirme las puertas de su hogar y considerarme como un hijo, han sido un pilar importante apoyándome desde el inicio hasta el final de esta meta.

A mis amigos que he ido conociendo durante toda mi vida, en especial a mi grupo de la universidad La Manada que han sido como una familia para mí, brindándome su fiel y más sincera amistad.

Mark Joan Andrade Castillo

A Dios por todas las bendiciones recibidas en todo momento, y por hacer posible alcanzar esta meta.

A mi Madre Lida Quiroga, por ser siempre mi apoyo, mi guía, por educarme con principios y valores para ser una persona de bien y en todo momento demostrarme su amor incondicional, a mis hermanos Paul Toscano y Viviana Toscano, por ser siempre un ejemplo y apoyarme en los momentos más difíciles.

A mi abuela Graciela Castillo por inculcarnos valores y demostrarme que el amor de Dios siempre debe reinar en nuestros corazones, a mi tía Norma Toscano por abrirme las puertas de su casa, considerarme su hijo y acompañarme en todos los momentos de mi vida.

A mis amigos de toda la vida, y a los amigos que la universidad me permitió conocer, La Manada que en todo momento nos hemos apoyado para escalar mas alto y no dejarnos caer, y siempre con la humildad en nuestros corazones.

Tonny Wesley Toscano Quiroga

TRIBUNAL DE EVALUACIÓN

MSc. Carolina Godoy	**PhD. Ángel Sappa**
PROFESOR DE MATERIA INTEGRADORA	TUTOR ACADÉMICO

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Mark Joan Andrade Castillo

Tonny Wesley Toscano Quiroga

RESUMEN

La implementación de sistemas de visión artificial en la industrial ha permitido obtener mejores resultados en el control de procesos, optimizando el tiempo, obteniendo resultados con mayor precisión y optimizando recursos.

En la Facultad de Ingeniería en Electricidad y Computación (FIEC), se encuentra el laboratorio de control de procesos industriales el cual cuenta con una planta Lucas Nülle, con diferentes estaciones de control, para obtener un sistema de envasado continuo, por lo cual se consideró realizar como proyecto un sistema de visión artificial para la detección y eliminación de errores en la estación IPA3 el cual corresponde al proceso de llenado de botellas.

El proyecto consiste en diseñar un sistema automático de detección de errores en las botellas que ingresan a la estación de control IPA3 para ser llenadas, utilizando dos cámaras para el sistema de visión artificial, creado en lenguaje C# basado en la librería AForge en conjunto con el PLC S7-300 para el control de los sensores y actuadores ubicados en la planta.

En el capítulo 1, una vez planteado los problemas a resolver se muestran los objetivos del proyecto, y el alcance del mismo, considerando que la solución planteada tiene como objetivo principal eliminar errores en el proceso de llenado.

En el capítulo 2, se describe brevemente cual será el funcionamiento del sistema y las herramientas utilizadas para su implementación.

En el capítulo 3, una vez analizadas de las condiciones iniciales de la estación de control IPA3, se describe detalladamente las etapas que se ejecutaron en el sistema para realizar su implementación, y los recursos utilizados en el mismo.

En el capítulo 4, después de realizar pruebas, detectar errores en la programación y corregirlos, se muestran los resultados de la implementación en el sistema de visión y en el sistema de control.

ÍNDICE GENERAL

CAPÍTULO 1

1. DELIMITACIÓN DEL PROBLEMA.

1.1 Planteamiento del problema.

Los sistemas automáticos industriales en Ecuador son de gran importancia para la economía de las empresas. A pesar de contar con la tecnología necesaria para cumplir con requisitos y estándares de calidad, los sistemas implementados no son suficientemente eficientes para no sobrepasar el máximo rango de errores permitidos en dichos sistemas. Por este motivo el Mipro estableció charlas entre empresas proveedoras de servicios y productos tecnológicos y representantes de las diferentes empresas del sector, permitiendo así conocer la realidad en la que se encuentran dichas empresas para poder establecer estrategias que dejen de limitar el desarrollo del sector tecnológico [1].

El llenado de botellas es una de las principales operaciones de envasado de productos líquidos de la industria de envase. Cuando se diseña una línea de embotellado se debe tener en mente la idea de que diferentes errores pueden aparecer dependiendo de las condiciones en las cuales se encuentre dicha línea [2].

Considerando la planta de control IPA3 Lucas Nülle las botellas entran por la cinta transportadora colocada en un recipiente de seis espacios (sixpack), las cuales pasan por dos sensores capacitivos para comprobar que efectivamente se encuentren las 6 botellas en sus respectivos espacios. Considerando 3 tipos de posibles fallas el cual la primera es que no se comprueba si las botellas se encuentran tapadas, segundo que ingresen a este proceso con un nivel de líquido. Al no controlarse dichos detalles, e intentar envasar en el caso de que la botella llega tapada, el líquido se derramará irremediablemente. En el caso de que la botella no ingrese vacía al proceso de llenado al envasar con una cantidad que sea superior a la del envase (20ml), dicho líquido se derramará y ocasionaría accidentes eléctricos o de otro tipo dependiendo del área de trabajo.

1.2 Objetivos.

1.2.1 Objetivo general.

Desarrollar un sistema de visión que permita un llenado libre de fallas en la estación de control IPA3 Lucas Nülle, optimizando los recursos de la estación de control.

1.2.2 Objetivos específicos.

- Determinar los errores que se presentan en la estación de control IPA3 Lucas Nülle.
- Integrar un sistema de visión con cámaras industriales para la detección de errores en el proceso de llenado.
- Construir una aplicación de visión usando la biblioteca AForge.NET en el software de desarrollo Microsoft Visual Studio.
- Reconstruir la programación en el PLC para que permita incorporar en el proceso al sistema de visión.
- Asociar un sistema de comunicación TCP/IP que permita unir la aplicación de visión y control.

1.3 Justificación.

El continuo avance tecnológico ha llevado a que las empresas evolucionen y mejoren sus formas de producción, es por lo que la automatización en los últimos años ha tomado cada vez más fuerza para mejorar la producción en cantidad, precisión y sin fallos. Las empresas son más competitivas cuando incorporan nuevas tecnologías, ya que esto genera mayor productividad, mayores recursos debido al aumento de ingresos, ya que las nuevas tecnologías facilitan las tareas de las empresas lo cual produce innovación [3].

Con los errores en la producción se debe tener mucho cuidado para evitar daños en la planta y en el producto final, por tal motivo se implementará un sistema de visión para reconocer y eliminar errores en el llenado de las botellas. Con este sistema se mejorará la calidad del producto, debido a que, si las botellas ingresan vacías a este proceso de llenado, entonces se evitará la contaminación

del producto final y el envase contará con la cantidad exacta para el llenado. Por último, las pérdidas en la producción y los riesgos de accidentes disminuirán al verificar previamente que todas las botellas se encuentren destapadas, evitando daños en las tarjetas de control, los motores que controlan la cinta transportadora o los sensores.

1.4 Alcance.

En la primera etapa de este proyecto se analizará por medio de pruebas las fallas que ocurren en la planta, para de esta manera optimizar el diseño del sistema, en la segunda etapa con los resultados obtenidos en el análisis previo, se diseñará el sistema de visión para lograr un proceso libre de fallas, para dicho sistema de visión se utilizará el software Visual Studio y para la programación del PLC el software TIA Portal, en la última etapa se implementará el sistema de visión en la estación de control IPA3 Lucas Nülle.

CAPÍTULO 2

2. ESTADO DEL ARTE

El laboratorio de Control de Procesos Industriales de la Escuela Superior Politécnica del litoral dispone de la planta Lucas Nülle, la cual cuenta con el sistema de mezcla de dos productos, con un almacenamiento inicial el cual no se encuentra actualmente funcionando, en el proceso de llenado del líquido en la botellas, se verifica que ingresen las 6 botellas en el six-pack para proceder a llenar de líquido, a continuación pasan al sistema de tapado y finaliza en un almacenamiento para el despacho, el mismo que se encuentra inactivo.

En la Industria los sistemas de visión artificial aumentan la producción reduciendo los tiempos, aumentan la efectividad de productos con alta calidad. En la industrial textil se implementó un sistema de visión artificial para verificar la calidad de la tela, la cual toma fotos de cada metro de tela para detectar las fallas de esta según datos establecidos por el programador, los resultados son mostrados en una pantalla y almacenados [4].

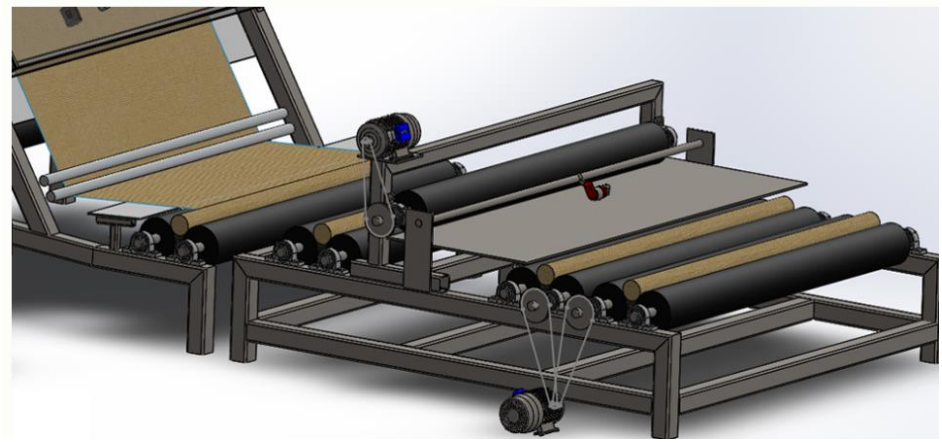


Figura 2.1 Máquina para la verificación de calidad de tela

Así mismo en la industria automotriz se controla la calidad final del pintado de los coches, en la cual se requiere eliminar imperfecciones o manchas antes de

entregar el coche, mediante el sistema de visión artificial las cámaras capturan imágenes para formar un modelo 3D del vehículo y de esta forma poder detectar cualquier tipo de imperfección en el acabado final de pintura puede ser detectado y corregido [5].

Figura 2.2 Cámara de chequeo de calidad de pintura en carrocería Ford

Hace siglos atrás se elaboraba calzado personalizado tomando manualmente medidas del cliente, gracias a la visión artificial se ha desarrollado un sistema para determinar la medida de los pies en 3 dimensiones basándose en un patrón textual. Con la utilización de una cámara y un sistema de iluminación adecuado se toman dichas dimensiones y automáticamente las medidas se las envía a la máquina de elaboración del calzado para que el cliente tenga su calzado a las medidas adecuadas [6].

Figura 2.3 Espacio para toma de medidas de calzado personalizado

En la industria farmacéutica se desarrolla sistemas de visión artificial para detectar defectos, contaminantes y entre otras irregularidades en la fabricación de los productos, para ello mediante cámaras y su respectivo sistema se puede confirmar presencia de pixeles, presencia de iconos, sellos o tapas para asegurar la calidad de productor e incluso el empaque sea correctamente sellado [7].

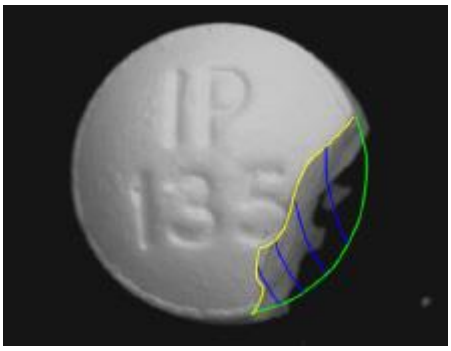


Figura 2.4 Muestra de producto con fallo

En trabajos de impresión es importante verificar que el trazado de la impresión sea la correcta y se encuentre al mismo nivel del exigido por el cliente por eso Moderngrab requería alcanzar a reducir tiempo en la verificación de dichos trabajos y conseguir la calidad deseada, con la utilización de ScanProffTM Moderngrab ha logrado implementar la herramienta para controlar la calidad de la impresión además que se redujo considerablemente el tiempo destinado [8].

Figura 2.5 Revisión de calidad de impresión

2.1. Marco teórico.

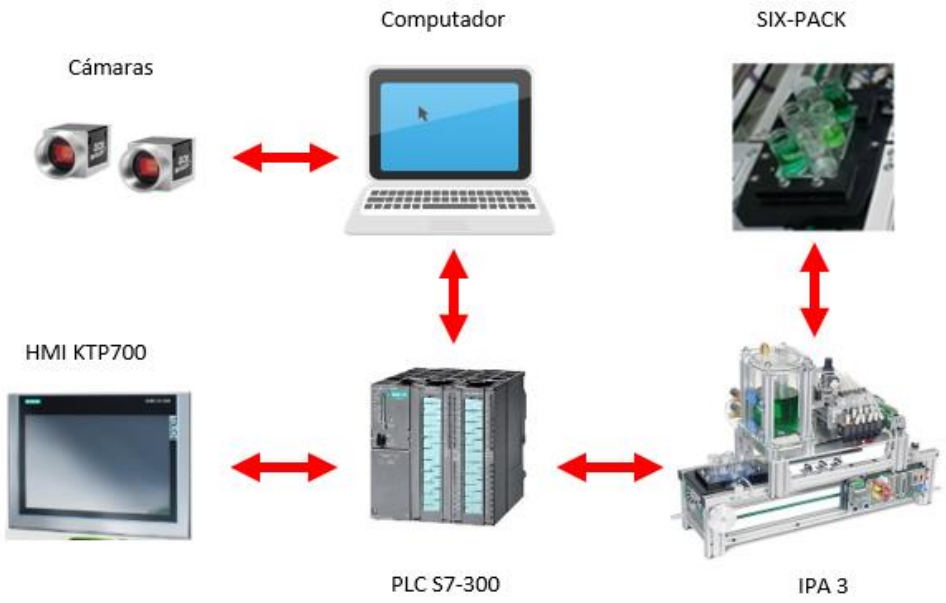


Figura 2.6 Diagrama de bloques del sistema

El diagrama de bloques del sistema es mostrado en la figura 2.6, donde se puede apreciar los componentes del proyecto y el flujo de la operación de este. Al incorporar el sistema de visión en el sistema, se requiere que el computador sea el controlador principal del proyecto, ya que éste maneja toda la información proveniente de las cámaras y a su vez enviará las señales necesarias hacia el PLC para su respectivo accionamiento.

2.2. Descripción del sistema.

El sistema lo conforman una aplicación de visión artificial, mediante dos cámaras industriales Basler, las cuales se encargarán de la identificación de los posibles errores que sucedan al momento del proceso de la planta, extrayendo la información de la matriz de pixeles que cada botella posee en condiciones correctas.

Dichos datos serán procesados y dependiendo de los resultados obtenidos, se enviará la información necesaria hacia el PLC, a través de una red TCP/IP (creada entre los dispositivos). Posteriormente el PLC se encargará de accionar la banda transportadora en la dirección correcta según los resultados obtenidos en el sistema. En la figura 2.7 se muestra el estado original de la estación de trabajo IPA3.

Figura 2.7 Estado original de la estación de trabajo IPA3

Basándose en las investigaciones de Ford, en la cual se considera que la iluminación es necesaria para la correcta utilización de las cámaras, debido a que las sombras provocadas sobre el sistema a analizar se ven afectas considerablemente, en el proceso de llenado IPA3 Lucas Nülle se implementará luces leds en el área de trabajo de las cámaras para que el sistema de visión artificial obtenga los resultados deseados.

CAPÍTULO 3

3. METODOLOGÍA DE TRABAJO.

En este capítulo se describirá y explicará la comunicación TCP/IP, serial y DB9, los métodos de visión artificial junto con la programación en TIA Portal necesaria para el reconocimiento y corrección de errores en las botellas del proceso de llenado en la estación IPA3 Lucas Nülle.

3.1 Proceso de envasado en estación IPA3.

En el proceso de envasado se cuenta con un recipiente de almacenamiento que contiene el líquido con el cual se ha de envasar cada botella, respetando el volumen que se establezca en el HMI para su llenado, pero con el software de visión artificial en funcionamiento, permitirá eliminar los tres errores mencionados en el capítulo 1.

Cuenta además con un limitador de presión con nanómetro, válvula de admisión de agua, válvula de salida de agua, sensores de nivel los cuales permiten verificar si el recipiente de almacenamiento está completamente vacío o si éste contiene la cantidad de líquido necesario para empezar con el llenado de las botellas, además tiene sensores de posición, puerto profibus y puertos para su alimentación y las conexiones de los sensores tal como se muestra en la figura 3.1.

