ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN CONTROL Y REDES INDUSTRIALES

"ESTUDIO DE LOS ALGORITMOS DE RECONOCIMIENTO DE PATRONES PARA LA AUTOMATIZACIÓN DE UN SEMÁFORO INTELIGENTE MEDIANTE FPGAs"

TESIS DE GRADO

Previo a la obtención del título de:

"INGENIERO EN ELECTRÓNICA, CONTROL Y REDES INDUSTRIALES"

Presentado por:

GABRIELA MERCEDES MAFLA MEDINA

ALDINIVER JOSÉ ORTIZ ESPINOSA

Riobamba – Ecuador

2014

A dios por regalarnos salud, vida y conocimiento para culminar con una meta más en nuestras vidas.

A nuestros padres por la paciencia, el amor y el apoyo incondicional que nos han brindado a lo largo de nuestras vidas.

A nuestro director de tesis Ing. Alberto Arellano y colaborador Ing. Fernando Mejía por la confianza brindada, los consejos, la paciencia, las palabras de ánimo y por el apoyo incondicional acompañándonos en los momentos difíciles, ya que fueron de gran ayuda para lograr culminar con el presente trabajo.

A la Escuela Superior Politécnica de Chimborazo por abrirnos sus puertas y a los docentes de la Escuela de Ingeniería Electrónica por los conocimientos brindados.

Y a todas las personas que participaron e hicieron posible la culminación de este trabajo, nuestros sinceros agradecimientos.

Dedico la paciencia y la perseverancia que refleja la presenta tesis a mis padres por su esfuerzo a lo largo de estos años, el amor, la motivación, los consejos, y por enseñarme que con dedicación y empeño todo es posible.

A mi hermano por el cariño.

A mi amiga Tere por la paciencia, el cariño, las palabras de ánimo y consuelo, el apoyo incondicional y por la amistad brindada a lo largo de este camino.

A mis amigas y cómplices de toda la vida Taty, Vane y Evelyn, por el cariño y la amistad sincera que me han brindado en cada momento de mi vida.

Gaby

Este trabajo quiero dedicarlo a mi madre por todo el cariño y el apoyo que me ha brindado toda mi vida, a mi padre por todo el esfuerzo realizado para ayudarme siempre, a mi hijo quien siempre fue mi motivación para salir adelante y jamás rendirme, a mis hermanas y amigos.

Aldo

FIRMAS DE RESPONSABLES Y NOTA

NOMBRE	FIRMA	FECHA
Ing. Iván Menes **DECANO FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	.....	……..
Ing. Msc. Alberto Arellano **DIRECTOR DE ESCUELA INGENIERÍA ELECTRÓNICA CONTROL Y REDES INDUSTRIALES**	.....	……..
Ing. Msc. Alberto Arellano **DIRECTOR DE TESIS**	.....	……..
Ing. Paulina Vélez **MIEMBRO DEL TRIBUNAL**	.....	……..
Ing. Eduardo Tenelanda **DIRECTOR DPTO DOCUMENTACIÓN**	.....	……..
NOTA DE LA TESIS	.....	

“Nosotros **Gabriela Mercedes Mafla Medina** y **Aldiniver José Ortiz Espinosa**, somos responsables de las ideas, doctrinas y resultados expuestos en esta tesis; y, el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO”.

…… ……

Gabriela Mercedes Mafla Medina Aldiniver José Ortiz Espinosa

ÍNDICE DE ABREVIATURAS

A	Amperio
CCD	Dispositivo de carga acoplada
cm	Centímetro
CMOS	Semiconductor complementario de óxido metálico
DCA	Método Estadístico de Diseño Completamente al Azar
ESPOCH	Escuela Superior Politécnica de Chimborazo
FPGA	Arreglo de Compuertas Programable en el Campo
IMAQ	Adquisición de Imágenes, paquete de instrumentos virtuales de National Instruments
KNN	Algoritmo del vecino más cercano
LABVIEW	Laboratorio de Instrumentación Virtual de Bancos de Trabajos de Ingeniería
NI	National Instruments
LED	Diodo Emisor de Luz
ms	Milisegundos
OCR	Reconocimiento Óptico de Caracteres
PC	Computadora Personal
RGB	Rojo Verde Azul
ROI	Región de interés

SRP	Sistema de Reconocimiento de Patrones
USB	Bus serial de Comunicación Universal
V	Voltaje
VI	Instrumento Virtual
VDC	Voltaje de corriente continua
VGA	Adaptador gráfico de video

ÍNDICE GENERAL

CAPÍTULO II

MARCO TEÓRICO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

INTRODUCCIÓN

El control automático del tránsito vehicular ha sido de interés por muchos años; al ser este un problema bastante complicado y que cada día toma mayor importancia en la vida cotidiana, se ha venido investigando métodos para la implementación de sistemas de supervisión automática del tránsito, los mismos que tienen como propósito mejorar los problemas de tráfico que tenemos actualmente.

Con el transcurso de los años y el rápido avance en el desarrollo de nuevas tecnologías, las técnicas de visión artificial, principalmente el reconocimiento de patrones y sistemas inteligentes como lógica difusa, redes neuronales y algoritmos genéticos, se han convertido en un tema de actualidad y de gran aplicación para beneficio de la sociedad, poseen un enorme valor técnico y científico por los innumerables campos en los que puede ser aplicado, entre ellas el control automático del tráfico vehicular.

Las FPGAs, son ampliamente utilizadas en técnicas de visión artificial, como son aplicaciones de monitoreo de tránsito vehicular, estas permiten manejar y procesar volúmenes de imágenes muy elevados, lo cual facilita la detección de objetos, en la gran mayoría de ocasiones en tiempo real.

En este trabajo se realizó un estudio comparativo entre los algoritmos de reconocimiento de patrones, con la finalidad de determinar cuál es el más adecuado para la implementación de un prototipo de semáforos inteligentes mediante FPGAs, el mismo tiene como función minimizar la congestión vehicular.

CAPÍTULO I

MARCO REFERENCIAL

1.1. ANTECEDENTES

El semáforo se creó a principios del siglo pasado para solventar las aglomeraciones vehiculares cada vez más frecuentes de las ciudades. Pero la intención de regular la circulación y el tránsito vehicular se remonta a la época de los romanos. Siglos después se recreó el sistema para controlar la circulación del tráfico ferrovial y fluvial. Estas señales lumínicas que controlaban tanto barcos como ferrocarriles son las antecedentes directas del semáforo actual.

La mayoría de semáforos que existen actualmente funcionan por tiempo. Estos son los más anticuados y conflictivos que podemos tener en las grandes urbes. Son configurados para que la luz, ya sea verde o roja, dure un tiempo determinado. El número de segundos varía en función de la zona, horario o de

quien lo opere, ya que este tipo de semáforos también pueden ser operados manualmente.

Hoy en día, las bombillas de los semáforos están perdiendo terreno y los semáforos LED se imponen, básicamente, por un ahorro considerable de energía, por su luminosidad y por su larga vida. Las lámparas de LED utilizan sólo el 10% de energía en comparación con las bombillas incandescentes y tienen una vida útil 50 veces superior.

El congestionamiento vehicular se ha ido intensificando con el pasar de los años de manera que representa una amenaza para la calidad de vida. Hoy en día, utilizar nuestro vehículo como método de transporte es la forma más habitual de desplazarnos. La infraestructura actual que disponen las ciudades, provocan largas colas de tráfico, incrementan la polución del ambiente que desencadenan frustraciones, originan accidentes, etc.

La necesidad de agilizar el transporte en la ciudad hace necesaria la implementación de “semáforos inteligentes” que den eficacia y eficiencia al tráfico en la ciudad.

La automatización dinámica del tráfico vehicular es un campo que ofrece grandes desafíos en la aplicación de técnicas de inteligencia artificial. Es así como las investigaciones realizadas actualmente se han enfocado a la solución del problema del congestionamiento vehicular, empleando una variedad de métodos que parten desde el procesamiento digital de imágenes, hasta llegar a utilizar técnicas más avanzadas como las redes neuronales artificiales, los algoritmos genéticos y reconocimiento de patrones, entre otros.

Pese a los avances tecnológicos, las soluciones para controlar de manera automática los problemas del tráfico vehicular, no han logrado alcanzar un estado de madurez que les permita maniobrar con total autonomía. Esto ocurre principalmente por la pérdida de información que presentan las distintas técnicas de segmentación. Los problemas en la segmentación de imágenes se deben a factores ambientales como baja iluminación, nubosidad, sobrexposición, entre otros, los cuales afectan la captura de las imágenes.

El reconocimiento de patrones es un punto clave en el campo de la visión artificial, normalmente esta técnica se ve afectada por el ruido al procesar las imágenes, entre las ventajas de este método se encuentra la efectividad y velocidad de procesamiento.

Los problemas antes mencionados, sugieren un empleo o desarrollo de nuevas técnicas que puedan ser aplicadas en la construcción de sistemas de control de tráfico vehicular.

Hoy en día las FPGAs son utilizadas en diversos campos como la visión artificial, sistemas de seguridad, sistemas biomédicos, entre otros, sin embargo la principal aplicación de la tecnología FPGA es en industrias que necesitan computación de alta velocidad.

Con el avance de la tecnología las FPGAs se han convertido en una herramienta indispensable en el diseño rápido y eficiente de proyectos que requieren procesamiento digital y lógica programable. Es por eso que se ha convertido en la principal herramienta para la implementación de los algoritmos de procesamiento digital de imágenes y reconocimiento de patrones.

1.2. JUSTIFICACIÓN

En ciudades como Quito, Guayaquil, Cuenca, Ibarra y Ambato ya se han implementado “semáforos inteligentes”, esta infraestructura posibilita modos de operación que se activan dependiendo de la situación real del tráfico, los mismos se instalaron en puntos estratégicos de las ciudades anteriormente mencionadas, con el fin de proporcionar información real sobre el flujo vehicular de las principales intersecciones.

El Ecuador consta con 217 ciudades, de las cuales solo 5 de ellas ya han implementado el nuevo sistema de semaforización inteligente, lo cual quiere decir que esta nueva tecnología es adoptada tan solo por el 3% aproximadamente de las ciudades de nuestro país.

El sistema de semaforización actual de la ciudad de Riobamba funciona con relés temporizadores, los mismos que resultan poco eficaces para controlar el tráfico vehicular sobre todo en las denominadas horas pico, este mal funcionamiento de los mismos ha provocado accidentes afectando la calidad de vida de los ciudadanos.

Los semáforos inteligentes que se han instalado en algunas de las ciudades de nuestro país funcionan con sensores inductivos y controladores lógicos programables como son los PLCs, pero aún estos no utilizan la tecnología de reconocimiento de patrones y tampoco la implementación de la circuitería sobre FPGAs.

Años atrás los sistemas de visión artificial funcionaban sólo con información obtenida de imágenes estáticas, esto se debía a que dichos sistemas trabajaban con una cantidad limitada de información y a velocidades relativamente lentas. En la actualidad, el procesamiento de imágenes permite trabajar con mayores

volúmenes de información y a mayor velocidad, debido al gran avance tecnológico que ha tenido el campo de la visión artificial se comienza a investigar algoritmos para el procesamiento de secuencias de imágenes.

La visión artificial se utiliza principalmente en aplicaciones donde el seguimiento y caracterización de objetos en movimiento es de primordial importancia. Una de las aplicaciones se puede evidenciar en sistemas de control de tráfico vehicular, donde es posible realizar una automatización dinámica en los sistemas de semaforización, para que actúen en función a las características del tráfico.

Por las razones presentadas anteriormente creemos indispensable la creación de un nuevo diseño para el sistema de control de tráfico vehicular que funcione a través de reconocimiento de patrones y FPGAs, lo cual garantizará un diseño menos costoso que los actuales, de fácil instalación, manejo y mantenimiento, actuarán inteligentemente de acuerdo a las condiciones de tráfico existentes y sobre todo este sistema estará a la vanguardia de las tecnologías actuales en el mundo.

1.3. OBJETIVOS

1.3.1. Objetivo General

Estudiar los algoritmos de reconocimiento de patrones utilizando técnicas de procesamiento digital de imágenes para la automatización de un semáforo inteligente mediante FPGAs.

1.3.2. Objetivos Específicos

- Analizar los algoritmos de reconocimiento de patrones.
- Evaluar las características técnicas de los dispositivos de lógica programable a utilizar en el semáforo inteligente.
- Diseñar el prototipo del semáforo inteligente.
- Implementar el prototipo y realizar pruebas para determinar la confiabilidad del prototipo.

1.4. HIPÓTESIS

El estudio de los algoritmos de reconocimiento de patrones ayudará a seleccionar el adecuado, para la construcción de un prototipo de semáforo inteligente mediante FPGAs, para minimizar la congestión vehicular.

CAPÍTULO II

MARCO TEÓRICO

2.1. HISTORIA DE LOS SEMÁFOROS

El semáforo tuvo sus inicios en la época de los romanos, posteriormente este sistema fue utilizado para controlar el tráfico ferrovial, el cual servía para indicar si un tren podía o no entrar a una estación y de esta manera evitar accidentes.

A partir del año 1842 en Inglaterra se empezaron a usar señales, las mismas que se fueron estandarizando durante los siguientes años, en 1870 se lograron unificar las ideas, de esta manera se pudo tener un sistema bastante entendible, que posteriormente se extendería por todo el mundo.

Los primeros semáforos eran operados de forma mecánica, éstos utilizaban unas paletas con colores y señales que se levantaban o bajaban a palanca o por poleas y contrapesos.

En el año 1868 el ingeniero ferroviario, J. P. Knight instaló el primer semáforo de luces de tránsito de la historia, el cual se ubicaba en el exterior del parlamento británico de Westminster, un lugar muy transitado, éste funcionaba con dos brazos mecánicos, uno rojo, uno verde, y una lámpara de gas para su uso nocturno.

Fuente: http://www.gstriatum.com/info/index.php?option=com_content&view=article&id=1498:lo-que-no-sabias-de-los-semaforos&catid=72:cultura&Itemid=59

Figura II.1. Primer semáforo de la historia

En 1910 Earnest Sirrine lo modificó para que sea automático y no requiera un operador. Pero el semáforo eléctrico que conocemos proviene de los EEUU que aunque era manual su operación utilizaba luz eléctrica para la roja y la verde, creado en 1912 por Lester Wire en Salt Lake City, poseía un zumbador para avisar del cambio de luces y además se podía cambiar su funcionamiento para que los bomberos y policía tuviesen control. El primer diseño automático data de 1917, patentado por William Ghiglieri de San Francisco, EEUU, con modo manual para aquel que lo requiera, pero la automatización empezaba a tomar el poder del semáforo.[1]

[1] http://www.redbull.com/cs/Satellite/es_AR/Article/La-historia-del-semáforo-021243312125605

El primer semáforo que combinaba todo, eléctrico, automático e interconectado fue instalado en Houston en 1922, y en el año 1927 se realizó el primer experimento de un semáforo totalmente automático en Wolverhampton, Inglaterra.

En los años 60’s se empezaron a utilizar relés y válvulas para controlar el uso de las luces interconectadas, y recién en los 90´s se emplearon en algunos países contadores en los semáforos.

En los últimos años se han desarrollado semáforos más ecológicos y duraderos, los cuales utilizan lámparas LED en vez de las ineficientes incandescentes.

Los semáforos inteligentes que existen actualmente detectan la cantidad de vehículos a través de sensores, los mismos son ubicados en la carpeta asfáltica, y en base a parámetros establecidos en un centro de control, van modificando los tiempos de paso o detención.

Fuente: http://www.crearcrear.com/semaforos-inteligentes/

Figura II.2. Semáforos inteligentes

2.2. VISIÓN ARTIFICIAL

Nilsson (2001, p.75-76) nos dice que:

La visión por computador es el campo de la Inteligencia Artificial que estudia los sistemas dotados con la capacidad de ver el entorno que les rodea. Este campo es muy extenso y abarca desde las técnicas generales hasta las más especializadas, cubriendo una gran gama de aplicaciones, que incluyen el reconocimiento de caracteres, la interpretación de fotografías, la identificación de huellas dactilares y el control de robots.

Aunque la visión es una actividad que aparentemente no supone ningún esfuerzo para los humanos, para las máquinas supone un problema muy complejo. Las mayores dificultades surgen cuando los sistemas tienen que operar en condiciones de iluminaciones variables y no controladas, con sombras, o tienen que tratar con objetos complejos y difíciles de describir, y con objetos ocultos, como ocurre en las escenas de interiores y en escenas con objetos no rígidos. Algunos de estos problemas se pueden reducir si tratamos con entornos artificiales como el interior de los edificios, y, por consiguiente, la visión por computador ha sido aplicada en dichos entornos con un mayor éxito.

El primer paso en la visión por computador es la creación de una imagen de la escena en una matriz de dispositivos fotosensibles, como, por ejemplo, las fotocélulas de una cámara de TV. La cámara forma la imagen a través de una lente que produce una proyección en perspectiva de la escena que está dentro del campo visual de la cámara. Las fotocélulas convierten la imagen en una matriz de valores de intensidad que depende del tiempo, I(x, y, t), donde x e y indican la localización de la fotocélula en la matriz, y t indica el tiempo en el que la imagen ha sido formada (cuando tratamos con imágenes en color se forman tres de estas imágenes – una por cada una de los colores primarios). Por tanto, los agentes reactivos basados en visión deben procesar esta matriz para crear una representación icónica del entorno que les rodea, o un conjunto de características a partir de las cuales pueden calcular la acción que deben tomar.

El tipo de información que se extrae de las imágenes depende del propósito y de las tareas del agente. Para navegar de forma segura por un entorno desordenado, un agente necesitará conocer la localización de los objetos, sus bordes, las puertas y las propiedades de la superficie sobre la que se define su trayectoria.