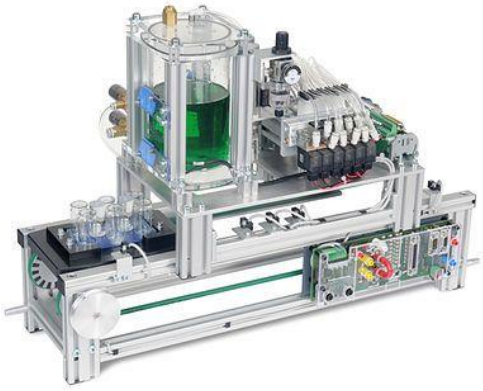


Figura 3.1 Componentes esenciales de la estación de envasado [9]

3.2 Descripción del funcionamiento de la Estación de llenado IPA3 Lucas Nülle.

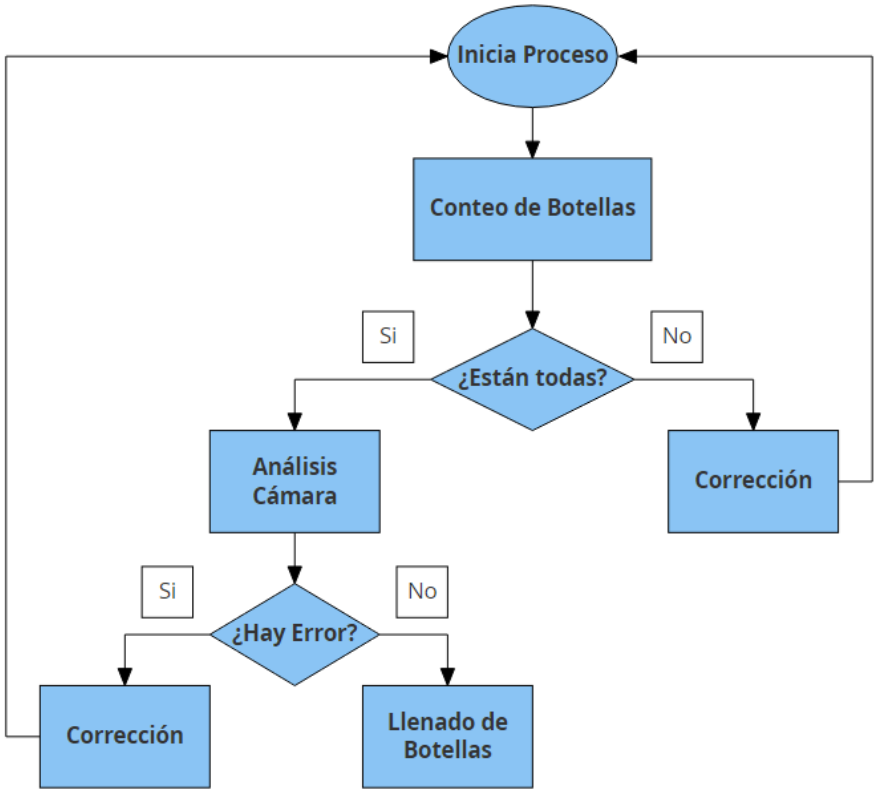


Figura 3.2. Diagrama de flujo del funcionamiento del sistema de llenado

El diagrama de la figura 3.2 describe el funcionamiento de la estación de llenado IPA3 Lucas Nülle y las cámaras Basler, inicialmente se realiza el conteo de botellas para verificar que estén las seis y la cámara pueda realizar el análisis respectivo. Una vez la cámara comience a trabajar valida si alguna de las seis botellas presenta al menos un error para que sea corregido y se repita el proceso. En caso de no existir errores se procede a llenar cada una de las botellas con el nivel de líquido especificado por el usuario.

3.3 Sistema de visión artificial.

3.3.1 Descripción.

Según AIA (Asociación de Imágenes Automatizadas), la visión artificial incluye todo tipo de aplicación industrial y no industrial donde se combinan software y hardware las cuales brindan una guía operativa a dispositivos en la ejecución de sus funciones basándose en la captura y el procesamiento de imágenes [10].

La visión artificial se compone de un conjunto de procesos destinados a analizar imágenes, donde los procesos son: captura imágenes, memorización de la información, procesar, interpretar y mostrar los resultados [11].

Esta aplicación considera los diferentes errores que pueden suscitarse en el proceso de llenado de las botellas en la planta, estas botellas son captadas por las cámaras para ser procesadas, obteniendo la matriz de pixeles de cada una. En la Fig. 3.3, se muestra los tres diferentes errores que puede suceder en este proceso.

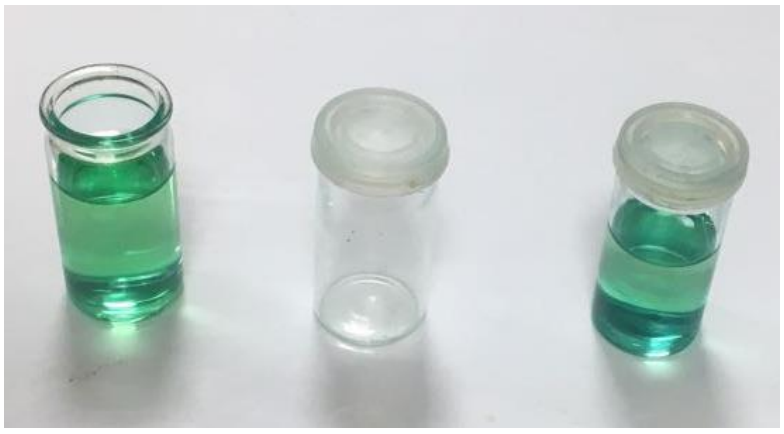


Figura 3.3 Posibles errores del sistema de llenado

Error	Descripción
N°1	La botella se encuentra sin tapa, pero contiene líquido
N°2	La botella se encuentra vacía, pero contiene tapa
N°3	La botella se encuentra tapada y contiene líquido

Tabla #1 Errores en el sistema de llenado

3.3.2 Etapas del sistema de visión artificial.

El sistema de visión artificial propuesto establece seis etapas principales las cuales se muestran en la figura 3.4. Dichas etapas ayudan a realizar un análisis más preciso de las botellas que ingresan a la estación. Obteniendo un mejor resultado en el conteo de pixeles de cada botella para determinar si contiene errores o no.

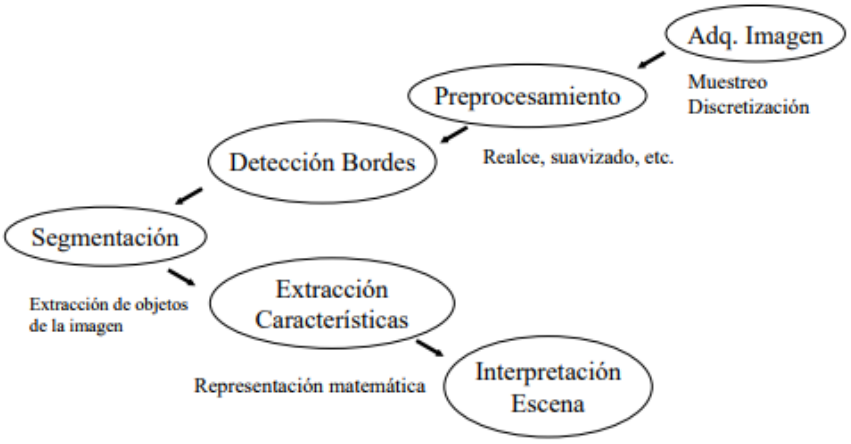


Figura 3.4 Etapas de un sistema de visión por computador [12].

3.4 Adquisición de imagen.

El proceso empieza cuando la imagen proyectada en el sensor es capturada, por medio de las ópticas para poder transferirlas al sistema electrónico. Las cámaras utilizadas requieren una serie de características que permiten que el control de estas se dispare para capturar piezas u objetos que pasan por delante de ella en la posición requerida.

3.4.1 Cámara Basler acA1300-75gc.

Para el correcto cumplimiento de esta etapa, la cámara es el principal componente ya que cuenta con un sensor CCD Sony ICX445, que ofrece 30 fotogramas por segundo y con una resolución de 1.3mp. Además, cuenta con una interfaz de comunicación gigabit Ethernet, lo que permite capturar y enviar gran cantidad de datos en tiempo cortos [13].

Figura 3.5 Cámara Basler acA1300-75gc [13].

Los sensores CCD (charge - Couple - Device) se basan en semiconductores fotosensibles para cuya lectura no es necesario un barrido electrónico.

3.4.2 Óptico Basler.

Las lentes de alta resolución son ideales para las aplicaciones actuales de visión artificial. Estos lentes se adaptan perfectamente a todas las cámaras Basler con un tamaño de sensor de menos de ½". Las lentes Basler C-125-0818-5M ofrecen una distancia focal fija de 8mm, un rango de apertura de F1.8- F22 y una resolución de 5megapixeles, el primero controla el enfoque de escena y el otro controla la cantidad de luz que recepta el sensor.

Figura 3.6 Lente Basler C125-0818-5M [14].

3.5 Preprocesamiento de imágenes.

En la etapa de preprocesamiento existen diversas técnicas que pueden ser implementadas por algoritmos computacionales, con los cuales se puede modificar, interpretar, analizar, y guardar información de las imágenes, lo cual permite resolver problemas específicos dependiendo de la aplicación a la cual esté orientada. Para este proyecto se eligió soluciones basadas en software libre (Ver código en Anexo 1).

3.5.1 AForge.

AForge fue desarrollado en código abierto C# para ser utilizado por desarrolladores e investigadores en los campos de inteligencia y visión artificial, el mismo que cuenta con diferentes librerías de las cuales se utiliza para este proyecto AForge.video y AForge.video.DirectShow [15].

3.5.2 Software.

Para la eliminación de errores en el proceso de llenado de botellas en la estación IPA3 se ha seleccionado AForge de la versión 2.2.5 ya que presta una mayor solidez que sus versiones anteriores y ha ampliado las librerías para otorgar mayores funciones de visión artificial.

Se ha escogido el programa Visual Studio 2017 y el lenguaje de programación C#, los cuales funcionan en un computador con sistema operativo Windows 10 de 64 bits, sin embargo, puede también funcionar en cualquier sistema operativo en el cual se encuentre instalado el programa Visual Studio 2017.

3.6 Detección de bordes.

Canny Edge fue desarrollado por John F. Canny en 1986, es conocido también como el detector óptimo, este algoritmo pretende satisfacer tres criterios principales: Baja tasa de error, Buena localización, Respuesta mínima [16].

Para lograr una correcta detección de bordes de una imagen que contiene diferentes objetos, es importante eliminar el ruido gráfico, ya que éste puedo provocar detección de objetos no deseados, lo cual puede generar problemas

al momento de analizar y emitir las señales de salida erróneas (Ver código en Anexo 2).

3.7 Segmentación.

La Segmentación divide la imagen en partes hasta un nivel de subdivisión en el cual, se aíslen objetos o regiones de interés para el análisis de esta [17]. En este proyecto se optó por segmentar cada imagen en 3 partes. La primera cámara obtiene la captura de las tres primeras botellas las mismas que son separadas una por uno por un proceso de manipulación de la matriz de pixeles de la imagen (Ver código en Anexo 3).

3.7.1 Obtención de matriz de pixeles de una imagen.

Para mostrar la matriz de pixeles de cualquier imagen, se utiliza las siguientes líneas de código:

```
b = (Bitmap)pictureBox2.Image;
label1.Text = b.Size.ToString();
```

Su resultado es:

```
{Width=1280, Height=720}
```

El significado de los comandos a usarse viene dado por:

Bitmap convierte una imagen en una matriz de pixeles.

Picturebox es el espacio en el que se muestra la imagen.

Size calcula el tamaño de la imagen en pixeles.

ToString convierte el valor a una cadena de caracteres para ser mostrado en label.

Label se usa para personalizar el texto a mostrar dependiendo de una etiqueta.

3.7.2 Manipulación de matriz de pixeles.

La manipulación de la matriz de pixeles de una imagen permite acceder a un espacio específico de dicha imagen para poder analizarla

dependiendo de lo requerido para la aplicación. Para dicha manipulación se requiere conocer el tamaño de la matriz de pixeles.

Una vez conocido el tamaño de la matriz de pixeles se divide la imagen en tres partes para analizar cada botella por separado. Mediante un lazo *for* se recorre la matriz de pixeles desde un valor de fila inicial hasta un valor máximo, y de la misma manera se limita un mínimo y un máximo para las columnas, para de este modo acceder a cada botella por separado. En este recorrido se buscan los pixeles de valor -1 correspondiente al color blanco, lo cual permite conocer si en la botella existe una alteración de acuerdo con los errores mencionados en el capítulo 1.

3.8 Extracción de características.

Para conocer las características de cada botella en un estado ideal, se procede a colocar las seis botellas en perfectas condiciones las cuales son: sin tapa, completamente vacías, y sin manchas o basura. Una vez realizado este proceso, se realiza la captura de las dos imágenes (una por cada cámara) para analizar una botella a la vez. Con el código (Ver Anexo 4) se contabiliza cuantos pixeles blancos existen en la imagen para guardar este valor como referencia y comparar cada vez que ingrese un six-pack en condiciones desconocidas para el sistema.

3.9 Pruebas con los datos adquiridos.

Con los valores obtenidos para los rangos en el análisis de cada botella se procede a analizar las botellas que van a ingresar en el sistema para verificar si tienen uno de los tres errores mencionados en el capítulo1. Por lo cual, si las botellas ingresadas no se encuentran dentro de este rango, se procede a verificar si la botella que presenta el error se encuentra con tapa, o con líquido en su interior o a su vez ocurren los dos errores de manera simultánea.

3.10 Herramientas de Trabajo SIEMENS.

El PLC y su software de desarrollo TIA Portal tienen como objetivo programar dicho dispositivo y controlar las variables provenientes de Visual Studio para eliminar los errores que suceden en la estación de llenado IPA 3 Lucas Nülle.

3.10.1 PLC SIMATIC S7-300 SIEMENS.

Conocido normalmente como controlador programable o PLC, sin embargo, en la actualidad es conocido bajo el nombre de “Totally Integrated Automation” SIMATIC [18].

Figura 3.7 PLC S7-300

3.10.2 Software de Control del PLC.

Este software optimiza el procesamiento, la operación y planificación de los procedimientos en una empresa. Utilizando junto a un HMI la interfaz de usuario permite una sencillez de control y operación de las funciones con una completa transparencia de datos [19].

TIA Portal nos permite realizar la programación en bloques para el PLC y la programación gráfica para el HMI logrando de esta manera controlar la estación de llenado IPA3 Lucas Nülle.

3.10.3 Grafcet.

El Grafcet está compuesto de una etapa en la cual se define el automatismo y se encuentran marcadas con un doble cuadrado, una acción asociada lo cual define que se va a realizar en esa etapa y de una transición las cuales son condiciones que deben cumplirse para poder pasar al siguiente estado.

A continuación, se mostrará la programación Grafcet para incluir el sistema de visión en la programación de la planta previamente establecida, realizada las respectivas modificaciones según sea el caso. Empezando por situar a las botellas en la posición correcta para que sean analizadas, dependiendo el caso si no hay ningún problema las botellas se trasladan a la estación para que sean llenadas con el nivel especificado por el usuario, caso contrario se trasladan a una posición inicial para que los errores sean corregidos y volver a analizar para corroborar si no existen más errores.

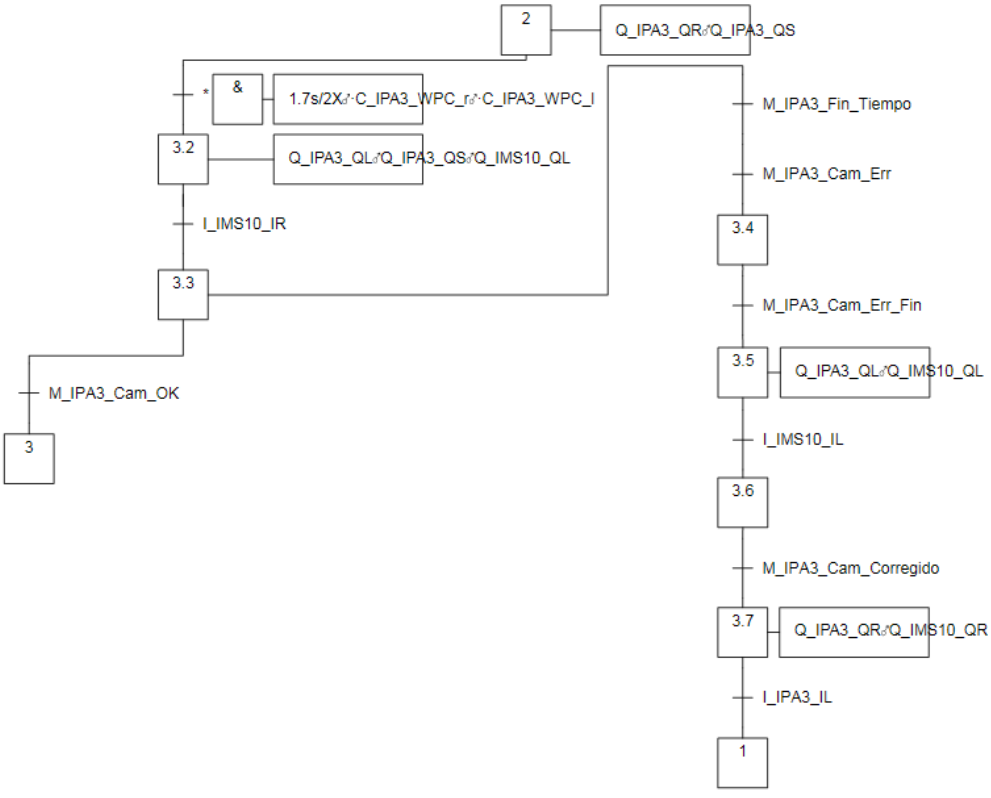


Figura 3.8 Programación Grafcet en TIA Portal

3.11 Software de visión artificial.

El software de Visión artificial tiene como objetivo eliminar los errores que suceden en el proceso de llenado en la planta ubicada en el laboratorio de Control de Procesos, este software permite mejorar en el proceso mencionado.

3.11.1 Descripción.

Las cámaras industriales capturan la imagen de las botellas a ser procesadas una a la vez, dicha imagen es procesada en el código C# en Visual Studio, y una vez obtenido los resultados de dicho procesamiento, una señal es enviada hacia el PLC para que éste muestre en el HMI el error encontrado o si el proceso continúa con normalidad.

Si el mensaje mostrado corresponde a algún error en cierta botella, el PLC envía la señal a los motores de la cinta transportadora para que el sixpack regrese al inicio del proceso y se realicen los respectivos correctivos, y una vez reseteado el error mostrado en el HMI el proceso retomará el estado en el que se encontraba para verificar si no existe otro error.

Figura 3.9 PLC y HMI del laboratorio de control de procesos industriales

3.11.2 Ubicación de los elementos.

Las botellas que son analizadas ingresan en un sixpack el cual se encuentra sobre una placa portadora de piezas de trabajo, las cámaras industriales se encuentran una de cada lado del sixpack tal como se muestra en la figura 3.10.

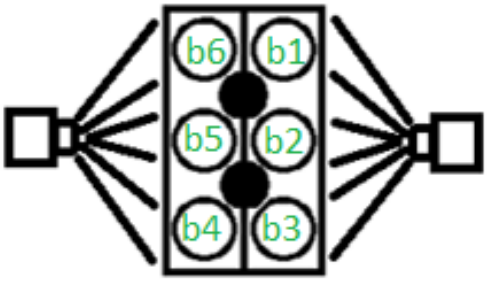


Figura 3.10 Posición de botellas y cámaras

3.12 Comunicación.

Los dispositivos del sistema están conectados mediante una red TCP/IP, serial y DB9, tal como se muestra en la figura 3.11, 3.12 y 3.13. Inicialmente el PLC establece comunicación con el HMI y la estación de control IPA3 Lucas Nülle para iniciar el proceso de conteo de botellas. El PLC se comunica con el computador para que permita trabajar a las cámaras. Por último, el computador se comunica con el PLC mediante la librería S7.net, realizando una programación cliente servidor respectivamente.

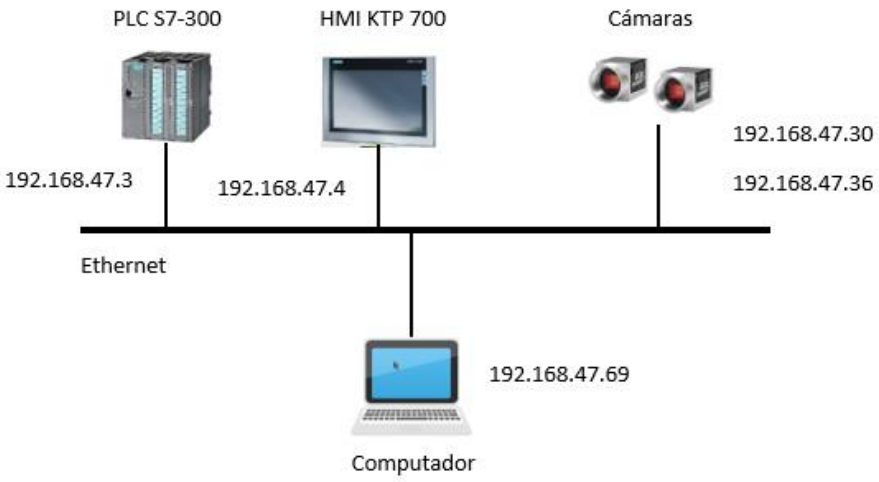


Figura 3.11 Diagrama de Red Ethernet del Sistema

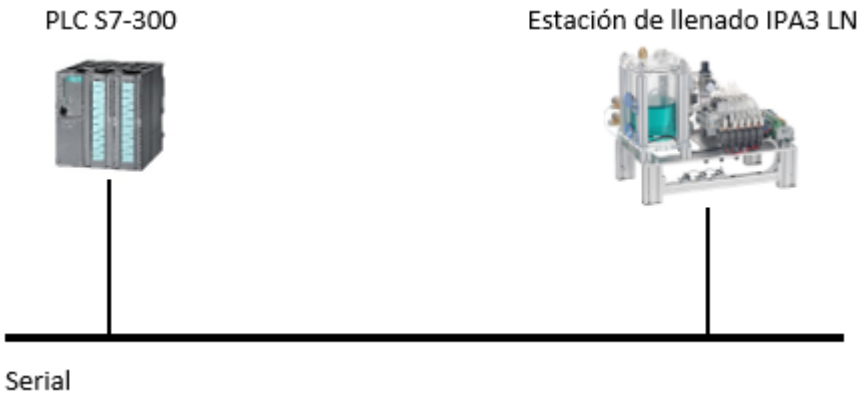


Figura 3.12 Diagrama de Conexión Serial del Sistema

Figura 3.13 Diagrama de Conexión DB9 del Sistema

3.12.1 Conexión entre Cámaras- PC.

Para efectuar la comunicación de las cámaras con el computador es necesario instalar el paquete de software Pylon 5.0 para asignarles las direcciones IP a las cámaras y ajustar el brillo de cada una de ellas [20].

3.12.2 Conexión entre PLC-Estación de llenado IPA3 LN.

La comunicación serial permite comunicar al PLC y la estación de llenado IPA3 LN en el cual se podrá controlar las variables necesarias que permiten el llenado en cada botella.

3.12.3 Conexión entre PLC-IMS 1.2 Sistema de transporte de DC.

Para efectuar la comunicación del PLC con la cinta transportadora es necesaria la conexión mediante el cable DB9, para controlar el movimiento de la banda hacia la derecha o izquierda, dependiendo de la activación de los sensores magnéticos ubicados en los extremos del IMS.

3.12.4 Conexión entre PC-PLC.

La comunicación entre el computador y el PLC es mediante la Red Ethernet para la transmisión de datos, y el uso de la librería S7.NET como puerta de conexión entre TIA Portal y Visual Studio.

CAPÍTULO 4

4. ANÁLISIS DE RESULTADOS.

En este capítulo se muestran los resultados obtenidos en la simulación del sistema completo en la estación IPA3 de la planta Lucas Nülle, en el mismo que se comprueba la eficiencia del sistema para su correcto funcionamiento.

4.1 Funcionamiento del sistema de visión artificial

4.1.1 Adquisición de imagen

Para la fase de adquisición de imagen en la figura 4.1 y figura 4.2 se muestran las botellas en diferentes estados de error, la cual es obtenida mediante la librería Aforge.net para el manejo de sus datos. Debido a la limitación de ancho de banda para el funcionamiento simultáneo de dos cámaras IP con el PLC, el resultado de imagen obtenida tiene un retraso de 3 segundos, el cual no afecta al rendimiento del sistema.

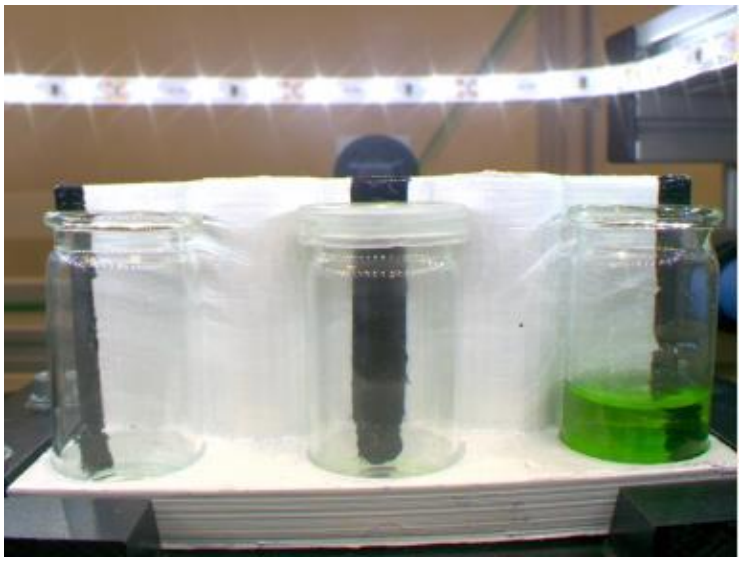


Figura 4.1 Captura de botellas obtenida por la primera cámara

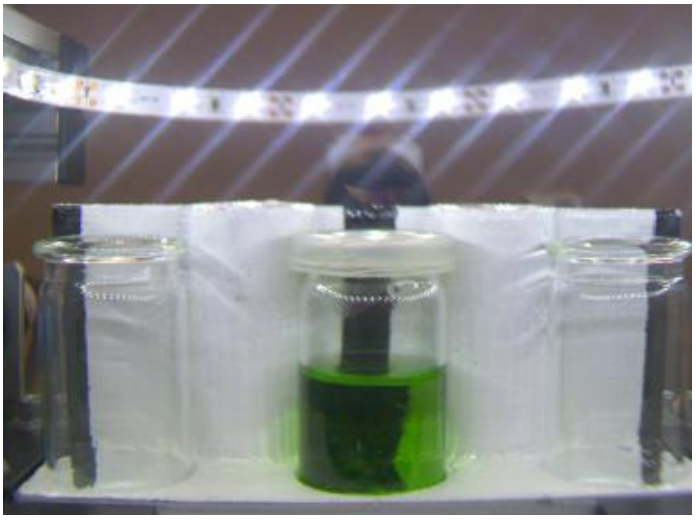


Figura 4.2 Captura de botellas obtenidas por la segunda cámara

4.1.2 Preprocesamiento de imágenes

Para el preprocesamiento de imágenes se utilizó el convertidor a escala de grises para obtener un suavizado requerido para el posterior procesamiento total de dicha imagen, en la figura 4.3 y figura 4.4 se muestra los resultados del filtro mencionado para las primeras tres botellas y las últimas tres botellas respectivamente.

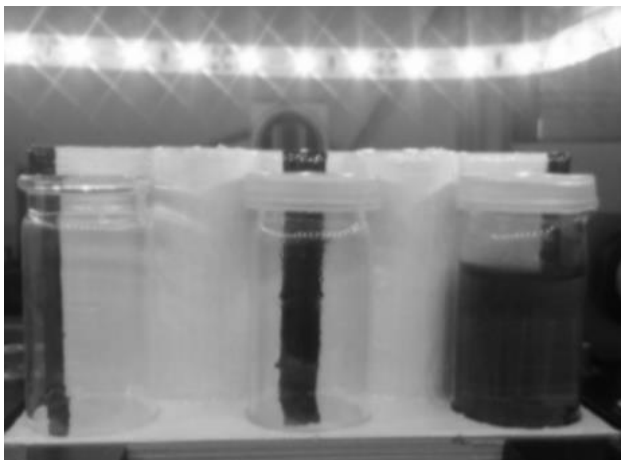


Figura 4.3 Resultado del primer filtro para las primeras tres botellas

Figura 4.4 Resultado del primer filtro para las últimas tres botellas

4.1.3 Detección de bordes

Con el filtro utilizado en el preprocesamiento de imágenes se eliminó parte del ruido gráfico, sin embargo, para una correcta detección de bordes también se utilizó el filtro gaussiano para eliminar en un 99% el ruido, y complementario a esto se encontró los gradientes con mayores magnitudes para la correcta marcación de los bordes en la imagen.

En la figura 4.5 se muestra el resultado de las primeras tres botellas y en la figura 4.6 se muestra el resultado de las últimas tres botellas de la implementación del filtro, las imágenes obtenidas varían en dependiendo de la iluminación, por lo que inicialmente no se obtiene la exactitud deseada, pero la cantidad de pixeles se encuentran dentro de un rango que se estableció después de tomar varias muestras, permitiendo que el sistema tenga 100% de precisión.

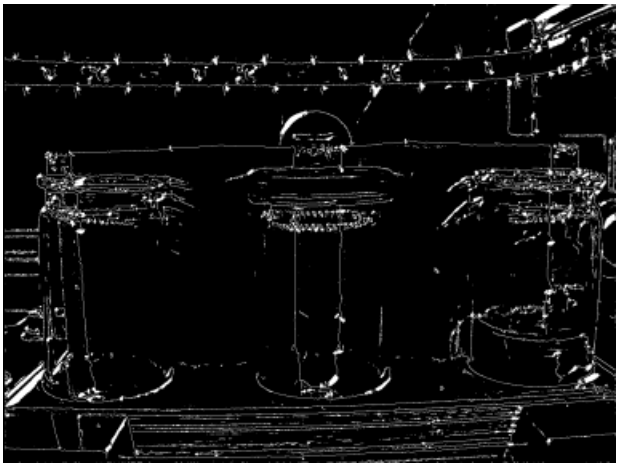


Figura 4.5 Detección de bordes de las primeras tres botellas

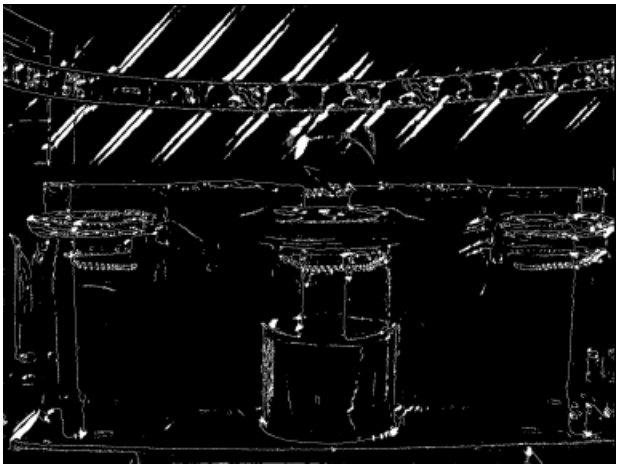


Figura 4.6 Detección de bordes de las primeras tres botellas

4.1.4 Segmentación

Para la correcta determinación de errores en cada botella se dividió la imagen obtenida por cada cámara en 3 partes, para de esta manera analizar las botellas por separado indicando claramente la numeración a la que corresponde cada parte.

Cada botella se analizó por separado desde la base hasta milímetros antes del pico y el pico de la misma, para determinar si se encuentra con tapa o sin tapa y a su vez con la otra parte de la segmentación verificar si se encuentra con líquido o sin líquido.

En la figura 4.7 se muestra la separación (línea roja) de cada botella para determinar correctamente si se encontraba tapada o con líquido.

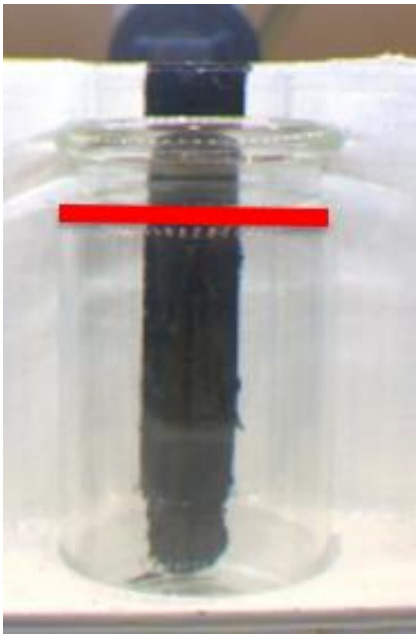


Figura 4.7 Segmentación de cada botella

4.1.5 Manipulación de matriz de pixeles

Tal como se muestra en la figura 4.8 el área de color verde muestra la zona de análisis para determinar si la botella se encontraba con o sin tapa, y el área de color rojo es la zona para determinar si la botella se encontraba con o sin líquido.

El six-pack contenedor de las botellas cuenta en su estructura con tres franjas negras de cada lado tal como se muestra en la figura 4.9, la franja se hizo más ancha con la presencia de líquido, lo cual ayudó para fijar el área de color rojo explicado en el párrafo anterior.