Para manipular objetos necesitará saber la localización de los objetos, tamaños, formas, composición y texturas. Para otros propósitos, puede ser necesario conocer su color y ser capaz de reconocerlos como elementos de una determinada clase. Basándonos en como varía esta información en un determinado intervalo de tiempo, un agente podría requerir la capacidad de predecir posibles cambios futuros. La extracción de esta información a partir de una o más imágenes es una labor compleja.

2.2.1. IMAGEN DIGITAL

Una imagen digital es aquella imagen que se captura por un medio electrónico y se representa por un número finito de elementos denominados píxeles, donde cada uno de ellos tiene una ubicación particular y un valor específico.

El término Píxel es la abreviatura de la expresión inglesa Picture Element o Elemento de Imagen, y es la unidad más pequeña que encontraremos en las imágenes compuestas por mapa de bits. En realidad cada píxel se compone de tres registros de color azul, rojo y verde, mediante la combinación de cierta cantidad de cada uno de estos registros de color el píxel adopta un color particular.

2.2.2. ADQUISICIÓN DE IMÁGENES

Para la adquisición de imágenes en tiempo real se utilizan cámaras de video, las cuales tienen la tarea de convertir las ondas de luz en señales eléctricas, o en su correspondiente formato de video.

LabView trabaja con un sinnúmero de cámaras para la adquisición de imágenes, entre ellas están las webcams, microscópicas, escáneres entre otras, las mismas que se clasifican en dos grupos, las cámaras analógicas y las digitales.

El proceso para adquirir una imagen digital es similar al sistema humano de visión:

- La imagen es adquirida por la cámara.
- La imagen adquirida es convertida de analógica a digital, para que pueda ser procesada.
- La imagen digital es utilizada como dato de entrada, para que posteriormente a través de un computador y dependiendo del programa de aplicación que se haya diseñado, el mismo se encargará de realizar los cálculos correspondientes.

Fuente: http://sensoresyequipos.blogspot.com/2012/05/vision-artificial.html

Figura II.3. Proceso para adquirir una imagen

2.2.3. DIGITALIZACIÓN DE IMÁGENES

Es el proceso de transformación de una imagen analógica a una imagen digital.

Cuando se digitalizada una imagen bidimensional, ésta queda formada por un conjunto de elementos básicos llamados píxeles. Cada píxel contiene cierta información sobre una región elemental de la imagen, como el color o brillo, y la posición. En imágenes en blanco y negro esta información es el brillo. En imágenes a color, la información corresponde a la intensidad de cada una de las componentes de un modelo de color como: RGB, CMYK, HSI, etc. [2]

Para la digitalización de una imagen se realizan dos procesos:

- Muestreo
- Cuantización

MUESTREO

En este proceso se obtiene la imagen, cuando un dispositivo de captura muestrea la imagen fotográfica, divide la imagen en píxeles, y se muestrea en una matriz con *m* filas y *n* columnas. El muestreo se utiliza para determinar el tamaño del píxel y el valor del brillo que se le dará a la imagen.

En las siguientes figuras se puede observar ejemplos de muestreos con diferentes valores para *m* y *n*.

[2] http://vision-artificial-matlab.blogspot.com/

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura II.4. Imagen con baja resolución **Figura II.5.** Imagen con alta resolución

CUANTIZACIÓN

En este proceso se asignan los valores a los elementos de la matriz. Cada uno de los valores asignados representa al valor de la variable física en ese punto. En la representación visual se asume un rango de valores y el valor más pequeño de dicho rango corresponde a un nivel de gris negro y el valor más grande al nivel de gris blanco.

2.2.4. PROPIEDADES DE UNA IMAGEN DIGITAL

Una imagen digital tiene tres propiedades básicas:

- ***Resolución*:** en esta propiedad se representa el número de filas y el número de columnas de píxeles que contiene la imagen.

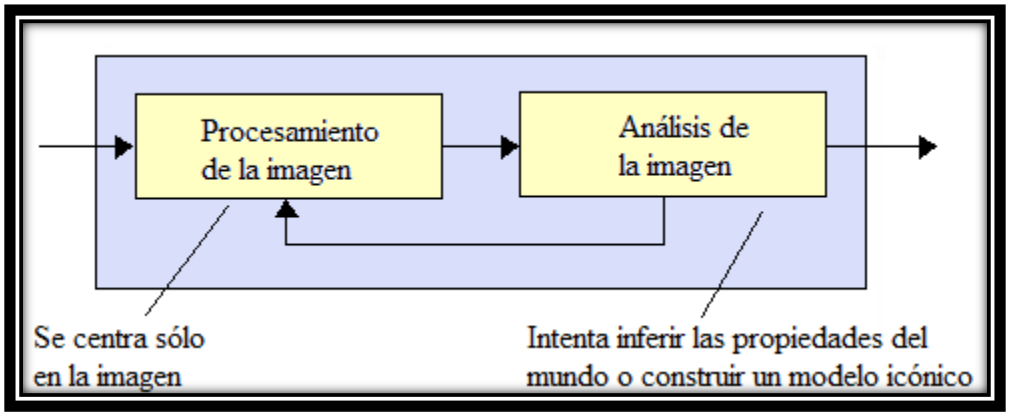
- ***Definición*:** muestra el grado de nitidez que se puede observar en una imagen. Esta propiedad está relacionado con el número de bits usados para codificar el valor de un píxel. Para un bit de profundidad *n*, un píxel puede tomar 2^n valores diferentes.

- ***Número de Planos:*** es el número de matrices o de arreglos de píxeles por los que se encuentra formada la imagen. Las imágenes que se encuentran en una escala de grises se compone por un solo plano, en cambio una imagen a color se compone por tres planos.

2.2.5. PROCESAMIENTO DE IMÁGENES

El procesamiento digital de imágenes consiste en la aplicación de varias operaciones de filtrado sobre un conjunto de datos de imagen, para reducir el ruido, remarcar los bordes y encontrar regiones dentro de la imagen, facilitando su interpretación.

Algunas de las operaciones utilizadas en el procesamiento de imágenes son reconocimiento de patrones, codificación, extracción de características, entre otras. Dichas operaciones se expresan generalmente en forma algorítmica, esto quiere decir implementado un software, aunque en algunas ocasiones debido a la velocidad que se requiere en algunas aplicaciones es necesario utilizar un hardware especializado.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura II.6. Etapas de la visión artificial

Las operaciones que se realizan en el procesamiento de imágenes y que tienen como resultado otra imagen se pueden clasificar en tres grupos:

Operaciones puntuales: la operación sobre un pixel de la imagen de salida se realiza sin tener en cuenta los pixeles vecinos.

Operaciones locales: la operación para obtener un pixel en la imagen de salida tiene en cuenta tanto el pixel correspondiente en la imagen de entrada como una cantidad arbitraria de vecinos de éste.

Operaciones globales: la operación para obtener un pixel en la imagen de salida tiene en cuenta tanto el pixel correspondiente en la imagen de entrada así como todos los demás píxeles en la imagen original.

2.2.6. ANÁLISIS DE IMÁGENES

El análisis de imágenes extrae información necesaria sobre el contenido de una imagen o de objetos dentro de la misma.

Combina técnicas que describen la distribución de la intensidad de los píxeles en la imagen, con el fin de determinar cuántos pixeles existen para cada valor de la escala de grises, y de esta manera comprender el contenido de la imagen y así poder decidir el tipo de herramienta de inspección a usar en una determinada aplicación.

Para realizar el análisis de imágenes se utiliza ciertas funciones como:

- Histograma
- Perfiles de línea
- Medición de intensidades

2.2.7. SEGMENTACIÓN

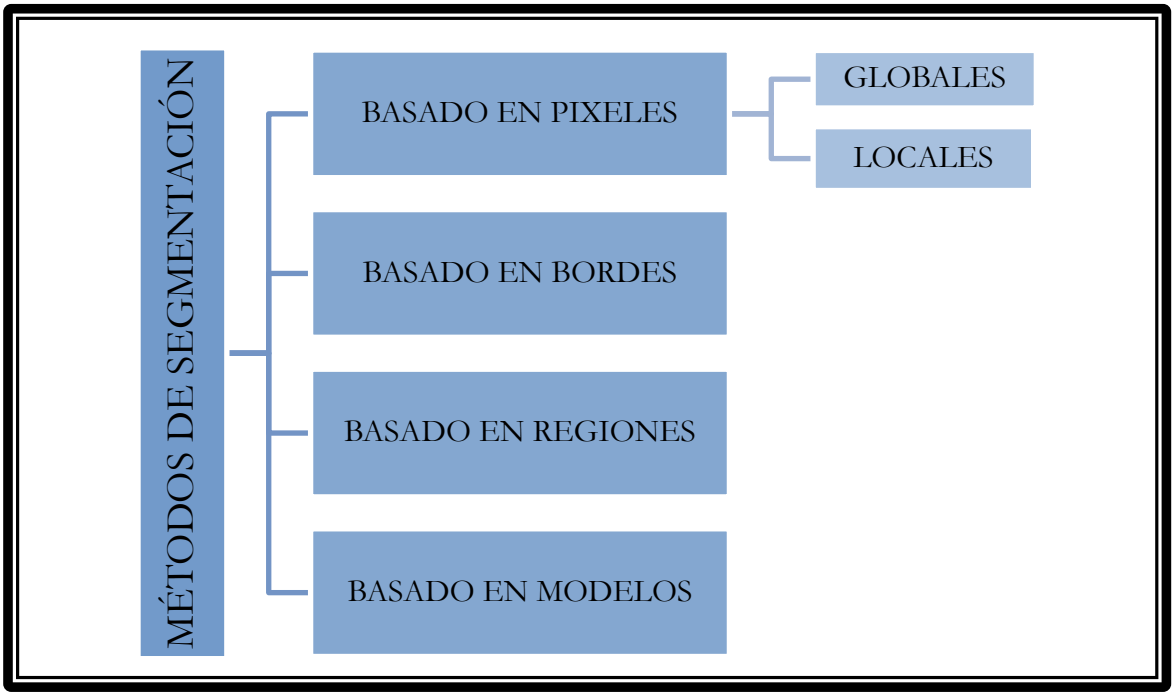
Segmentar una imagen es dividir la misma en regiones u objetos de interés. Para la segmentación de una imagen se pueden tomar en cuenta varias características como la textura, la dirección de los bordes, los tonos de gris, la magnitud del gradiente, entre otras.

La operación de segmentación tiene como objetivo identificar si un píxel pertenece, o no, al objeto de estudio, para posteriormente producir una imagen binaria.

Existen algoritmos para la segmentación de imágenes, éstos se basan en algunas propiedades como:

- Discontinuidad: dividir la imagen en base a los cambios de intensidad.
- Similitud: dividir la imagen en regiones u objetos que sean similares en base a criterios previamente definidos.
- Conectividad: de los pixeles que conforman la imagen.

Para segmentar una imagen existen diferentes métodos, los cuales se pueden agrupar en cuatro clases diferentes, como se puede observar en la figura II.7.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura II.7. Métodos de segmentación

2.2.8. DESCRIPTORES

Un descriptor de una imagen es una manera de representar a una imagen por sus características, con fines de almacenamiento y recuperación.[3]

2.2.8.1. MÉTODOS DE DESCRIPCIÓN

Existen varios métodos para describir los objetos presentes en una imagen digital. Los más comunes son:

- ***DESCRIPTORES DE FORMA***

  Los descriptores de forma parten de una información binaria de pertenencia de un pixel al objeto. Dentro de estos podemos distinguir los siguientes:

[3] http://recuperaciondeinformacionmultimodal.blogspot.com/2011/05/descriptores-de-imagenes.html

a. Descriptores de Contorno: se encargan de la información binaria de pertenencia al contorno.
b. Descriptores de Región: se encargan de la información binaria de pertenencia al interior del objeto, no sólo con los del contorno.

❖ ***DESCRIPTORES DE TEXTURA***

a. Niveles de gris: aquellos que parten del histograma del objeto.
b. Distribución espacial de niveles: aquellos que parten de información de distribución espacial del nivel de gris.

2.2.9. ILUMINACIÓN

La parte más crítica para un sistema de visión es la iluminación. Una iluminación adecuada facilita la identificación del objeto de interés, resalta sus características, disminuye la complejidad de la imagen y mejora el tiempo de respuesta del procesamiento digital.[4]

Las cámaras tienen como objetivo capturar la luz reflejada de los objetos. En aplicaciones de visión la iluminación tiene como propósito controlar la forma en que la cámara va a ver el objeto.

Existen diferentes tipos de fuente de luz entre los principales tenemos:

- Lámparas de Tungsteno.
- Lámparas fluorescentes
- Halógenos
- Diodos emisores de luz
- Láser

[4] https://decibel.ni.com/content/docs/DOC-31879

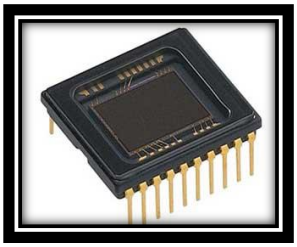
2.2.10. CÁMARA

La función de la cámara en un sistema de visión artificial es adquirir la imagen proyectada por el sensor para posteriormente transferirla a un sistema electrónico.

Debido a su rápida evolución las cámaras se clasifican de acuerdo al tipo de sensor que utilizan. Estos pueden ser CCD (Charge Coupled Device o Dispositivo de carga acoplada) y CMOS (Complementary Metal Oxide Semiconductor o Semiconductor complementario de óxido metálico).

TECNOLOGÍA CCD

Los sensores CCD se encargan de convertir una señal luminosa en una señal eléctrica. Para definir los píxeles en una cámara CCD ésta se compone de filas y columnas. Este tipo de sensores tienen mayor sensibilidad lumínica, lo que conlleva a que las imágenes captadas tengan mejor eficacia, incluso en condiciones de poca luz.



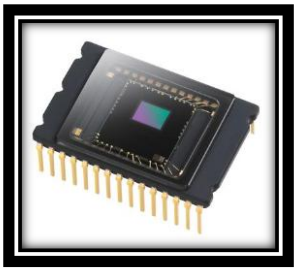
Fuente: http://www.videovigilancia.eu.com/blog/videovigilancia/sensores-ccd-y-cmos

Figura II.8. Sensor CCD

TECNOLOGÍA CMOS

Los sensores CMOS nos ofrecen imágenes de buena calidad a menor costo, con el avance de la tecnología cada día se acercan a su homólogo CCD en relación a términos de calidad de la imagen. Estos sensores tienen un tiempo

menor de lectura y una disipación de energía menor a nivel del chip, una de las desventajas es que tienen menor sensibilidad a la luz.



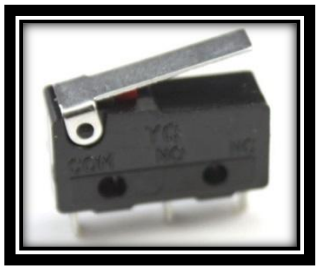
Fuente:http://ounae.com/sony-tambien-expande-su-produccion-de-cmos/

Figura II.9. Sensor CMOS

2.3. SENSORES

MICRO SWITCH

El micro switch es un conmutador de 2 posiciones con retorno a la posición de reposo y viene con un botón o con una palanca de accionamiento.[5]



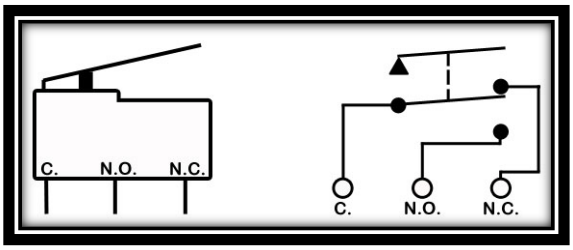
Fuente: http://www.aliexpress.com/micro-switch-sensor_reviews.html

Figura II.10. Micro Switch

[5] http://www.profesormolina.com.ar/tecnologia/sens_transduct/tipos.htm

Funcionamiento

En estado de reposo la patita común (COM) y la de contacto normal cerrado (NC), se encuentran en contacto permanente, se mantiene este estado hasta que la pequeña platina acerada interior salta debido a la presión aplicada a la palanca del micro switch, cuando esto sucede el contacto pasa de la posición de normal cerrado a la de normal abierto (NO), cuando el micro switch cambia de estado, se oye un pequeño clic, esto sucede casi al final del recorrido de la palanca.[6]



Fuente: http://www.tamiyausa.com/items/geniuseries-educational-kits-50/elecraft-series-41000/5a-micro-switch-75016

Figura II.11. Diagrama de conexión

Características

Tabla II.1. Micro Switch - Características

Amperaje	0.1 A, 6 A, 10.1 A
Circuitos	SPDT, SPNO, SPNC
Fuerza de funcionamiento	0,78 oz a 11,01 oz
Terminaciones	Conexión rápida, soldadura, pcb
Actuadores / Palancas	Émbolo de aguja, recto, rodillo, sim. rodillo
Voltaje	125 Vac, 250 Vac, 6(2) A 250 Vac
Aprobaciones	UL, cUL, ENEC, CE
Temperatura de funcionamiento	-25 °C to 85 °C [-13 °F to 185 °F]
Contactos	Plata, plata chapada en oro, plata-óxido de estaño e indio
Material de la carcasa	Poliamida (nylon)

Fuente: http://sensing.honeywell.com/honeywell-sensing-micro-switch-basic-switches-line-guide-004649-7-en.pdf

[6] http://www.profesormolina.com.ar/tecnologia/sens_transduct/tipos.htm

Beneficios

Son de tamaño pequeño y peso ligero con una amplia capacidad eléctrica, bajo costo y de larga vida. Los terminales reducen la posibilidad de cambiar la contaminación de la cavidad.