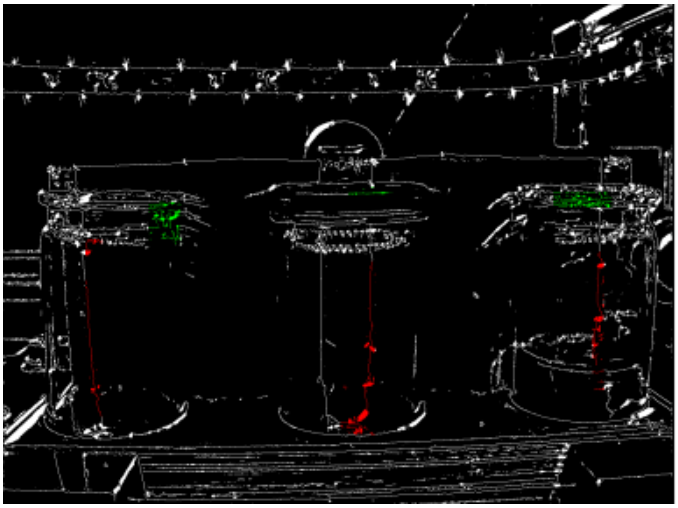


Figura 4.8 Conteo de pixeles para cada estado

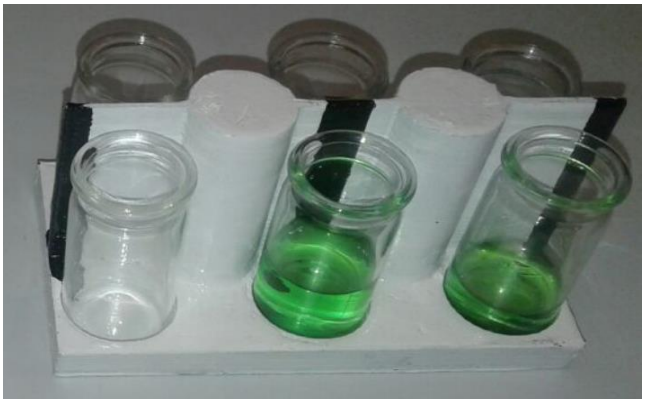


Figura 4.9 Six-pack

4.1.6 Extracción de características

Una vez definidas las matrices a recorrer para comprobar si se encuentra con o sin tapa y con o sin líquido, se obtuvo el rango de pixeles para las condiciones sin errores en cada botella siendo los resultados obtenidos los expuestos a continuación:

Tapa botella 1, 3, 4 y 6:	**>550**
Sin líquido botella1, 3, 4 y 6:	**<700**
Tapa botella 2 y 5:	**>550**
Sin líquido botella2 y 5:	**<2000**

Para obtener resultados 100% efectivos se realizó más de 100 pruebas, en las cuales se modificó principalmente la iluminación del área de trabajo, en la figura 4.10 se muestra la zona de trabajo antes y después de la modificación mencionada.

Antes **Después**

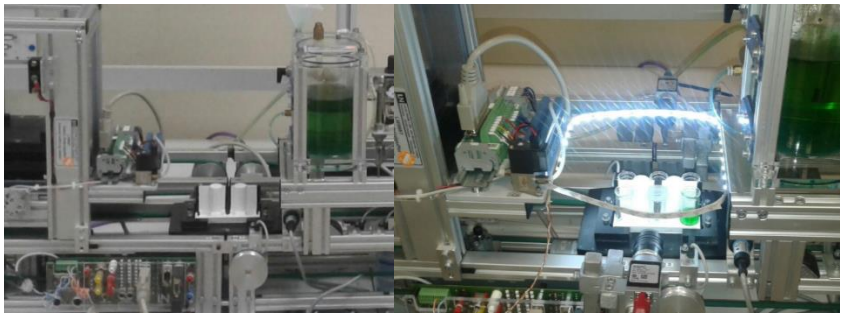


Figura 4.10 Modificación de iluminación en área de trabajo

4.1.7 Uso de datos adquiridos

En la figura 4.11 la imagen de la izquierda muestra a la botella 1 sin errores, la botella 2 se encuentra con tapa y la botella 3 con líquido, por lo cual se escribió desde el código de C# hacia el PLC el valor de TRUE en las direcciones DB5.DBX1.3, DB5.DBX2.2 correspondientes a los errores de las dos botellas mencionadas.

De la misma manera la imagen de la derecha muestra a la botella 4 y 6 sin errores, la botella 5 con tapa y líquido, por lo cual se escribió desde el sistema de Visual Studio hacia el PLC el valor de TRUE en la dirección DB5.DBX3.2 correspondientes a los errores en la botella 5.

Debido a los errores encontrados en el sistema, se escribió en las direcciones DB5.DBX0.2 y DB5.DBX03 del PLC el valor de TRUE, correspondientes al indicador de error en por lo menos una botella y análisis de las botellas ha culminado con éxito.

Los resultados obtenidos para cada botella fueron los esperados, mediante la determinación correcta del área de análisis y el correcto

conteo de pixeles, se determinó cuál botella contenía algún tipo de error y además se indicó cuál era su error, para posteriormente corregirlo.

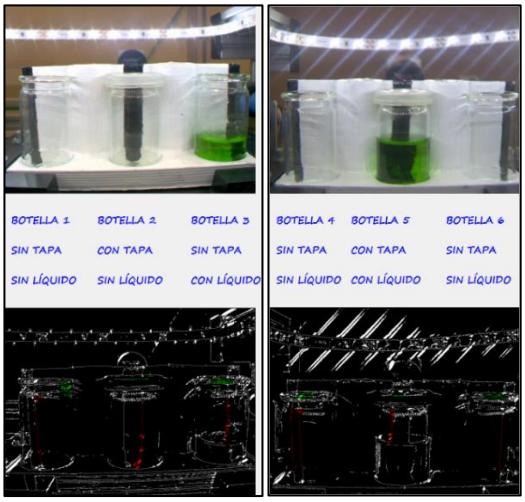


Figura 4.11 Resultados con errores mostrados en Visual Studio

Una vez corregidos los errores se volvió a realizar el análisis de las botellas para verificar que fueron corregidos, y el sistema pudo continuar con el llenado del líquido para cada botella, en la figura 4.13 muestra los resultados del análisis del sistema corregidos los errores encontrados.

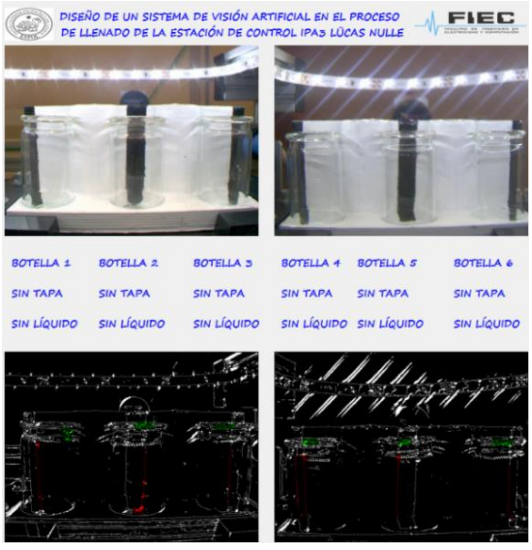


Figura 4.13 Resultados sin errores mostrados en Visual Studio

4.2 Funcionamiento del sistema Scada

4.2.1 Conteo de botellas

En la figura 4.14 el led indicador de “falta al menos una botella” se encuentra apagado lo cual indicó que de las seis botellas que deben ingresar no faltó alguna, permitiendo que el sistema de visión artificial empiece a trabajar para analizar cada botella.

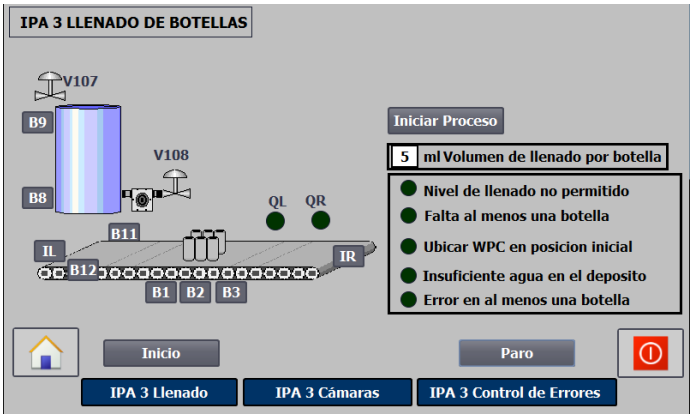


Figura 4.14 Pantalla de visualización de estados de los sensores

4.2.2 Revisión de errores en las botellas

En la figura 4.15 se muestran los errores encontrados en cada botella, correspondientes a las señales enviadas desde el software Visual Studio en lenguaje C#, después de ser analizadas por el sistema.

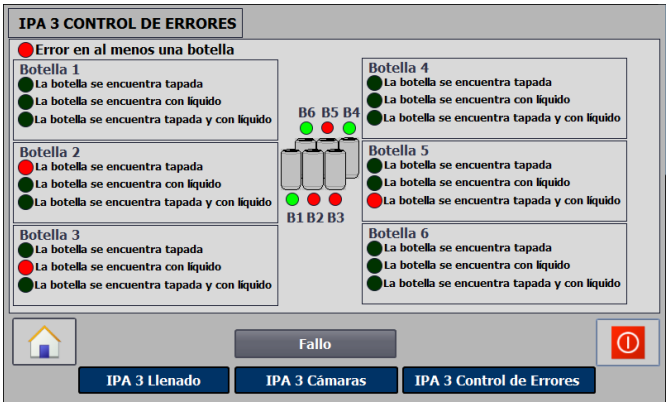


Figura 4.15 Pantalla de visualización de estados de botellas con errores

Al momento en que se mostraron los errores, las botellas son regresadas hacia un punto en el cual pueden ser manipuladas para la corrección manual, para ello el PLC envió la señal de retroceder a la cinta transportadora en la cual se movilizan las botellas hasta llegar al sensor IL, como lo indica en la figura 4.16.

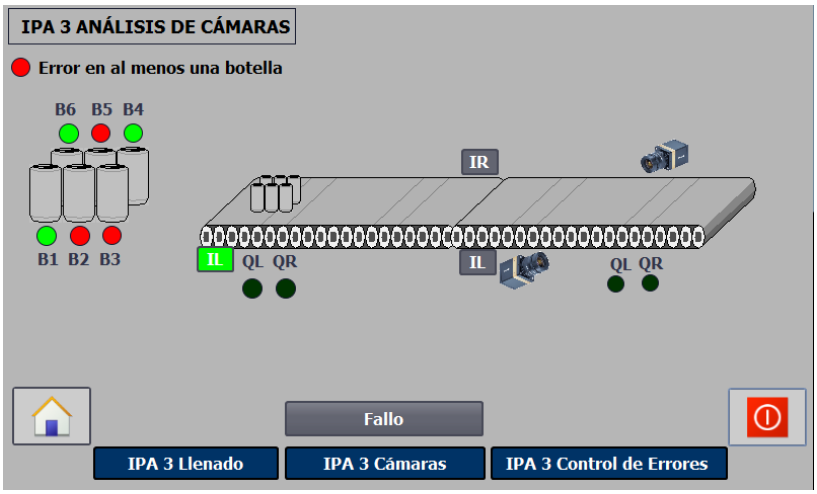


Figura 4.16 Botellas retrocediendo debido a los errores encontrados

Realizado esto se presionó en Fallo para que el sistema vuelva a contar las botellas y verifique una vez más si se encuentra algún error.

En la figura 4.17 muestra que el sistema ya no encontró ningún error por lo cual en el HMI no se encendió ningún indicador de error en las botellas, y en la figura 4.18 muestra que el proceso de llenado continuó hasta llenar las seis botellas y avanzó hasta llegar al sensor IR indicando que el proceso terminó exitosamente.

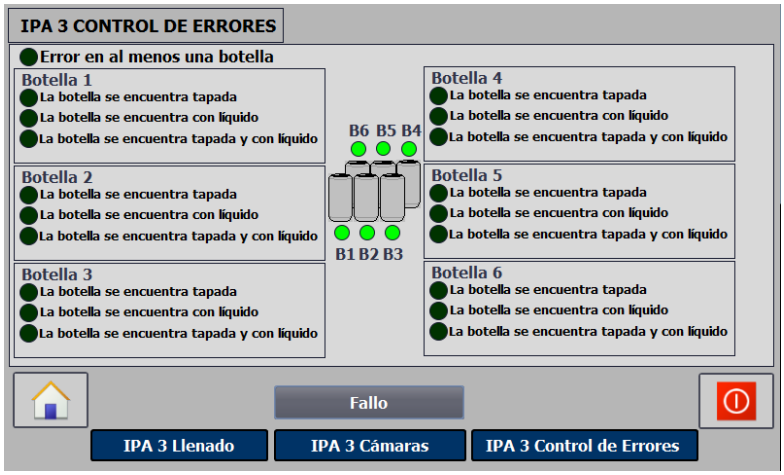


Figura 4.17 Pantalla de visualización de estados de botellas sin errores

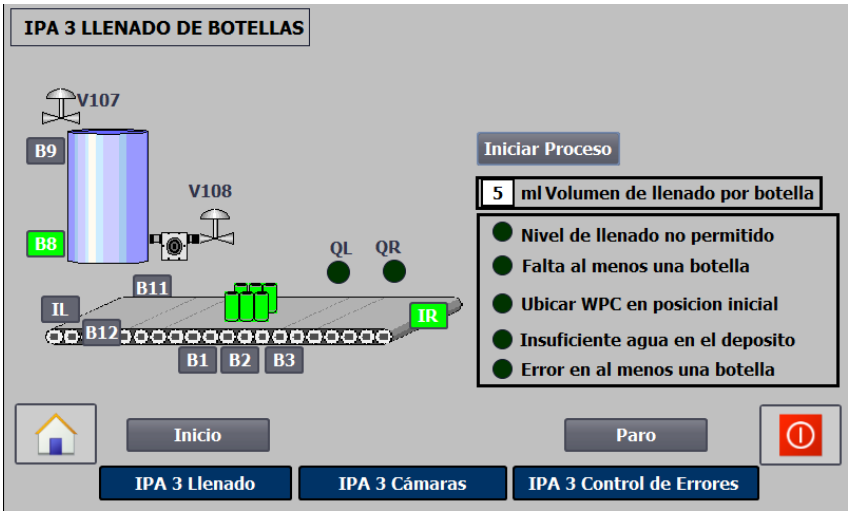


Figura 4.18 Proceso de llenado finalizado

CONCLUSIONES Y RECOMENDACIONES

Se determinaron tres errores existentes en el proceso de llenado de la estación de control IPA3, lo cual genera considerables pérdidas económicas para una industria, el primer error encontrado fue que las botellas ingresan con líquido, el segundo error cuando ingresan tapadas y el tercer error cuando ingresaban con líquido y tapadas. Al integrar el sistema de visión artificial se elimina la necesidad del control humano para dicha detección, aumentando la efectividad del sistema, optimizando el tiempo de detección de errores y eliminando las pérdidas del material, esto fue posible con la utilización de dos cámaras industriales y la creación de la aplicación para el funcionamiento del automatizado.

Al reconstruir la programación en el PLC se logra establecer la correcta comunicación entre las variables de este con el sistema de visión artificial en lenguaje C# en el software Visual Studio 2017, lo cual permitió una eficaz detección de errores y realizar las acciones correspondientes en tiempo real, esto es posible al asociar el sistema de comunicación TCP/IP al proyecto, ya que permite el enlace entre los sistemas de visión y de control (PLC), debido a que el tiempo de respuesta obtenido en la implementación es instantáneo se concluye que el método empleado para la comunicación es óptimo, estable y no cambiará con el paso del tiempo.

Los sistemas de visión artificial en conjunto con otros elementos que complementen su labor, como un brazo robótico, Programadores autónomos entre otros, son de extrema ayuda para las industrias que buscan optimizar recurso, y que tengan una visión a largo plazo, debido a que la inversión para estos sistemas dependiendo de cuál será su aplicación tendrán costos considerables los cuales podrían ser considerados como gastos, pero mediante un análisis se puede demostrar que las ganancias aumentarán considerablemente, ya que las pérdidas económicas se reducirán en un porcentaje considerable.

Este proyecto en comparación con otras soluciones posibles para la eliminación de errores, en cuanto al tiempo de vida útil de los elementos es mayor y las posibilidades para implementar cambios en su sistema, tales como modificar el área

de trabajo para el análisis, modificar la posición de las cámaras dependiendo del espacio disponible son más factibles y fáciles de hacerlo, por lo que queda demostrado que ésta solución implementada va conforme al avance tecnológico que en la actualidad se está dando.

Para el correcto funcionamiento del sistema se recomienda mejorar la iluminación del área en el cual actúan las cámaras, debido a que las variaciones de luminosidad pueden provocar inconvenientes en el procesamiento de las imágenes ya que este proceso se lo realiza en tiempo real, se recomienda también realizar mínimo 100 pruebas para encontrar los diferentes factores que pueden afectar al sistema y poder eliminarlos para su correcta implementación.

Al crear vínculos para las variables de los sistemas de control y visión, se recomienda establecer nombres de acuerdo con lo que se desee realizar, ya que estos sistemas quedaran implementados, pero en caso de mantenimiento o modificaciones en el sistema, al programador se le hará más fácil entender la programación y hacer uso de las variables para evitar conflictos y errores.

BIBLIOGRAFÍA

[1] Mipro, «Ministerio de Industrias y Productividad,» [En línea] Available: http://www.industrias.gob.ec/empresas-de-tecnologia-y-servicios-empresariales-optimizaran-oferta-de-productos-a-la-industria-nacional/ [Último acceso: 11 Enero 2018]

[2] Guía técnica ainia de envase y embalaje, «Guía técnica por sectores,» [En línea]. Available:http://www.guiaenvase.com/bases/guiaenvase.nsf/V02wp/55C1539B41E9E38BC1256F250063FA82?Opendocument

[3] Gestion, «El impacto de la tecnología en la empresa,» [En línea] Available: https://www.gestion.org/gestion-tecnologica/nuevas-tecnologias/29672/el-impacto-de-la-tecnologia-en-la-empresa/ [Último acceso: 11 Enero 2018]

[4] Unal, «Universidad Nacional de Colombia,» [En línea] Available: https://minas.medellin.unal.edu.co/noticias/facultad/775-proponen-sistema-de-control-de-calidad-a-la-industria-textil-mediante-vision-artificial [Último acceso: 11 Enero 2018]

[5] INFAIMON, «Su solución en visión artificial,» [En línea] Available: https://blog.infaimon.com/ford-utiliza-sistema-vision-artificial-detectar-defectos-pintura-coches/ [Último acceso: 14 Enero 2018]

[6] INFAIMON, «Utilización de cámara uEye de IDS para dice;ar calzado a medida,» [En línea] Available: https://blog.infaimon.com/utilizacion-de-camaras-ueye-de-ids-para-disenar-calzado-a-medida/ [Último acceso: 17 Enero 2018]

[7] COGNEX, «Aplicaciones de visión Artificial,» [En línea] Available: https://www.cognex.com/what-is/machine-vision/applications/inspection?langtype=1034&locale=mx [Último acceso: 17 Enero 2018]

[8] AIS, «Sistemas de visión Artificial,» [En línea] Available: http://www.aisvision.com/es/casos-de-estudio/moderngrag/ [Último acceso: 17 Enero 2018]

[9] Prof. Dr. N. Becker/Dipl. Ing (FH) M.Eggeling, Manual IPA3: Estación de envasado

[10] COGNEX, «Principios básicos de visión artificial, Beneficio de la visión artificial,» [En línea]. Available: http://www.cognex.com/what-is/machine-vision/whats-is-machine-vision/?lantype=1034&locale=mx [Último acceso :1 Noviembre 2107].

[11] ETITUDELA, «Visión artificial, conceptos generales,» [En línea] Available: http://www.etitudela.com/celula/downloads/visionartificial.pdf [Último acceso: 1 Noviembre 2017]

[12] INFOPLC, «Sistemas de visión artificial,» [En línea] Available: http://www.infoplc.net/files/documentacion/vision_artificial/infoplc_net_tema_1_va_bn_vision_.pdf [Último acceso: 1 Noviembre 2017]

[13] Basler AG, «Basler AG – Industrial Camera Manufacturer,» [En línea]. Available: https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1300-30gc/ [Último acceso: 1 Noviembre 2017]

[14] Basler AG, «Basler AG – Industrial Camera Manufacturer,» [En línea]. Available: https://www.baslerweb.com/en/products/vision-components/lenses/basler-lens-c125-0818-5m-f1-8-f8mm/ [Último acceso: 6 Noviembre 2017]

[15] AForge, «AForge.NetFramework,» [En línea]. Available: http://www.aforgenet.com/framework/ [Último acceso: 20 Diciembre 2017]

[16] AForge, «CannyEdgedetector class,» [En línea] Available: http://www.aforgenet.com/framework/features/edge_detectors_filters.html [Último acceso: 20 Diciembre 2017]

[17] ALOJAMIENTOS, «Segmentación de imágenes,» [En línea] Available: http://alojamientos.us.es/gtocoma/pid/tema4.pdf [Último acceso: 6 Noviembre 2017]

[18] ES, «Electricidad Serrano,» [En línea] Available: http://electricidadserrano.com.ar/marcas/siemens/plc-simatic-s7-300-siemens/ [Último acceso: 9 Noviembre 2017]

[19] SIEMENS, «Totally Integrated Automation Portal,» [En línea] Available: http://w5.siemens.com/spain/web/es/industry/automatizacion/simatic/tia-portal/pages/tiaportal.aspx [Último acceso:10 Noviembre 2017]

[20] Basler AG, «Basler AG - Industrial Camera Manufacter,» [En línea]. Available: http://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1300-75gc. [Último acceso: 19 Enero 2018].

ANEXOS

Programación en C#

A1. Imagen con Primer Filtro

```
float TH, TL, Sigma;
int MaskSize;

TH = (float)Convert.ToDouble(TxtTH.Text);              //gradiente máximo
TL = (float)Convert.ToDouble(TxtTL.Text);             //gradiente mínimo

MaskSize = Convert.ToInt32(TxtGMask.Text);            //valor de filtrado
Sigma = (float)Convert.ToDouble(TxtSigma.Text);       //constante de filtrado

CannyData = new Canny((Bitmap)pictureBox1.Image, TH, TL, MaskSize, Sigma);
//selección de imagen a ser procesada
pictureBox2.Image = CannyData.DisplayImage(CannyData.FilteredImage);
//mostrar imagen en el espacio del picturebox2
```

A2. Código de Canny

```
float TH, TL, Sigma;
int MaskSize;

TH = (float)Convert.ToDouble(TxtTH.Text);              //gradiente máximo
TL = (float)Convert.ToDouble(TxtTL.Text);             //gradiente mínimo

MaskSize = Convert.ToInt32(TxtGMask.Text);            //valor de filtrado
Sigma = (float)Convert.ToDouble(TxtSigma.Text);       //constante de filtrado

CannyData = new Canny((Bitmap)pictureBox1.Image, TH, TL, MaskSize, Sigma);
//selección de imagen a ser procesada
pictureBox2.Image = CannyData.DisplayImage(CannyData.GNH);
//mostrar imagen en el espacio del picturebox2


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
using System.Drawing.Imaging;

using System.Threading;
```

```
namespace S7_net_example
{
class Canny
{

//  static TimeSpan waitTime2 = new TimeSpan(0, 0, 2);

public int Width, Height;
public Bitmap Obj;
public int[,] GreyImage;
//Gaussian Kernel Data
int[,] GaussianKernel;
int KernelWeight;
int KernelSize = 5;
float Sigma = 1;   // for N=2 Sigma =0.85  N=5 Sigma =1, N=9 Sigma = 2    2*Sigma =
(int)N/2
//Canny Edge Detection Parameters
float MaxHysteresisThresh, MinHysteresisThresh;
public float[,] DerivativeX;
public float[,] DerivativeY;
public int[,] FilteredImage;
public float[,] Gradient;
public float[,] NonMax;
public int[,] PostHysteresis;
int[,] EdgePoints;
public float[,] GNH;
public float[,] GNL;
public int[,] EdgeMap;
public int[,] VisitedMap;

public Canny(Bitmap Input)
{
   // Gaussian and Canny Parameters
   MaxHysteresisThresh = 20F;
   MinHysteresisThresh = 10F;
   Obj = Input;
   Width = Obj.Width;
   Height = Obj.Height;
   EdgeMap = new int[Width, Height];
   VisitedMap = new int[Width, Height];

   ReadImage();
   DetectCannyEdges();
   return;
}

public Canny(Bitmap Input, float Th, float Tl)
{
```

```
    // Gaussian and Canny Parameters

    MaxHysteresisThresh = Th;
    MinHysteresisThresh = Tl;

    Obj = Input;
    Width = Obj.Width;
    Height = Obj.Height;

    EdgeMap = new int[Width, Height];
    VisitedMap = new int[Width, Height];

    ReadImage();
    DetectCannyEdges();
    return;
}

public Canny(Bitmap Input, float Th, float Tl, int GaussianMaskSize, float
SigmaforGaussianKernel)
{

    // Gaussian and Canny Parameters

    MaxHysteresisThresh = Th;
    MinHysteresisThresh = Tl;
    KernelSize = GaussianMaskSize;
    Sigma = SigmaforGaussianKernel;
    Obj = Input;

    Width = Obj.Width;
    Height = Obj.Height;

    EdgeMap = new int[Width, Height];
    VisitedMap = new int[Width, Height];

    ReadImage();
    DetectCannyEdges();
    return;
}

public Bitmap DisplayImage()
{
    int i, j;
    Bitmap image = new Bitmap(Obj.Width, Obj.Height);
    BitmapData bitmapData1 = image.LockBits(new Rectangle(0, 0, Obj.Width,
Obj.Height),
                        ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb);
```

```
    unsafe
    {
        byte* imagePointer1 = (byte*)bitmapData1.Scan0;

        for (i = 0; i < bitmapData1.Height; i++)
        {
            for (j = 0; j < bitmapData1.Width; j++)
            {
                // write the logic implementation here
                imagePointer1[0] = (byte)GreyImage[j, i];
                imagePointer1[1] = (byte)GreyImage[j, i];
                imagePointer1[2] = (byte)GreyImage[j, i];
                imagePointer1[3] = (byte)255;
                //4 bytes per pixel
                imagePointer1 += 4;
            }//end for j

            //4 bytes per pixel
            imagePointer1 += (bitmapData1.Stride - (bitmapData1.Width * 4));
        }//end for i
    }//end unsafe

    image.UnlockBits(bitmapData1);
    return image;// col;
}     // Display Grey Image
/*
public Bitmap DisplayImage(float[,] GreyImage)
{
    int i, j;
    int W, H;
    W = GreyImage.GetLength(0);
    H = GreyImage.GetLength(1);
    Bitmap image = new Bitmap(W, H);
    BitmapData bitmapData1 = image.LockBits(new Rectangle(0, 0, W, H),
                             ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb);
    unsafe
    {

        byte* imagePointer1 = (byte*)bitmapData1.Scan0;

        for (i = 0; i < bitmapData1.Height; i++)
        {
            for (j = 0; j < bitmapData1.Width; j++)
            {
                // write the logic implementation here
                imagePointer1[0] = (byte)((int)(GreyImage[j, i]));
                imagePointer1[1] = (byte)((int)(GreyImage[j, i]));
```

```
                imagePointer1[2] = (byte)((int)(GreyImage[j, i]));
                imagePointer1[3] = (byte)255;
                //4 bytes per pixel
                imagePointer1 += 4;
            }   //end for j
            //4 bytes per pixel
            imagePointer1 += (bitmapData1.Stride - (bitmapData1.Width * 4));
        }//End for i
    }//end unsafe

    image.UnlockBits(bitmapData1);
    return image;// col;
}     // Display Grey Imag
*/
public Bitmap DisplayImage(int[,] GreyImage)
{
    int i, j;
    int W, H;
    W = GreyImage.GetLength(0);
    H = GreyImage.GetLength(1);
    Bitmap image = new Bitmap(W, H);
    BitmapData bitmapData1 = image.LockBits(new Rectangle(0, 0, W, H),
                             ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb);
    unsafe
    {

        byte* imagePointer1 = (byte*)bitmapData1.Scan0;

        for (i = 0; i < bitmapData1.Height; i++)
        {
            for (j = 0; j < bitmapData1.Width; j++)
            {
                // write the logic implementation here
                imagePointer1[0] = (byte)GreyImage[j, i];
                imagePointer1[1] = (byte)GreyImage[j, i];
                imagePointer1[2] = (byte)GreyImage[j, i];
                imagePointer1[3] = (byte)255;
                //4 bytes per pixel
                imagePointer1 += 4;
            }   //end for j
            //4 bytes per pixel
            imagePointer1 += (bitmapData1.Stride - (bitmapData1.Width * 4));
        }//End for i
    }//end unsafe

    image.UnlockBits(bitmapData1);
    return image;// col;
}     // Display Grey Image
```

```
private void ReadImage()
{
    int i, j;
    GreyImage = new int[Obj.Width, Obj.Height];  //[Row,Column]
    Bitmap image = Obj;
    BitmapData bitmapData1 = image.LockBits(new Rectangle(0, 0, image.Width,
image.Height),
                        ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb);
    unsafe
    {
        byte* imagePointer1 = (byte*)bitmapData1.Scan0;

        for (i = 0; i < bitmapData1.Height; i++)
        {
            for (j = 0; j < bitmapData1.Width; j++)
            {
                GreyImage[j, i] = (int)((imagePointer1[0] + imagePointer1[1] +
imagePointer1[2]) / 3.0);
                //4 bytes per pixel
                imagePointer1 += 4;
            }//end for j
            //4 bytes per pixel
            imagePointer1 += bitmapData1.Stride - (bitmapData1.Width * 4);
        }//end for i
    }//end unsafe
    image.UnlockBits(bitmapData1);
    return;
}

private void GenerateGaussianKernel(int N, float S, out int Weight)
{

    float Sigma = S;
    float pi;
    pi = (float)Math.PI;
    int i, j;
    int SizeofKernel = N;

    float[,] Kernel = new float[N, N];
    GaussianKernel = new int[N, N];
    float[,] OP = new float[N, N];
    float D1, D2;


    D1 = 1 / (2 * pi * Sigma * Sigma);
    D2 = 2 * Sigma * Sigma;
```

```
    float min = 1000;

    for (i = -SizeofKernel / 2; i <= SizeofKernel / 2; i++)
    {
        for (j = -SizeofKernel / 2; j <= SizeofKernel / 2; j++)
        {
            Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] = ((1 / D1) * (float)Math.Exp(-
(i * i + j * j) / D2));
            if (Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] < min)
                min = Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j];

        }
    }
    int mult = (int)(1 / min);
    int sum = 0;
    if ((min > 0) && (min < 1))
    {

        for (i = -SizeofKernel / 2; i <= SizeofKernel / 2; i++)
        {
            for (j = -SizeofKernel / 2; j <= SizeofKernel / 2; j++)
            {
                Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] =
(float)Math.Round(Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] * mult, 0);
                GaussianKernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] =
(int)Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j];
                sum = sum + GaussianKernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j];
            }

        }

    }
    else
    {
        sum = 0;
        for (i = -SizeofKernel / 2; i <= SizeofKernel / 2; i++)
        {
            for (j = -SizeofKernel / 2; j <= SizeofKernel / 2; j++)
            {
                Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] =
(float)Math.Round(Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j], 0);
                GaussianKernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] =
(int)Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j];
                sum = sum + GaussianKernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j];
            }

        }
```

```
    }
    //Normalizing kernel Weight
    Weight = sum;

    return;
}

private int[,] GaussianFilter(int[,] Data)
{
    GenerateGaussianKernel(KernelSize, Sigma, out KernelWeight);

    int[,] Output = new int[Width, Height];
    int i, j, k, l;
    int Limit = KernelSize / 2;

    float Sum = 0;


    Output = Data; // Removes Unwanted Data Omission due to kernel bias while convolution


    for (i = Limit; i <= ((Width - 1) - Limit); i++)
    {
        for (j = Limit; j <= ((Height - 1) - Limit); j++)
        {
            Sum = 0;
            for (k = -Limit; k <= Limit; k++)
            {

                for (l = -Limit; l <= Limit; l++)
                {
                    Sum = Sum + ((float)Data[i + k, j + l] * GaussianKernel[Limit + k, Limit + l]);

                }
            }
            Output[i, j] = (int)(Math.Round(Sum / (float)KernelWeight));
        }

    }


    return Output;
}

private float[,] Differentiate(int[,] Data, int[,] Filter)
{
```

```
    int i, j, k, l, Fh, Fw;

    Fw = Filter.GetLength(0);
    Fh = Filter.GetLength(1);
    float sum = 0;
    float[,] Output = new float[Width, Height];

    for (i = Fw / 2; i <= (Width - Fw / 2) - 1; i++)
    {
        for (j = Fh / 2; j <= (Height - Fh / 2) - 1; j++)
        {
            sum = 0;
            for (k = -Fw / 2; k <= Fw / 2; k++)
            {
                for (l = -Fh / 2; l <= Fh / 2; l++)
                {
                    sum = sum + Data[i + k, j + l] * Filter[Fw / 2 + k, Fh / 2 + l];


                }
            }
            Output[i, j] = sum;

        }

    }
    return Output;

}

private void DetectCannyEdges()
{
    Gradient = new float[Width, Height];
    NonMax = new float[Width, Height];
    PostHysteresis = new int[Width, Height];

    DerivativeX = new float[Width, Height];
    DerivativeY = new float[Width, Height];

    //Gaussian Filter Input Image

    FilteredImage = GaussianFilter(GreyImage);
    //Sobel Masks
    int[,] Dx = {{1,0,-1},
                 {1,0,-1},
                 {1,0,-1}};

    int[,] Dy = {{1,1,1},
```

```
                {0,0,0},
                {-1,-1,-1}};


    DerivativeX = Differentiate(FilteredImage, Dx);
    DerivativeY = Differentiate(FilteredImage, Dy);

    int i, j;

    //Compute the gradient magnitude based on derivatives in x and y:
    for (i = 0; i <= (Width - 1); i++)
    {
        for (j = 0; j <= (Height - 1); j++)
        {
            Gradient[i, j] = (float)Math.Sqrt((DerivativeX[i, j] * DerivativeX[i, j]) +
(DerivativeY[i, j] * DerivativeY[i, j]));

        }

    }
    // Perform Non maximum suppression:
    // NonMax = Gradient;

    for (i = 0; i <= (Width - 1); i++)
    {
        for (j = 0; j <= (Height - 1); j++)
        {
            NonMax[i, j] = Gradient[i, j];
        }
    }

    int Limit = KernelSize / 2;
    int r, c;
    float Tangent;


    for (i = Limit; i <= (Width - Limit) - 1; i++)
    {
        for (j = Limit; j <= (Height - Limit) - 1; j++)
        {

            if (DerivativeX[i, j] == 0)
                Tangent = 90F;
            else
                Tangent = (float)(Math.Atan(DerivativeY[i, j] / DerivativeX[i, j]) * 180 /
Math.PI); //rad to degree
```

```
            //Horizontal Edge
            if (((-22.5 < Tangent) && (Tangent <= 22.5)) || ((157.5 < Tangent) && (Tangent
<= -157.5)))
            {
                if ((Gradient[i, j] < Gradient[i, j + 1]) || (Gradient[i, j] < Gradient[i, j - 1]))
                    NonMax[i, j] = 0;
            }


            //Vertical Edge
            if (((-112.5 < Tangent) && (Tangent <= -67.5)) || ((67.5 < Tangent) && (Tangent
<= 112.5)))
            {
                if ((Gradient[i, j] < Gradient[i + 1, j]) || (Gradient[i, j] < Gradient[i - 1, j]))
                    NonMax[i, j] = 0;
            }

            //+45 Degree Edge
            if (((-67.5 < Tangent) && (Tangent <= -22.5)) || ((112.5 < Tangent) && (Tangent
<= 157.5)))
            {
                if ((Gradient[i, j] < Gradient[i + 1, j - 1]) || (Gradient[i, j] < Gradient[i - 1, j +
1]))
                    NonMax[i, j] = 0;
            }

            //-45 Degree Edge
            if (((-157.5 < Tangent) && (Tangent <= -112.5)) || ((67.5 < Tangent) &&
(Tangent <= 22.5)))
            {
                if ((Gradient[i, j] < Gradient[i + 1, j + 1]) || (Gradient[i, j] < Gradient[i - 1, j -
1]))
                    NonMax[i, j] = 0;
            }

        }
    }


    //PostHysteresis = NonMax;
    for (r = Limit; r <= (Width - Limit) - 1; r++)
    {
        for (c = Limit; c <= (Height - Limit) - 1; c++)
        {

            PostHysteresis[r, c] = (int)NonMax[r, c];
        }
```

```
    }

    //Find Max and Min in Post Hysterisis
    float min, max;
    min = 100;
    max = 0;
    for (r = Limit; r <= (Width - Limit) - 1; r++)
        for (c = Limit; c <= (Height - Limit) - 1; c++)
        {
            if (PostHysteresis[r, c] > max)
            {
                max = PostHysteresis[r, c];
            }

            if ((PostHysteresis[r, c] < min) && (PostHysteresis[r, c] > 0))
            {
                min = PostHysteresis[r, c];
            }
        }

    GNH = new float[Width, Height];
    GNL = new float[Width, Height]; ;
    EdgePoints = new int[Width, Height];

    for (r = Limit; r <= (Width - Limit) - 1; r++)
    {
        for (c = Limit; c <= (Height - Limit) - 1; c++)
        {
            if (PostHysteresis[r, c] >= MaxHysteresisThresh)
            {

                EdgePoints[r, c] = 1;
                GNH[r, c] = 255;
            }
            if ((PostHysteresis[r, c] < MaxHysteresisThresh) && (PostHysteresis[r, c] >=
MinHysteresisThresh))
            {

                EdgePoints[r, c] = 2;
                GNL[r, c] = 255;

            }

        }

    }
```

```
    HysterisisThresholding(EdgePoints);

    for (i = 0; i <= (Width - 1); i++)
        for (j = 0; j <= (Height - 1); j++)
        {
            EdgeMap[i, j] = EdgeMap[i, j] * 255;
        }

    return;

}

private void HysterisisThresholding(int[,] Edges)
{

    int i, j;
    int Limit = KernelSize / 2;


    for (i = Limit; i <= (Width - 1) - Limit; i++)
        for (j = Limit; j <= (Height - 1) - Limit; j++)
        {
            if (Edges[i, j] == 1)
            {
                EdgeMap[i, j] = 1;

            }

        }

    for (i = Limit; i <= (Width - 1) - Limit; i++)
    {
        for (j = Limit; j <= (Height - 1) - Limit; j++)
        {
            if (Edges[i, j] == 1)
            {
                EdgeMap[i, j] = 1;
                Travers(i, j);
                VisitedMap[i, j] = 1;
            }
        }
    }



    return;
}
```

```
private void Travers(int X, int Y)
{


    if (VisitedMap[X, Y] == 1)
    {
        return;
    }

    //1
    if (EdgePoints[X + 1, Y] == 2)
    {
        EdgeMap[X + 1, Y] = 1;
        VisitedMap[X + 1, Y] = 1;
        Travers(X + 1, Y);
        return;
    }
    //2
    if (EdgePoints[X + 1, Y - 1] == 2)
    {
        EdgeMap[X + 1, Y - 1] = 1;
        VisitedMap[X + 1, Y - 1] = 1;
        Travers(X + 1, Y - 1);
        return;
    }

    //3

    if (EdgePoints[X, Y - 1] == 2)
    {
        EdgeMap[X, Y - 1] = 1;
        VisitedMap[X, Y - 1] = 1;
        Travers(X, Y - 1);
        return;
    }

    //4

    if (EdgePoints[X - 1, Y - 1] == 2)
    {
        EdgeMap[X - 1, Y - 1] = 1;
        VisitedMap[X - 1, Y - 1] = 1;
        Travers(X - 1, Y - 1);
        return;
    }
    //5
    if (EdgePoints[X - 1, Y] == 2)
```

```
    {
        EdgeMap[X - 1, Y] = 1;
        VisitedMap[X - 1, Y] = 1;
        Travers(X - 1, Y);
        return;
    }
    //6
    if (EdgePoints[X - 1, Y + 1] == 2)
    {
        EdgeMap[X - 1, Y + 1] = 1;
        VisitedMap[X - 1, Y + 1] = 1;
        Travers(X - 1, Y + 1);
        return;
    }
    //7
    if (EdgePoints[X, Y + 1] == 2)
    {
        EdgeMap[X, Y + 1] = 1;
        VisitedMap[X, Y + 1] = 1;
        Travers(X, Y + 1);
        return;
    }
    //8

    if (EdgePoints[X + 1, Y + 1] == 2)
    {
        EdgeMap[X + 1, Y + 1] = 1;
        VisitedMap[X + 1, Y + 1] = 1;
        Travers(X + 1, Y + 1);
        return;
    }


    //VisitedMap[X, Y] = 1;
    return;
}

//Canny Class Ends

}
}
```