2.4. SOFTWARES PARA VISIÓN ARTIFICIAL

Existen diferentes lenguajes de programación que se utilizan para trabajar con visión artificial. Entre los principales tenemos: LabView, Matlab, OpenCV, Python, entre otros.

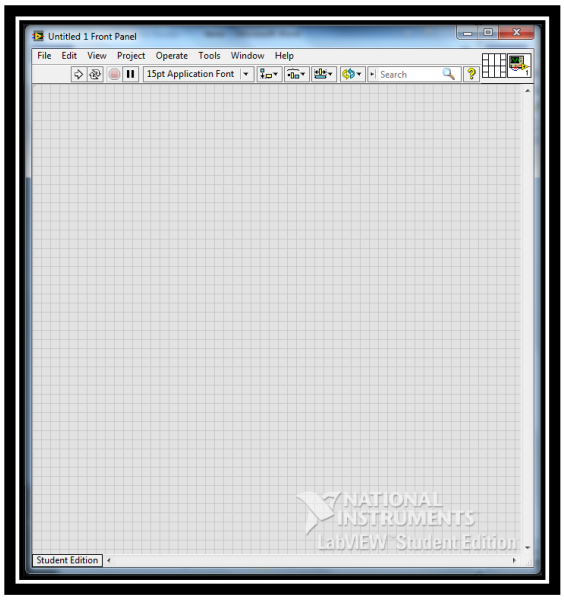
2.4.1. LABVIEW

LabView *(Laboratory Virtual Instrument Engineering Workbench)* es un lenguaje de programación gráfico para el diseño de sistemas de medición, adquisición de datos, procesamiento digital de señales, entre otros.

Este programa nos permite una fácil integración con distintos tipos de hardware como tarjetas de medición, adquisición de imágenes y procesamiento de datos.

Para la programación LabView consta de dos paneles, un panel frontal y un panel de programación o también llamado diagrama de bloques.

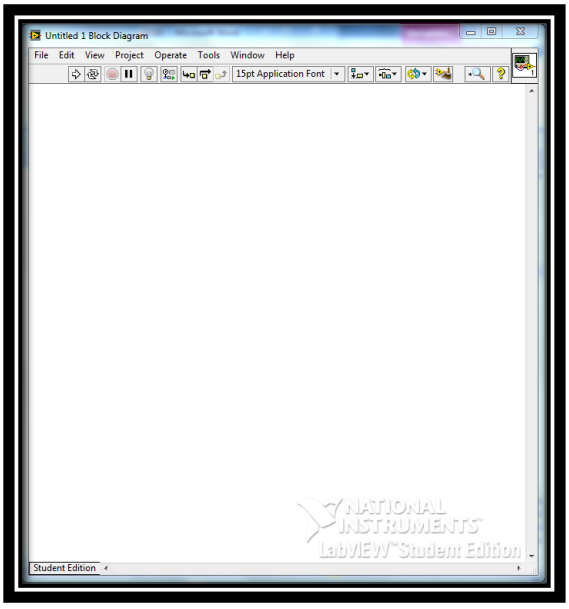
Panel frontal: en este panel se diseña la interfaz gráfica con el usuario, esto se lo puede realizar mediante el uso de los controles e indicadores.



Fuente: LabView 2012

Figura II.12. Panel Frontal

Panel de programación: en este panel se relacionan los elementos utilizados en la interfaz, se crea el código gráfico y se desarrolla toda la programación.



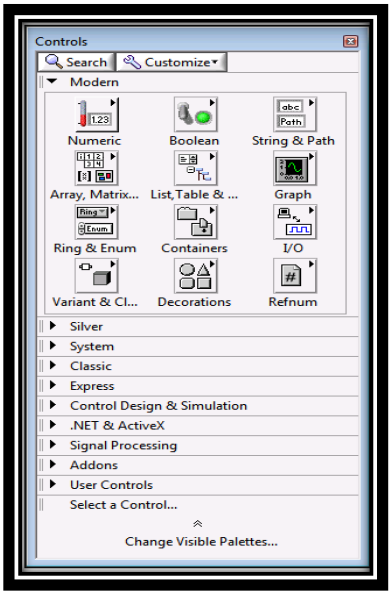
Fuente: LabView 2012

Figura II.13. Panel de Programación

ENTORNO LAVBIEW

- **Paleta de controles**

La paleta de controles nos sirve para desarrollar el panel frontal, ya que ésta contiene los controles e indicadores. Al colocar un control o un indicador en el panel frontal automáticamente se creará un terminal en el diagrama de bloques.



Fuente: http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

Figura II.14. Paleta de controles

Un control es un objeto que sirve para aportar datos al diagrama de bloques del VI.

Un indicador es un objeto que muestra los datos obtenidos por el programa durante su ejecución. Estos pueden estar representados por LEDs, secuencias de estado, tablas y gráficas.

♣ ***Controles e Indicadores Numéricos***

Los datos numéricos representan números de tipo entero o real.

♣ ***Controles e Indicadores Booleano***

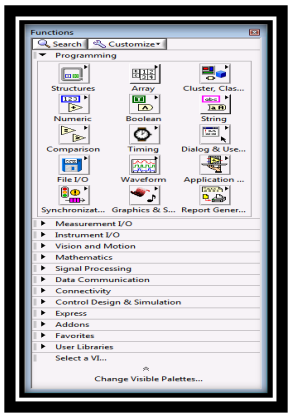
El tipo de datos Booleano representa datos que solamente tienen dos estados posibles, como TRUE y FALSE u ON y OFF. Los objetos Booleano simulan interruptores, botones y LEDs.[7]

♣ ***Controles e Indicadores de Cadena de Caracteres***

El tipo de datos cadena de caracteres es una secuencia de caracteres ASCII.[8] Estos sirven para mostrar texto al usuario. Para representar este tipo de datos se utiliza comúnmente tablas y cuadros de texto.

- **Paleta de Funciones**

En la paleta de funciones se encuentran los VIs, constantes y funciones para poder crear el diagrama de bloques.



Fuente: http://www.ni.com/images/gettingstarted/neutral/functions_palette.gif

Figura II.15. Paleta de funciones

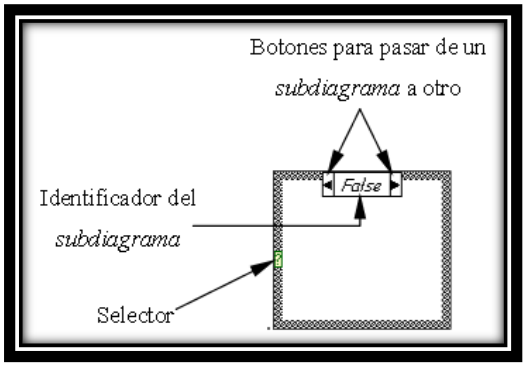
[7] http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm
[8] http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

♣ Estructuras

Dentro de la paleta de funciones podemos encontrar las diferentes estructuras con las que cuenta LabView, éstas son las que van a controlar el programa mediante la ejecución de bucles, secuenciación de acciones, etc.

CASE

La estructura case ejecuta los códigos dependiendo de la condición. Esta estructura tiene dos subdiagramas, True y False, se ejecutará solo uno de los dos, dependiendo del valor que se conecte al selector.

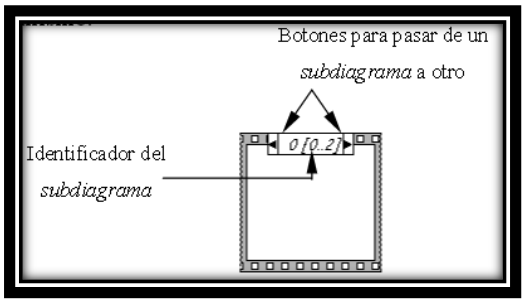


Fuente: http://webs.uvigo.es/euiti_ie1/ie2/Tutorial%20de%20Labview.pdf

Figura II.16. Estructura Case

SEQUENCE

Esta estructura sigue una secuencia para la ejecución del programa, de forma que primero se ejecutará el Frame n^{o} 0, seguidamente el n^{o} 1, y así sucesivamente.

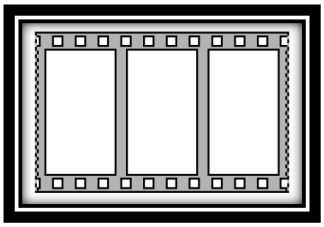


Fuente: http://webs.uvigo.es/euiti_ie1/ie2/Tutorial%20de%20Labview.pdf

Figura II.17. Estructura Sequence

FLAT SEQUENCE

En este caso los frames se pueden observar uno a continuación del otro, su orden de ejecución va de izquierda a derecha.