A3. Segmentación de Imagen

```
//---------------conteo de pixeles-------------------------//
//-------------------------botella-----------------------//

//----------------tapa--------------------//
for (int i = 105; i <= 270; i++)
{
   for (int j = 500; j <= 530; j++)
   {
      int a = b.GetPixel(i, j).ToArgb();
      if (a.Equals(-1))
      {
         tapa = tapa + 1;
         b.SetPixel(i, j, cambio1);
      }
   }
}
//----------------tapa--------------------//

for (int i = 110; i <= 270; i++)
{
   for (int j = 625; j <= 950; j++)
   {
      int a = b.GetPixel(i, j).ToArgb();

      if (a.Equals(-1))

      {
         blancos = blancos + 1;
         b.SetPixel(i, j, cambio);
      }
   }
}
//-------------------------botella fin-----------------------------------------//


//-----------------------------botella 1-----------------------------------//

//----------------tapa1--------------------//
for (int i = 560; i <= 740; i++)
   {
      for (int j = 500; j <= 530; j++)
      {
         int a = b.GetPixel(i, j).ToArgb();
         if (a.Equals(-1))
         {
            tapa1 = tapa1 + 1;
```

```
                b.SetPixel(i, j, cambio1);
            }
        }
    }
//----------------tapa1--------------------//

for (int i = 565; i <= 700; i++)
{
    for (int j = 630; j <= 930; j++)
    {
        int a = b.GetPixel(i, j).ToArgb();
        if (a.Equals(-1))

        {
            blancos1 = blancos1 + 1;
            b.SetPixel(i, j, cambio);
        }
    }
}
//--------------------------------botella 1 fin-------------------------------//


//-------------------------botella 2---------------------------------------//

//----------------tapa2--------------------//
for (int i = 1008; i <= 1150; i++)
    {
        for (int j = 485; j <= 530; j++)
        {
            int a = b.GetPixel(i, j).ToArgb();
            if (a.Equals(-1))
            {
                tapa2 = tapa2 + 1;
                b.SetPixel(i, j, cambio1);
            }
        }
    }
//----------------tapa2--------------------//

for (int i = 1023; i <= 1135; i++)
{
    for (int j = 635; j <= 925; j++)
    {
        int a = b.GetPixel(i, j).ToArgb();
        if (a.Equals(-1))
        {
            blancos2 = blancos2 + 1;
            b.SetPixel(i, j, cambio);
```

```
        }
    }
}


blancos4 = 0;
blancos5 = 0;
blancos6 = 0;
tapa4 = 0;
tapa5 = 0;
tapa6 = 0;
bot4 = 0;
bot5 = 0;
bot6 = 0;
//--------------------------botella 2fin----------------------------------------//


//-----------------------botella 4 ----------------------------------------//
//----------------tapa 4--------------------//
for (int i = 110; i <= 305; i++)
{
    for (int j = 410; j <= 440; j++)
    {
        int d = c.GetPixel(i, j).ToArgb();
        if (d.Equals(-1))
        {
            tapa4 = tapa4 + 1;
            c.SetPixel(i, j, cambio1);
        }
    }
}
//----------------tapa 4--------------------//

for (int i = 130; i <= 300; i++)
{
    for (int j = 520; j <= 840; j++)
    {
        int d = c.GetPixel(i, j).ToArgb();

        if (d.Equals(-1))

        {
            blancos4 = blancos4 + 1;
            c.SetPixel(i, j, cambio);
        }
    }
}
```

```
//-------------------------botella 4 fin-----------------------------------//


//----------------------------botella 5-----------------------------------//

//----------------tapa5--------------------//
for (int i = 540; i <= 715; i++)
{
    for (int j = 410; j <= 445; j++)
    {
        int d = c.GetPixel(i, j).ToArgb();
        if (d.Equals(-1))
        {
            tapa5 = tapa5 + 1;
            c.SetPixel(i, j, cambio1);
        }
    }
}
//----------------tapa5--------------------//

for (int i = 520; i <= 707; i++)
{
    for (int j = 515; j <= 840; j++)
    {
        int d = c.GetPixel(i, j).ToArgb();
        if (d.Equals(-1))
        {
            blancos5 = blancos5 + 1;
            c.SetPixel(i, j, cambio);
        }
    }
}
//---------------------------botella 5 fin----------------------------------//


//---------------------------botella 6-----------------------------------//

//----------------tapa6--------------------//
for (int i = 985; i <= 1150; i++)
{
    for (int j = 410; j <= 445; j++)
    {
        int d = c.GetPixel(i, j).ToArgb();
        if (d.Equals(-1))
        {
            tapa6 = tapa6 + 1;
            c.SetPixel(i, j, cambio1);
        }
```

```
    }
}
//----------------tapa6--------------------//

for (int i = 980; i <= 1130; i++)
{
    for (int j = 530; j <= 810; j++)
    {
        int d = c.GetPixel(i, j).ToArgb();
        if (d.Equals(-1))
        {
            blancos6 = blancos6 + 1;
            c.SetPixel(i, j, cambio);
        }
    }
}

//--------------------------botella 6fin----------------------------------//
```

A4. Extracción de Características

```
                //-------------------------------------botella-------------------------------//

    //----------------tapa--------------------//
    for (int i = 105; i <= 270; i++)
    {
        for (int j = 500; j <= 530; j++)
        {
            int a = b.GetPixel(i, j).ToArgb();
            if (a.Equals(-1))
            {
                tapa = tapa + 1;
                b.SetPixel(i, j, cambio1);
            }
        }
    }

    if (tapasup < tapa)
        tapasup = tapa;
    if (tapa != 0)
        if (tapainf > tapa)
        tapainf = tapa;
       //----------------tapa--------------------//

    for (int i = 110; i <= 270; i++)
    {
        for (int j = 625; j <= 950; j++)
```

```
        {
            int a = b.GetPixel(i, j).ToArgb();
            if (a.Equals(-1))
            {
                blancos = blancos + 1;
                b.SetPixel(i, j, cambio);
            }
        }
    }

    if (blanossup < blancos)
        blanossup = blancos;
    if (blancos != 0)
        if (blancosinf > blancos)
        blancosinf = blancos;
    label1.Text = tapasup.ToString();
    label2.Text = tapainf.ToString();
    label3.Text = blanossup.ToString();
    label4.Text = blancosinf.ToString();



//--------------------------------botella fin---------------------------------//


//--------------------------------botella 1---------------------------------//
    //----------------tapa1--------------------//
    for (int i = 560; i <= 740; i++)
    {
        for (int j = 500; j <= 530; j++)
        {
            int a = b.GetPixel(i, j).ToArgb();
            if (a.Equals(-1))
            {
                tapa1 = tapa1 + 1;
                b.SetPixel(i, j, cambio1);
            }
        }
    }
    if (tapasup1 < tapa1)
        tapasup1 = tapa1;
    if (tapa1 != 0)
        if (tapainf1 > tapa1)
        tapainf1 = tapa1;
    //----------------tapa1--------------------//
```

```
    for (int i = 565; i <= 700; i++)
    {
        for (int j = 630; j <= 930; j++)
        {
            int a = b.GetPixel(i, j).ToArgb();
            if (a.Equals(-1))
            {
                blancos1 = blancos1 + 1;
                b.SetPixel(i, j, cambio);
            }
        }
    }


    if (blanossup1 < blancos1)
        blanossup1 = blancos1;
    if (blancos1 != 0)
        if (blancosinf1 > blancos1)
        blancosinf1 = blancos1;
    label5.Text = tapasup1.ToString();
    label6.Text = tapainf1.ToString();
    label7.Text = blanossup1.ToString();
    label8.Text = blancosinf1.ToString();
//-----------------botella 1 fin-----------------------------------------//


//-----------------------------------botella 2-------------------------------//

    //----------------tapa2--------------------//
    for (int i = 1008; i <= 1150; i++)
    {
        for (int j = 485; j <= 530; j++)
        {
            int a = b.GetPixel(i, j).ToArgb();
            if (a.Equals(-1))
            {
                tapa2 = tapa2 + 1;
                b.SetPixel(i, j, cambio1);
            }
        }
    }
    if (tapasup2 < tapa2)
        tapasup2 = tapa2;
    if (tapa2 != 0)
        if (tapainf2 > tapa2)
        tapainf2 = tapa2;
```

```
    //----------------tapa2--------------------//

    for (int i = 1023; i <= 1165; i++)
    {
        for (int j = 635; j <= 925; j++)
        {
            int a = b.GetPixel(i, j).ToArgb();
            if (a.Equals(-1))
            {
                blancos2 = blancos2 + 1;
                b.SetPixel(i, j, cambio);
            }
        }
    }

    if (blanossup2 < blancos2)
        blanossup2 = blancos2;
    if (blancos2 != 0)
        if (blancosinf2 > blancos2)
        blancosinf2 = blancos2;
    label9.Text = tapasup2.ToString();
    label10.Text = tapainf2.ToString();
    label11.Text = blanossup2.ToString();
    label12.Text = blancosinf2.ToString();
//------------------------------botella 2fin---------------------------------//


//----------------------botella4 ------------------------------------------//
    //----------------tapa4--------------------//
    for (int i = 110; i <= 305; i++)
    {
        for (int j = 410; j <= 440; j++)
        {
            int d = c.GetPixel(i, j).ToArgb();
            if (d.Equals(-1))
            {
                tapa4 = tapa4 + 1;
                c.SetPixel(i, j, cambio1);
            }
        }
    }
    if (tapasup4 < tapa4)
        tapasup4 = tapa4;
    if (tapa4 != 0)
        if (tapainf4 > tapa4)
        tapainf4 = tapa4;

    //----------------tapa4--------------------//
```

```
    for (int i = 130; i <= 300; i++)
    {
        for (int j = 520; j <= 840; j++)
        {
            int d = c.GetPixel(i, j).ToArgb();
            if (d.Equals(-1))
            {
                blancos4 = blancos4 + 1;
                c.SetPixel(i, j, cambio);
            }
        }
    }

    if (blanossup4 < blancos4)
        blanossup4 = blancos4;
    if (blancos4 != 0)
        if (blancosinf4 > blancos4)
        blancosinf4 = blancos4;
    label13.Text = tapasup4.ToString();
    label14.Text = tapainf4.ToString();
    label15.Text = blanossup4.ToString();
    label16.Text = blancosinf4.ToString();


//------------------------------botella4  fin--------------------------------//


//-------------------------botella5-----------------------------------------//

    //----------------tapa5--------------------//
    for (int i = 540; i <= 715; i++)
    {
        for (int j = 410; j <= 445; j++)
        {
            int d = c.GetPixel(i, j).ToArgb();
            if (d.Equals(-1))
            {
                tapa5 = tapa5 + 1;
                c.SetPixel(i, j, cambio1);
            }
        }
    }
    if (tapasup5 < tapa5)
        tapasup5 = tapa5;
    if (tapa5 != 0)
        if (tapainf5 > tapa5)
```

```
            tapainf5 = tapa5;
        //----------------tapa5--------------------//

        for (int i = 520; i <= 707; i++)
        {
            for (int j = 515; j <= 840; j++)
            {
                int d = c.GetPixel(i, j).ToArgb();
                // b.SetPixel(i, j, cambio);
                if (d.Equals(-1))
                {
                    blancos5 = blancos5 + 1;
                    c.SetPixel(i, j, cambio);
                }
            }
        }

        if (blanossup5 < blancos5)
            blanossup5 = blancos5;
        if (blancos5 != 0)
            if (blancosinf5 > blancos5)
            blancosinf5 = blancos5;
        label17.Text = tapasup5.ToString();
        label18.Text = tapainf5.ToString();
        label19.Text = blanossup5.ToString();
        label20.Text = blancosinf5.ToString();
    //--------------------------botella5 fin-----------------------------------//



    //------------------------botella6-----------------------------------//

        //----------------tapa6--------------------//
        for (int i = 985; i <= 1150; i++)
        {
            for (int j = 410; j <= 445; j++)
            {
                int d = c.GetPixel(i, j).ToArgb();
                if (d.Equals(-1))
                {
                    tapa6 = tapa6 + 1;
                    c.SetPixel(i, j, cambio1);
                }
            }
        }
        if (tapasup6 < tapa6)
            tapasup6 = tapa6;
```

```
    if (tapa6 != 0)
        if (tapainf6 > tapa6)
        tapainf6 = tapa6;

    //----------------tapa6--------------------//
    for (int i = 980; i <= 1130; i++)
    {
        for (int j = 530; j <= 810; j++)
        {
            int d = c.GetPixel(i, j).ToArgb();
            if (d.Equals(-1))
            {
                blancos6 = blancos6 + 1;
                c.SetPixel(i, j, cambio);
            }
        }
    }

    if (blanossup6 < blancos6)
        blanossup6 = blancos6;
    if (blancos6 != 0)
        if (blancosinf6 > blancos6)
        blancosinf6 = blancos6;
    label21.Text = tapasup6.ToString();
    label22.Text = tapainf6.ToString();
    label23.Text = blanossup6.ToString();
    label24.Text = blancosinf6.ToString();
//----------------------------botella6 fin---------------------------------//
```

A5. Conversión de variables del PLC

```
using System;
using System.Globalization;

namespace S7.Net
{
    /// <summary>
    /// Conversion methods to convert from Siemens numeric format to C# and back
    /// </summary>
    public static class Conversion
    {
        /// <summary>
        /// Converts a binary string to Int32 value
        /// </summary>
        /// <param name="txt"></param>
        /// <returns></returns>
        public static int BinStringToInt32(this string txt)
        {
            int cnt = 0;
            int ret = 0;

            for (cnt = txt.Length - 1; cnt >= 0; cnt += -1)
            {
                if (int.Parse(txt.Substring(cnt, 1)) == 1)
                {
                    ret += (int)(Math.Pow(2, (txt.Length - 1 - cnt)));
                }
            }
            return ret;
        }

        /// <summary>
        /// Converts a binary string to a byte. Can return null.
        /// </summary>
        /// <param name="txt"></param>
        /// <returns></returns>
        public static byte? BinStringToByte(this string txt)
        {
            int cnt = 0;
            int ret = 0;

            if (txt.Length == 8)
            {
                for (cnt = 7; cnt >= 0; cnt += -1)
                {
                    if (int.Parse(txt.Substring(cnt, 1)) == 1)
                    {
```

```
                ret += (int)(Math.Pow(2, (txt.Length - 1 - cnt)));
            }
        }
        return (byte)ret;
    }
    return null;
}

/// <summary>
/// Converts the value to a binary string
/// </summary>
/// <param name="value"></param>
/// <returns></returns>
public static string ValToBinString(this object value)
{
    int cnt = 0;
    int cnt2 = 0;
    int x = 0;
    string txt = "";
    long longValue = 0;

    try
    {
        if (value.GetType().Name.IndexOf("[]") < 0)
        {
            // ist nur ein Wert
            switch (value.GetType().Name)
            {
                case "Byte":
                    x = 7;
                    longValue = (long)((byte)value);
                    break;
                case "Int16":
                    x = 15;
                    longValue = (long)((Int16)value);
                    break;
                case "Int32":
                    x = 31;
                    longValue = (long)((Int32)value);
                    break;
                case "Int64":
                    x = 63;
                    longValue = (long)((Int64)value);
                    break;
                default:
                    throw new Exception();
            }
```

```
                for (cnt = x; cnt >= 0; cnt += -1)
                {
                    if (((Int64)longValue & (Int64)Math.Pow(2, cnt)) > 0)
                        txt += "1";
                    else
                        txt += "0";
                }
            }
            else
            {
                // ist ein Array
                switch (value.GetType().Name)
                {
                    case "Byte[]":
                        x = 7;
                        byte[] ByteArr = (byte[])value;
                        for (cnt2 = 0; cnt2 <= ByteArr.Length - 1; cnt2++)
                        {
                            for (cnt = x; cnt >= 0; cnt += -1)
                                if ((ByteArr[cnt2] & (byte)Math.Pow(2, cnt)) > 0) txt += "1";
else txt += "0";
                        }
                        break;
                    case "Int16[]":
                        x = 15;
                        Int16[] Int16Arr = (Int16[])value;
                        for (cnt2 = 0; cnt2 <= Int16Arr.Length - 1; cnt2++)
                        {
                            for (cnt = x; cnt >= 0; cnt += -1)
                                if ((Int16Arr[cnt2] & (byte)Math.Pow(2, cnt)) > 0) txt += "1";
else txt += "0";
                        }
                        break;
                    case "Int32[]":
                        x = 31;
                        Int32[] Int32Arr = (Int32[])value;
                        for (cnt2 = 0; cnt2 <= Int32Arr.Length - 1; cnt2++)
                        {
                            for (cnt = x; cnt >= 0; cnt += -1)
                                if ((Int32Arr[cnt2] & (byte)Math.Pow(2, cnt)) > 0) txt += "1";
else txt += "0";
                        }
                        break;
                    case "Int64[]":
                        x = 63;
                        byte[] Int64Arr = (byte[])value;
                        for (cnt2 = 0; cnt2 <= Int64Arr.Length - 1; cnt2++)
                        {
```

```
                        for (cnt = x; cnt >= 0; cnt += -1)
                            if ((Int64Arr[cnt2] & (byte)Math.Pow(2, cnt)) > 0) txt += "1";
else txt += "0";
                    }
                    break;
                default:
                    throw new Exception();
            }
        }
        return txt;
    }
    catch
    {
        return "";
    }
}

/// <summary>
/// Helper to get a bit value given a byte and the bit index.
/// Example: DB1.DBX0.5 -> var bytes = ReadBytes(DB1.DBW0); bool bit =
bytes[0].SelectBit(5);
/// </summary>
/// <param name="data"></param>
/// <param name="bitPosition"></param>
/// <returns></returns>
public static bool SelectBit(this byte data, int bitPosition)
{
    int mask = 1 << bitPosition;
    int result = data & mask;

    return (result != 0);
}

/// <summary>
/// Converts from ushort value to short value; it's used to retrieve negative
values from words
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
public static short ConvertToShort(this ushort input)
{
    short output;
    output = short.Parse(input.ToString("X"), NumberStyles.HexNumber);
    return output;
}

/// <summary>
```

```
        /// Converts from short value to ushort value; it's used to pass negative values
to DWs
        /// </summary>
        /// <param name="input"></param>
        /// <returns></returns>
        public static ushort ConvertToUshort(this short input)
        {
            ushort output;
            output = ushort.Parse(input.ToString("X"), NumberStyles.HexNumber);
            return output;
        }

        /// <summary>
        /// Converts from UInt32 value to Int32 value; it's used to retrieve negative
values from DBDs
        /// </summary>
        /// <param name="input"></param>
        /// <returns></returns>
        public static Int32 ConvertToInt(this uint input)
        {
            int output;
            output = int.Parse(input.ToString("X"), NumberStyles.HexNumber);
            return output;
        }

        /// <summary>
        /// Converts from Int32 value to UInt32 value; it's used to pass negative values
to DBDs
        /// </summary>
        /// <param name="input"></param>
        /// <returns></returns>
        public static UInt32 ConvertToUInt(this int input)
        {
            uint output;
            output = uint.Parse(input.ToString("X"), NumberStyles.HexNumber);
            return output;
        }

        /// <summary>
        /// Converts from double to DWord (DBD)
        /// </summary>
        /// <param name="input"></param>
        /// <returns></returns>
        public static UInt32 ConvertToUInt(this double input)
        {
            uint output;
            output =
S7.Net.Types.DWord.FromByteArray(S7.Net.Types.Double.ToByteArray(input));
```

```
            return output;
        }

        /// <summary>
        /// Converts from DWord (DBD) to double
        /// </summary>
        /// <param name="input"></param>
        /// <returns></returns>
        public static double ConvertToDouble(this uint input)
        {
            double output;
            output =
S7.Net.Types.Double.FromByteArray(S7.Net.Types.DWord.ToByteArray(input));
            return output;
        }
    }
}
```

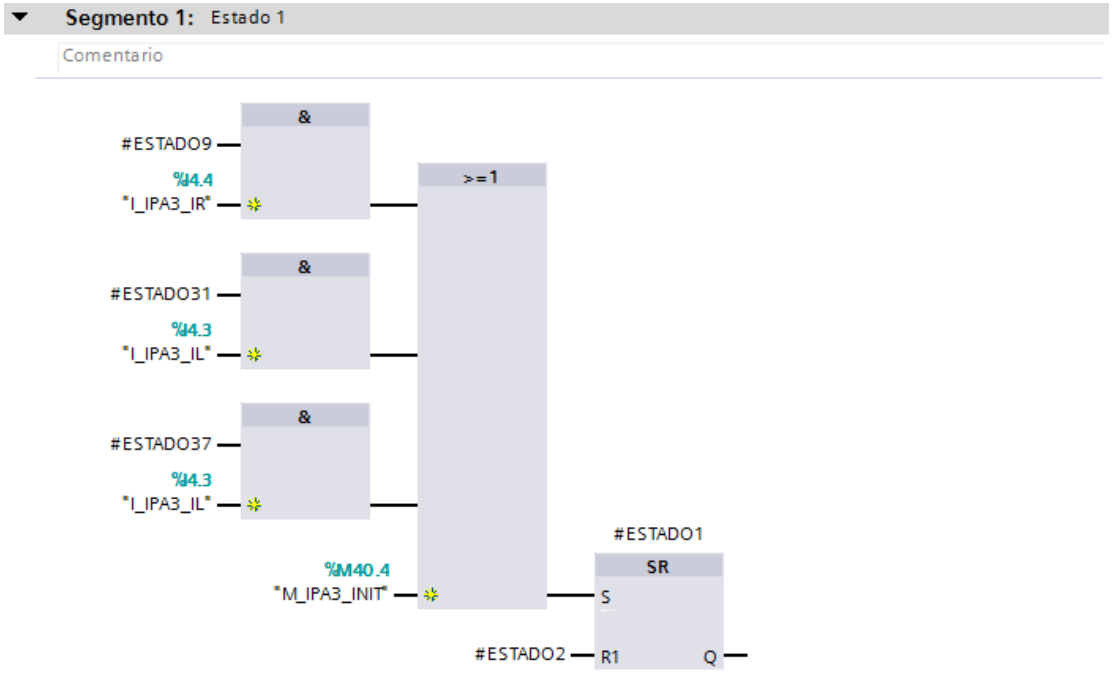
Programación en TIA Portal

A6. Bloque de comunicación

Bloque de datos_Comunicacion

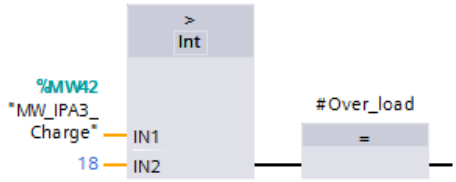
	Nombre	Tipo de datos	Offset	Valor de arranq...	Remanen...	Visible en ..	Valor de a..	Comentario
1	Static				☐	☐	☐	
2	M_IPA3_Cam_Activac...	Bool	0.0	false	☑	☑	☐	Bandera que activa a la camara
3	M_IPA3_Cam_OK	Bool	0.1	false	☑	☑	☐	No hay ningun error en las botellas
4	M_IPA3_Cam_Err	Bool	0.2	false	☑	☑	☐	Al menos una de las botellas presenta un error
5	M_IPA3_Cam_Err_Fin	Bool	0.3	false	☑	☑	☐	
6	M_IPA3_Bot1_Err	Bool	0.4	false	☑	☑	☐	La botella 1 presenta un error
7	M_IPA3_Bot2_Err	Bool	0.5	false	☑	☑	☐	La botella 2 presenta un error
8	M_IPA3_Bot3_Err	Bool	0.6	false	☑	☑	☐	La botella 3 presenta un error
9	M_IPA3_Bot4_Err	Bool	0.7	false	☑	☑	☐	La botella 4 presenta un error
10	M_IPA3_Bot5_Err	Bool	1.0	false	☑	☑	☐	La botella 5 presenta un error
11	M_IPA3_Bot6_Err	Bool	1.1	false	☑	☑	☐	La botella 6 presenta un error
12	M_IPA3_Tapa1	Bool	1.2	false	☑	☑	☐	La botella 1 esta vacia pero con tapa
13	M_IPA3_Tapa2	Bool	1.3	false	☑	☑	☐	La botella 2 esta vacia pero con tapa
14	M_IPA3_Tapa3	Bool	1.4	false	☑	☑	☐	La botella 3 esta vacia pero con tapa
15	M_IPA3_Tapa4	Bool	1.5	false	☑	☑	☐	La botella 4 esta vacia pero con tapa
16	M_IPA3_Tapa5	Bool	1.6	false	☑	☑	☐	La botella 5 esta vacia pero con tapa
17	M_IPA3_Tapa6	Bool	1.7	false	☑	☑	☐	La botella 6 esta vacia pero con tapa
18	M_IPA3_Llena1	Bool	2.0	false	☑	☑	☐	La botella 1 esta sin tapa pero con liquido
19	M_IPA3_Llena2	Bool	2.1	false	☑	☑	☐	La botella 2 esta sin tapa pero con liquido
20	M_IPA3_Llena3	Bool	2.2	false	☑	☑	☐	La botella 3 esta sin tapa pero con liquido
21	M_IPA3_Llena4	Bool	2.3	false	☑	☑	☐	La botella 4 esta sin tapa pero con liquido
22	M_IPA3_Llena5	Bool	2.4	false	☑	☑	☐	La botella 5 esta sin tapa pero con liquido
23	M_IPA3_Llena6	Bool	2.5	false	☑	☑	☐	La botella 6 esta sin tapa pero con liquido
24	M_IPA3_TapaLlena1	Bool	2.6	false	☑	☑	☐	La botella 1 esta tapada y con liquido
25	M_IPA3_TapaLlena2	Bool	2.7	false	☑	☑	☐	La botella 2 esta tapada y con liquido
26	M_IPA3_TapaLlena3	Bool	3.0	false	☑	☑	☐	La botella 3 esta tapada y con liquido
27	M_IPA3_TapaLlena4	Bool	3.1	false	☑	☑	☐	La botella 4 esta tapada y con liquido
28	M_IPA3_TapaLlena5	Bool	3.2	false	☑	☑	☐	La botella 5 esta tapada y con liquido
29	M_IPA3_TapaLlena6	Bool	3.3	false	☑	☑	☐	La botella 6 esta tapada y con liquido