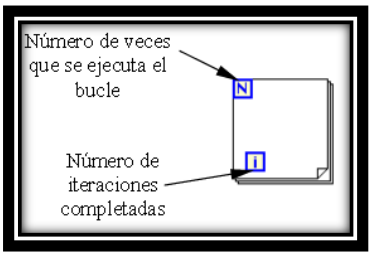


Fuente: http://www.gte.us.es/ASIGN/IE_4T/Tutorial%20de%20Labview.pdf

Figura II.18. Estructura flat sequence

FOR LOOP

Con esta estructura el código dispuesto en su interior se ejecuta un número determinado de veces.

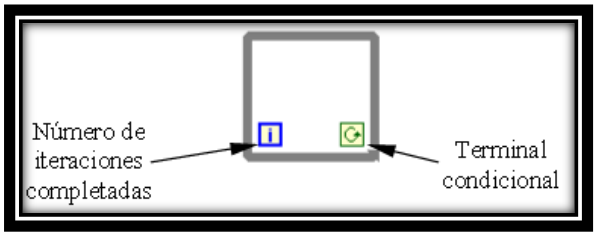


Fuente: http://webs.uvigo.es/euiti_ie1/ie2/Tutorial%20de%20Labview.pdf

Figura II.19. Estructura For

WHILE LOOP

Su funcionamiento es similar al del bucle for, el código de su interior se ejecuta hasta que se cumpla una condición, la cual es evaluada en cada iteración.



Fuente: http://webs.uvigo.es/euiti_ie1/ie2/Tutorial%20de%20Labview.pdf

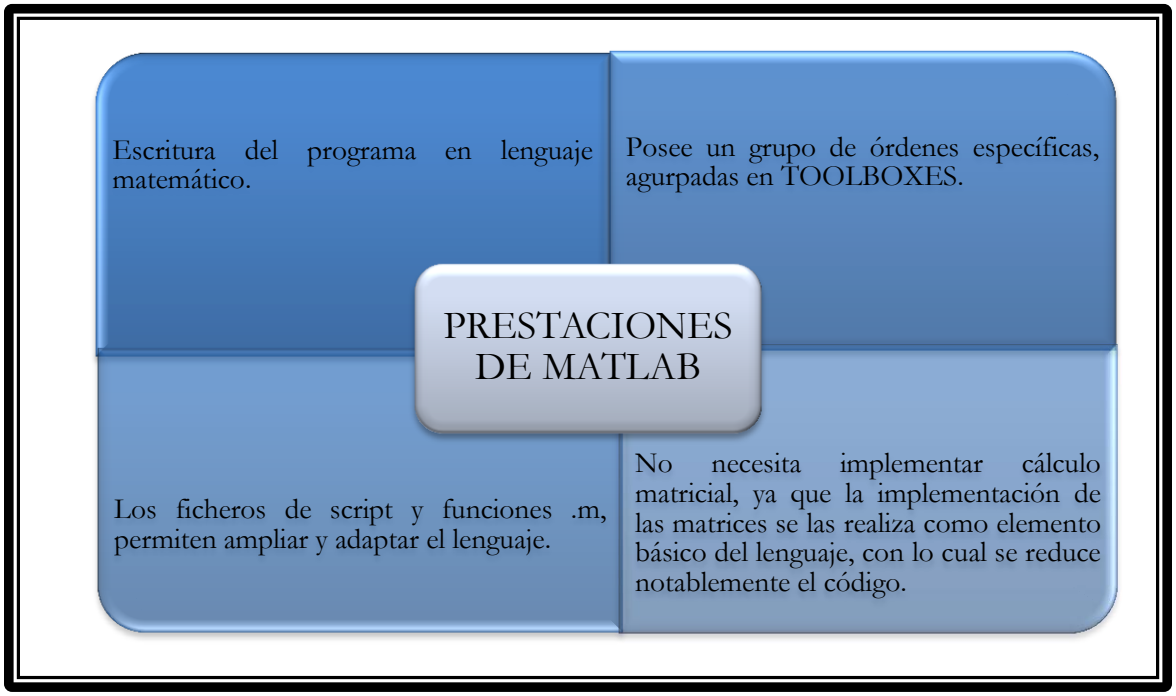
Figura II.20. Estructura While

2.4.2. MATLAB

Matlab (Matrix Laboratory) es un sistema interactivo, para su funcionamiento este software utiliza una matriz que no requiere de dimensionamiento. Fue especialmente diseñado para resolver problemas numéricos en fracciones de tiempo. Cuenta con una serie de soluciones específicas denominadas TOOLBOXES, las cuales sirven para resolver clases particulares de problemas como:

- Identificación de sistemas
- Diseño de sistemas de control
- Procesamiento de señales
- Redes neuronales

Debido a su capacidad de crecimiento Matlab permite al usuario convertirse en autor y contribuyente a la vez.



Fuente: http://nereida.deioc.ull.es/~pcgull/ihiu01/cdrom/matlab/contenido/node2.html

Figura II.21. Prestaciones de Matlab

2.4.3. OPENCV

OpenCV es una biblioteca libre de visión artificial, fue desarrollada específicamente para procesar imágenes en tiempo real, esta librería se creó con el objetivo de acelerar los procesos de visión, esto se logra utilizando algoritmos ya estudiados y optimizados directamente para el desarrollo de aplicaciones.

Esta librería posee una licencia BSD, que permite ser usada de forma gratuita para propósitos de investigación así como para propósitos comerciales.

Entre sus principales aplicaciones tenemos:

- Seguimiento del movimiento.
- Detección de características 2D y 3D.
- Segmentación y reconocimiento.

- Identificación de objetos.
- Visión estéreo y calibración.
- Estimación de movimiento.
- Reconocimiento facial y de gestos.
- Estructura del movimiento.

Este lenguaje fue originalmente escrito en C, debido a su mejoramiento en los últimos años, actualmente es posible trabajar con OpenCV en C++, lo cual facilita el trabajo y se evita problemas como la liberación de memoria.

OpenCV es compatible con otras plataformas de programación como Java, C#, Python o Ruby. Actualmente las últimas versiones cuentan con soporte GPU para aumentar la eficiencia del programa.

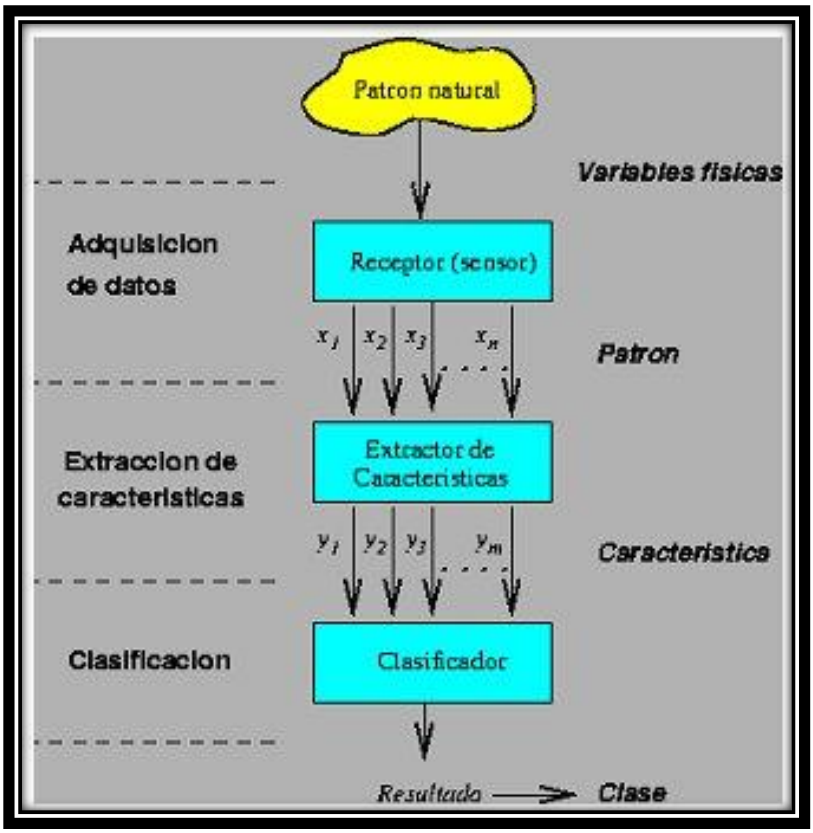
2.5. RECONOCIMIENTO DE PATRONES

El reconocimiento de patrones es una ciencia que se encarga de la clasificación y descripción de personas, objetos, señales, etc.

Con el reconocimiento de patrones se puede identificar características únicas que permiten diferenciar a un sujeto de otro.