A7. Bloque de transiciones



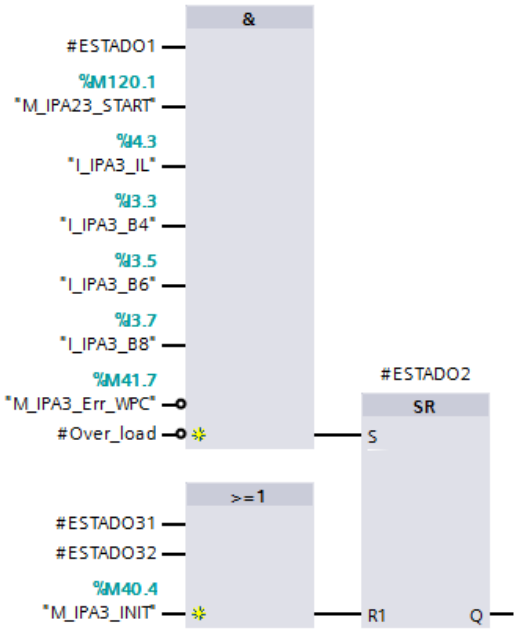
Segmento 2: SobreNivel

Comentario



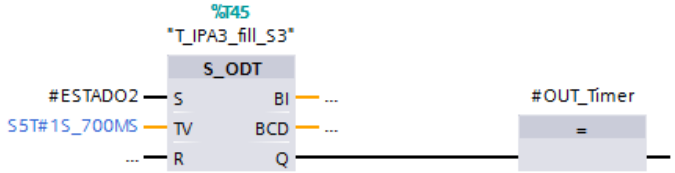
Segmento 3: Estado 2

Comentario



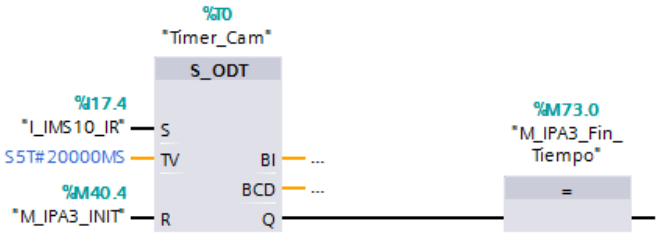
Segmento 4: Fuera de tiempo

Comentario



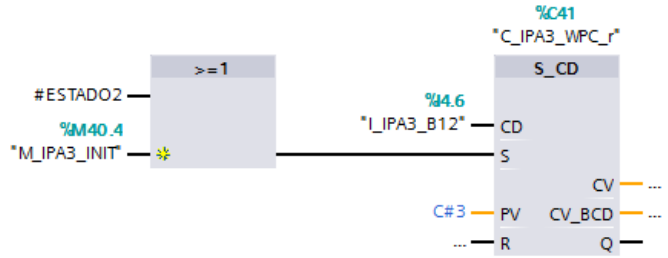
Segmento 5: Tiempo de espera para que trabaje la camara

Comentario



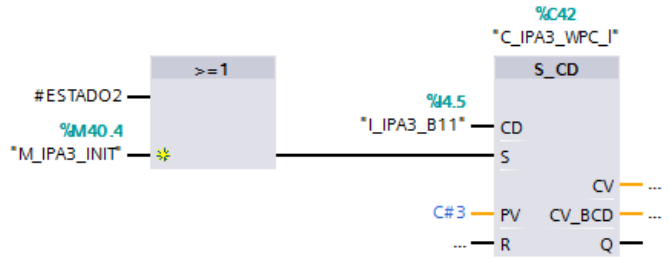
Segmento 6: Cuenta de botellas del lado derecho

Comentario



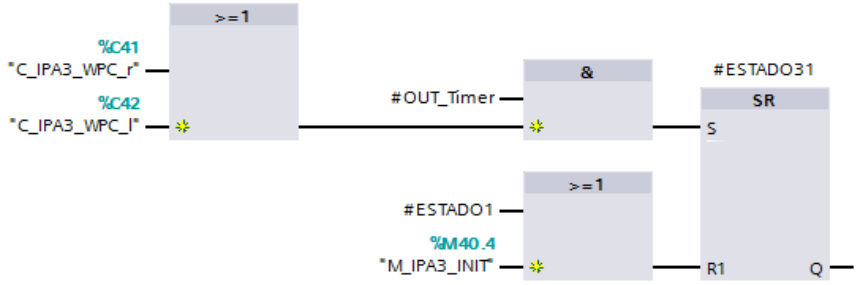
Segmento 7: Cuenta de botellas del lado izquierdo

Comentario



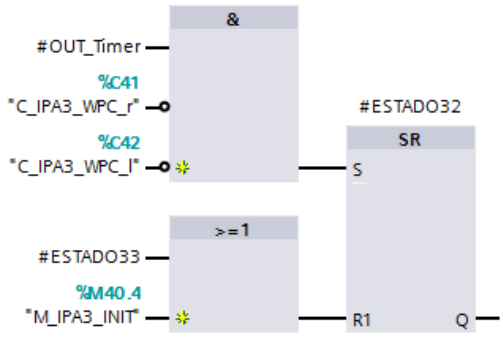
Segmento 8: Estado 3.1

Comentario



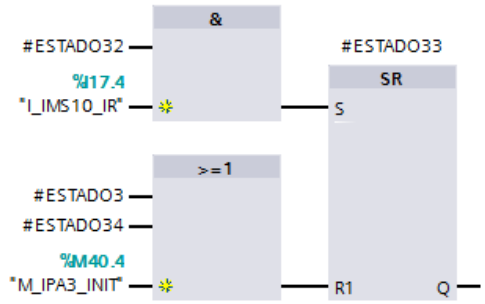
Segmento 9: Estado 3.2

Comentario



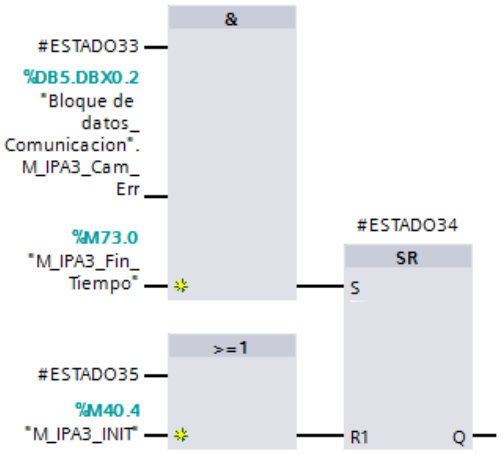
Segmento 10: Estado 3.3

Comentario



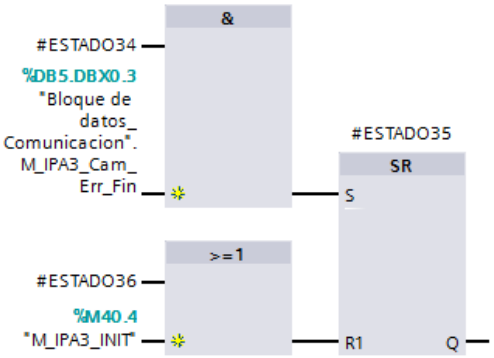
Segmento 11: Estado 3.4

Comentario



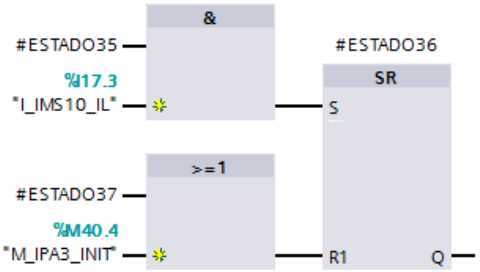
Segmento 12: Estado 3.5

Comentario



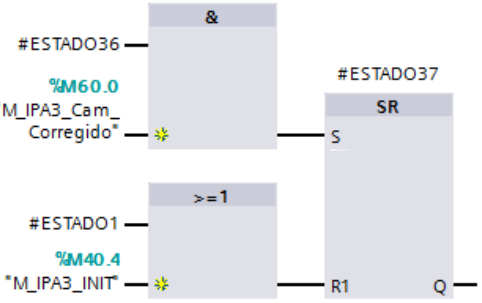
Segmento 13: Estado 3.6

Comentario



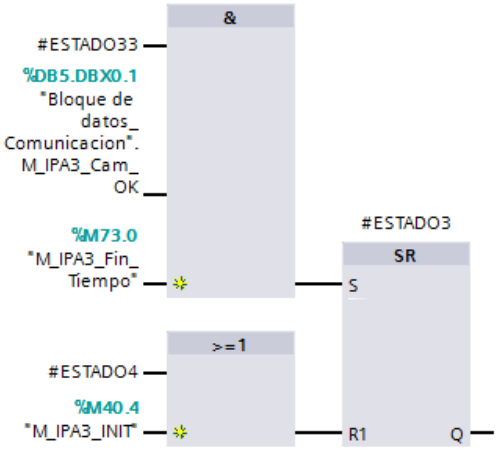
Segmento 14: Estado 3.7

Comentario



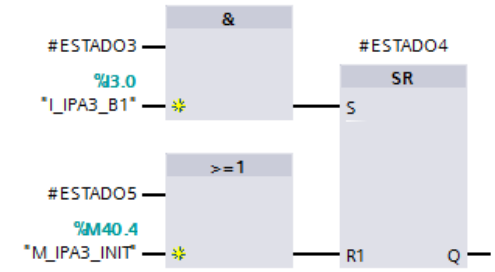
Segmento 15: Estado 3

Comentario



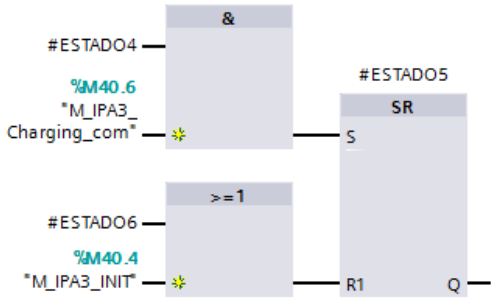
Segmento 16: Estado 4

Comentario



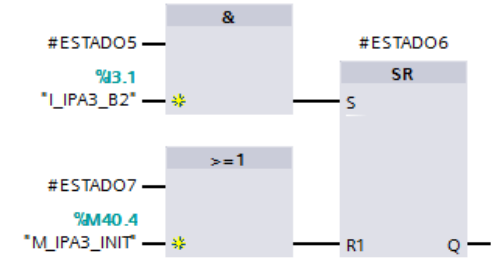
Segmento 17: Estado 5

Comentario



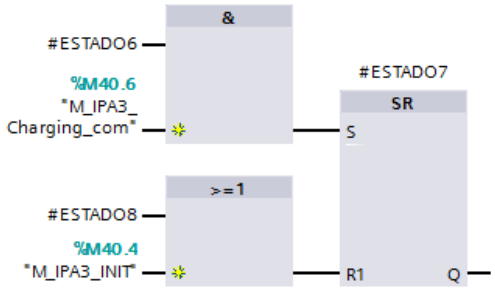
Segmento 18: Estado 6

Comentario



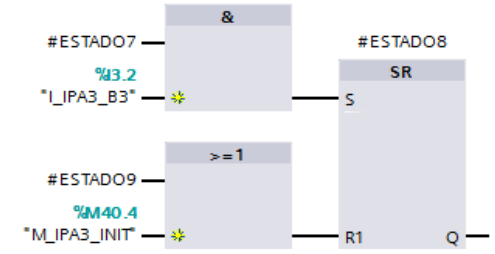
Segmento 19: Estado 7

Comentario



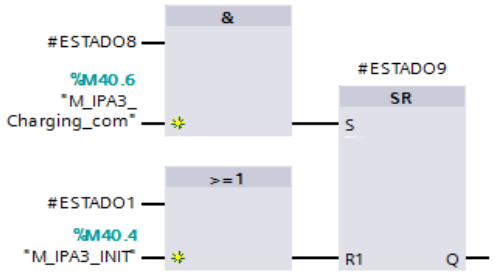
Segmento 20: Estado 8

Comentario



Segmento 21: Estado 9

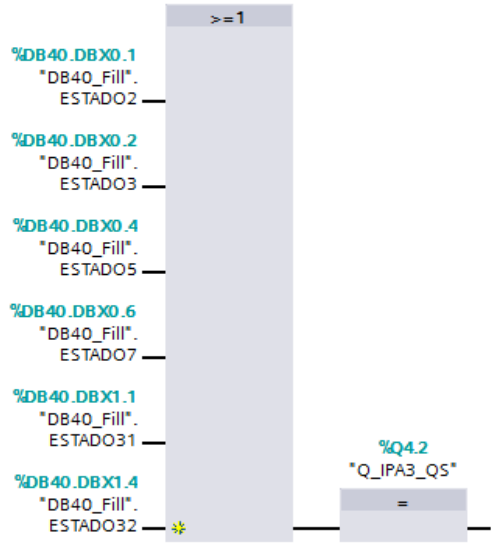
Comentario



A8. Bloque de acciones

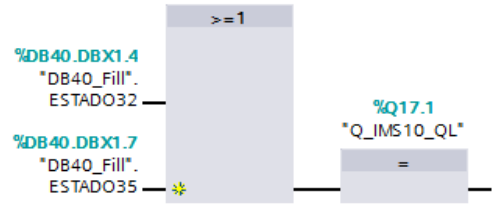
Segmento 3: Desplazamiento de cinta en marcha lenta

Comentario



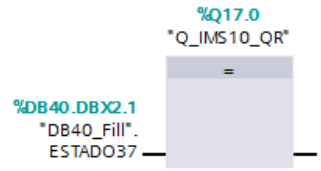
Segmento 4: Desplazamiento hacia la izquierda del IMS10

Comentario



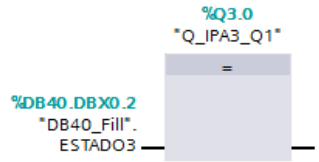
Segmento 5: Desplazamiento hacia la derecha del IMS10

Comentario



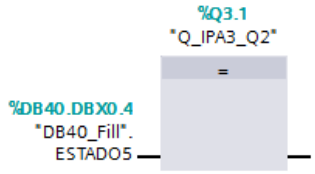
Segmento 6: Descenso de cilindro de parada en posición 1

Comentario



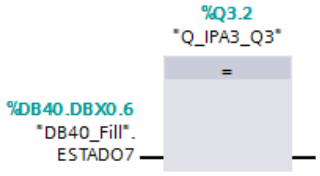
Segmento 7: Descenso de cilindro de parada en posición 2

Comentario



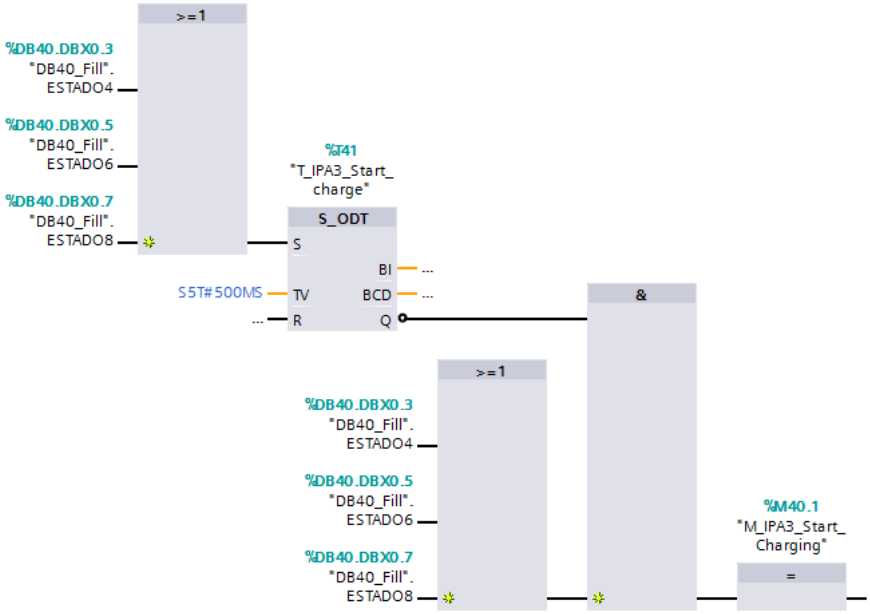
Segmento 8: Descenso de cilindro de parada en posición 3

Comentario



Segmento 9: Bandera para iniciar dosificado de botellas

Comentario



Segmento 10: Bandera que indica error en el numero de botellas

Comentario

Segmento 11: Bandera que indica que la camara puede empezar a trabajar

Comentario