El sistema de reconocimiento de patrones consta de tres fases o etapas:

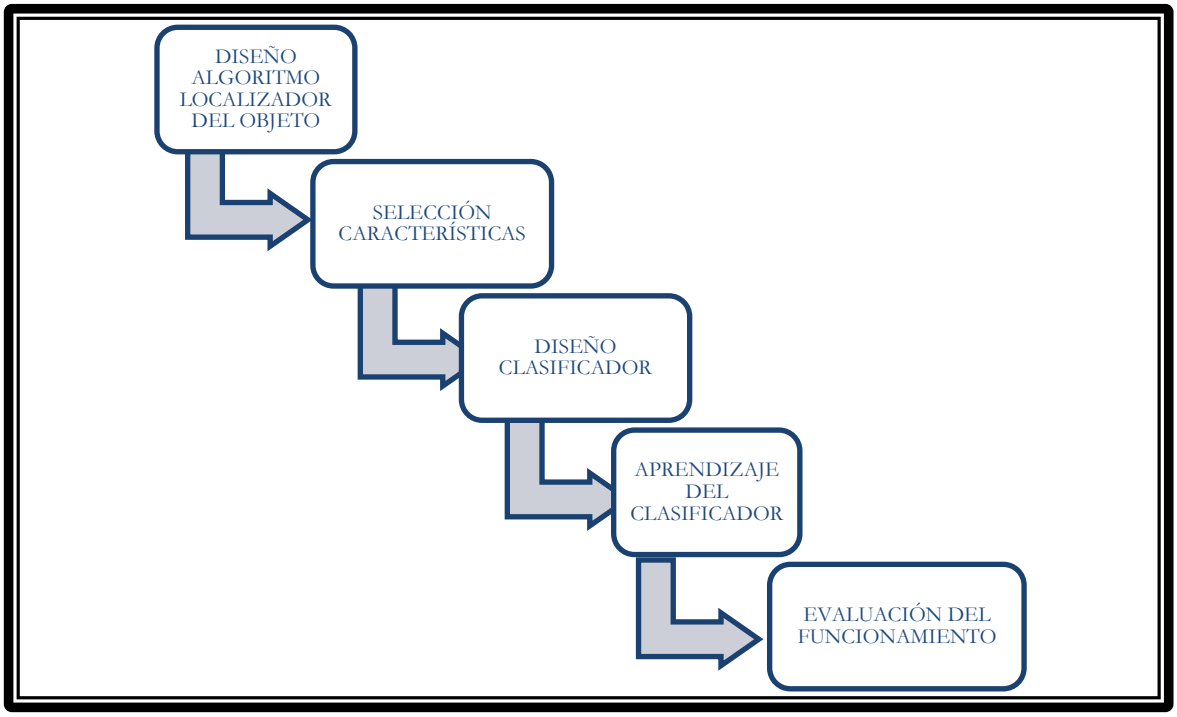
- Segmentación de la imagen.
- Extracción de las características.
- Clasificación en base a la distancia que se encuentra el objeto.



Fuente: http://www.sites.upiicsa.ipn.mx/polilibros/portal/Polilibros/P_terminados/Ing-del-Conoc/Ingenieria_conocimiento/reconocimiento_definicion.html

Figura II.22. Etapas de un sistema de reconocimiento de patrones

Para el diseño del sistema de reconocimiento de patrones se debe seguir una serie de pasos, los cuales se detallan en la siguiente figura.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura II.23. Diseño del sistema de reconocimiento de patrones

Al diseñar un sistema de reconocimiento de patrones (SRP), se pueden presentar los siguientes problemas:

- No se puede conocer totalmente el modelo.
- Restricciones económicas debido a la complejidad del SRP a diseñar.

METODOLOGÍAS PARA EL RECONOCIMIENTO DE PATRONES

Se puede decir que existen tres metodologías básicas que sirven para el reconocimiento de patrones, estas son las matemáticas, heurísticas y lingüísticas.

Matemáticas

Esta metodología se basa en reglas de clasificación formuladas en un marco matemático y hacen uso de las propiedades comunes de los patrones.

Se divide en dos categorías:

- Determinísticas: dentro de estos métodos podemos encontrar a los algoritmos de aprendizaje.
- Estadísticas: dentro de estos métodos están los clasificadores basados en las reglas de clasificación de Bayes.

Heurísticas

Este tipo de metodología se basa en la experiencia y la intuición humana.

Lingüísticas

Para esta metodología se usa los elementos primitivos que componen los patrones.

2.6. MÉTODOS DE APRENDIZAJE

Los sistemas de reconocimiento de patrones, en su gran mayoría son sistemas complejos, por lo cual no es posible realizar suposiciones sobre lo que sería una meta al diseñar el sistema de clasificación. Los métodos que incorporan información sobre un conjunto de entrenamiento en el diseño del clasificador, obligatoriamente deben emplear algún tipo de aprendizaje, razón por la cual se debe considerar nociones sobre aprendizaje.

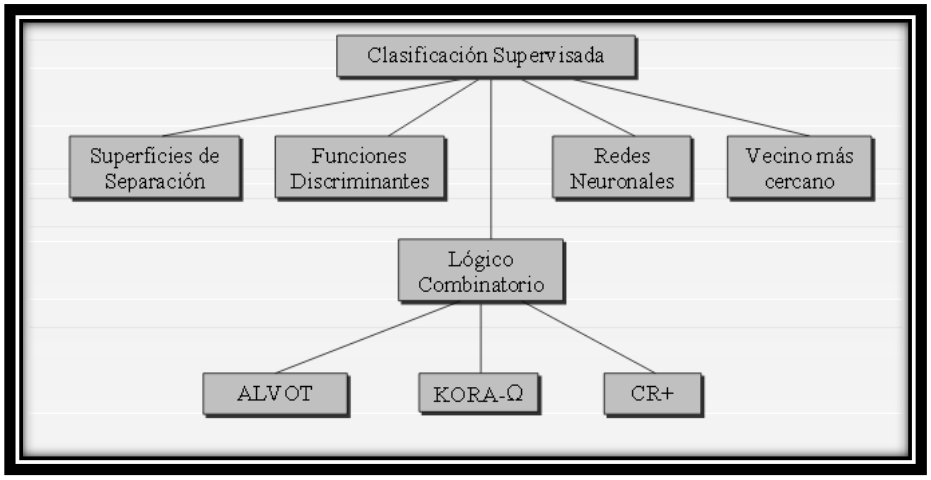
Para la construcción de clasificadores es primordial asignar algún tipo de generalidad para diseñar el modelo, forma de modelo o forma de un clasificador, por lo cual se aconseja establecer patrones de entrenamiento para despejar las incógnitas existentes en los patrones del modelo.

El aprendizaje se refiere a algún tipo de algoritmo que ayude a reducir la cantidad de errores que pueden existir en la información, la cual se utiliza para el entrenamiento[9].

TIPOS DE APRENDIZAJE

- **APRENDIZAJE SUPERVISADO**

  En un conjunto de entrenamiento el usuario es quien suministra una marca para cada patrón.



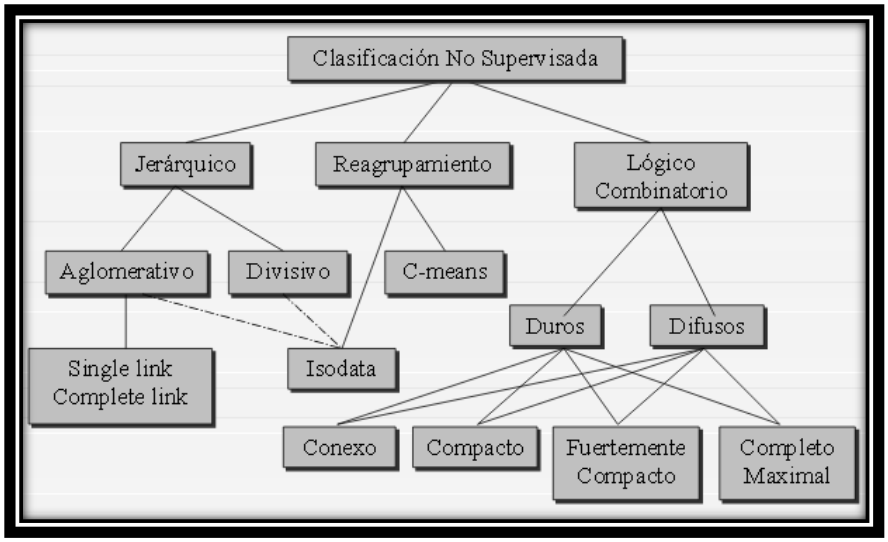
Fuente: http://ccc.inaoep.mx/~ariel/recpat.pdf

Figura II.24. Modelos de clasificación supervisada

- **APRENDIZAJE NO SUPERVISADO**

  En este caso se da una muestra no clasificada y se debe encontrar la clasificación de la misma.

[9] http://www.maginvent.org/articles/pidht/pidtoot/Reconocimiento_Patrones.html



Fuente: http://ccc.inaoep.mx/~ariel/recpat.pdf

Figura II.25. Tipos de clasificación no supervisada

- **APRENDIZAJE REFORZADO**

  También conocido como aprendizaje con la asistencia de un crítico o editor. Esto quiere decir que aunque no existe una categoría de referencia, en cambio la categoría asignada en un nuevo grupo de sugerencias puede interpretarse como correcta o incorrecta, aunque no se justifique ni el por qué ni el cómo de que esté equivocada[10].

2.6.1. COINCIDENCIA DE COLORES

El color es un eficaz descriptor que facilita la identificación de objetos y simplifica la extracción del mismo dentro de una escena.

La comparación de colores se trata de un proceso que requiere acceder al color de cada pixel que conforma la imagen.

[10] http://www.maginvent.org/articles/pidht/pidtoot/Reconocimiento_Patrones.html

- **PROCESAMIENTO DEL COLOR**

Se divide en dos casos que son:

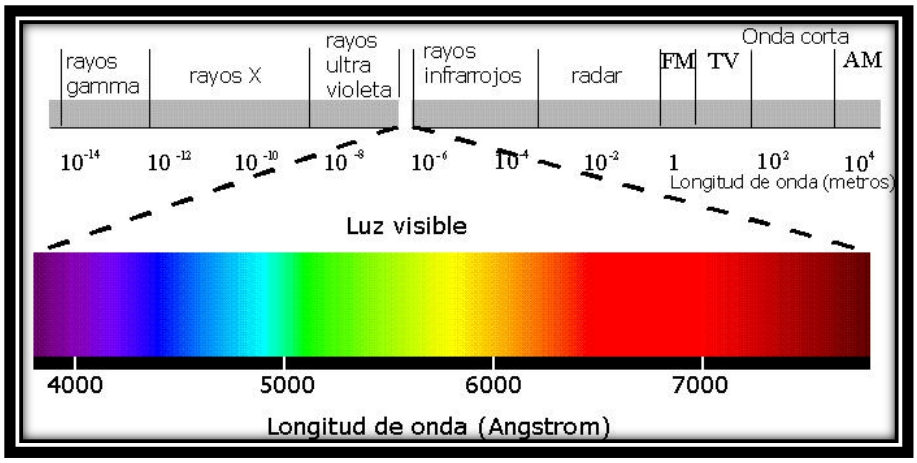
1. ***Procesamiento del color:*** se adquieren las imágenes con un sensor de color.

2. Procesamiento del seudo-color: se asigna un color a un rango de intensidades monocromas.

- **FUNDAMENTOS DEL COLOR**

Los colores que los seres humanos percibimos en un objeto se encuentran definidos por la naturaleza de la luz reflejada sobre dicho objeto.

Cuando el objeto refleja luz balanceada en todas las longitudes de luz visible se verá blanco, mientras que si la reflectancia favorece al objeto en un rango del espectro visible se podrá observar una gama de colores.



Fuente: http://reflexionesfotograficas.blogspot.com/2011/01/el-espectro-visible-los-colores.html

Figura II.26. Longitudes de onda del espectro visible

- **MODELOS DE COLOR**

Un modelo de color se trata de la especificación de un sistema de coordenadas donde se va a representar a cada color.

Los modelos más comunes orientados a hardware son:

- ✓ RGB (rojo, verde, azul), utilizados para cámaras de video.
- ✓ CMY (cian, magenta, amarillo) y CMYK, utilizados para impresiones a color.
- ✓ HSI (tono, saturación, intensidad), es el más similar al ojo humano para interpretar el color.

2.6.1.1. ALGORITMO BÁSICO DE COINCIDENCIA DE COLORES

Los algoritmos de coincidencia de colores tienen como objetivo identificar colores en una imagen cualquiera.

El algoritmo para reconocer un color específico en una imagen consta de las siguientes etapas:

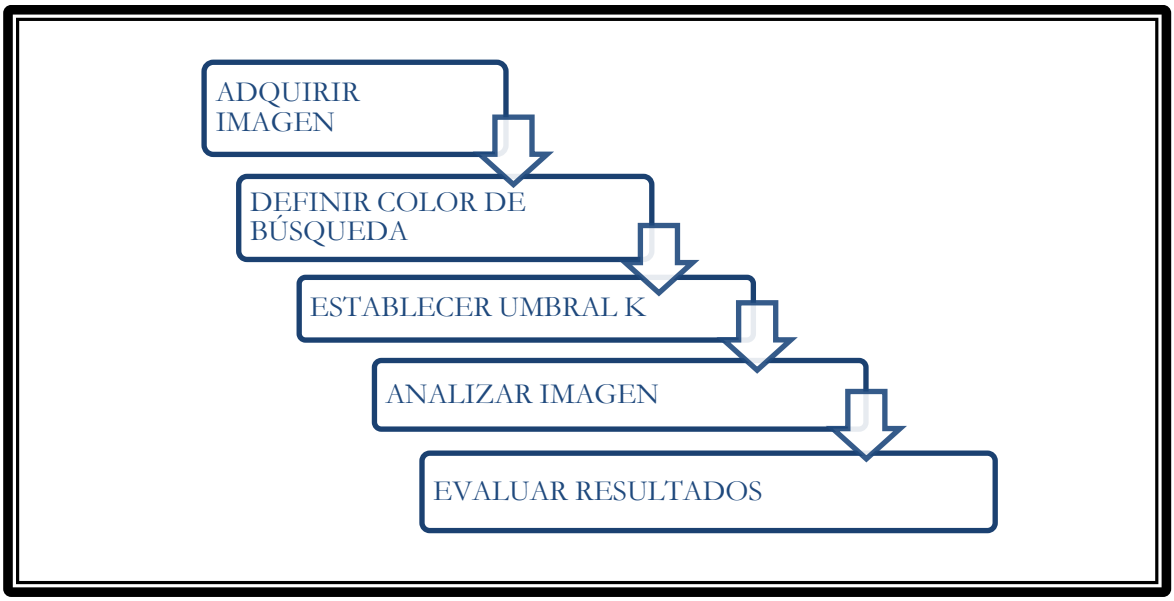
- ♣ Definir el color de interés C0 = (r0, g0, b0).
- ♣ Establecer el umbral K de la distancia entre los colores del mapa de bits y el color de interés. Es decir definir un rango de variación para el cual se pueda considerar que el color del píxel se aproxima al color de interés.
- ♣ Para cada píxel de la imagen se realiza el siguiente cálculo:

Se extrae el color C de cada píxel.

Se calcula la distancia D(C, C0).

Si la distancia obtenida entre C y C0 es menor que el umbral de comparación K, entonces se marca en blanco, de lo contrario se marca con negro[11].

En la presente figura se observa el esquema básico del algoritmo.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura II.27. Esquema algoritmo de coincidencia de colores

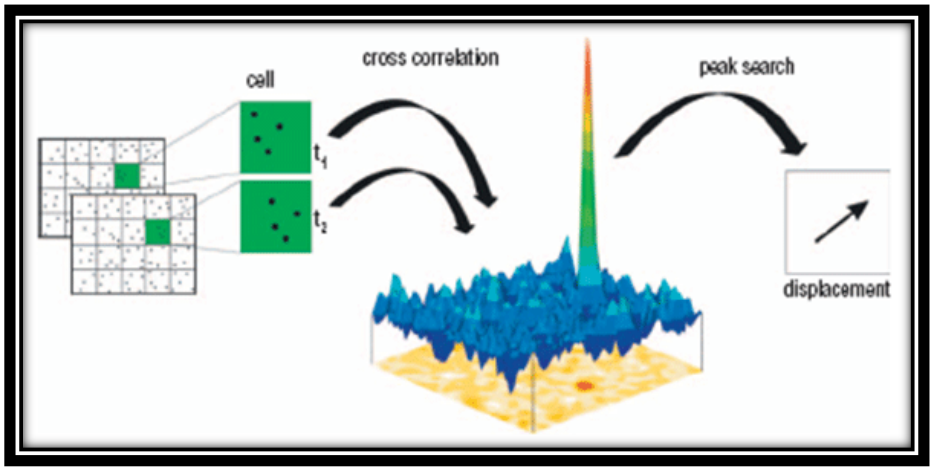
2.6.2. CORRELACIÓN CRUZADA

La correlación cruzada de imágenes tiene como objetivo localizar de manera automática un punto de estudio dentro de una imagen.

[11] Aplicación de visión con LabVIEW para la detección de frascos con turbiedades. (2009). ESPOL. Desde http://www.dspace.espol.edu.ec/bitstream/123456789/8103/1/Aplicacion%20de%20Vision%20con%20LabVIEW%20para%20la%20deteccion%20de%20frascos%20con%20turbiedades.pdf

La correlación también llamada *"matching"* explica el proceso de identificación automática de los puntos homólogos en imágenes digitales[12].

Este es uno de los métodos más común para el reconocimiento de patrones en una imagen, y debido a que requiere de un mayor número de procesos su tiempo de ejecución es elevado.



Fuente: http://www.analiticasa.com.ar/index_optica_sistemas_lavision.html

Figura II.28. Correlación digital de imágenes

2.6.2.1. ALGORITMO BÁSICO DE CORRELACIÓN CRUZADA

El algoritmo de correlación cruzada tiene como objetivo el cálculo de la similitud entre dos señales.

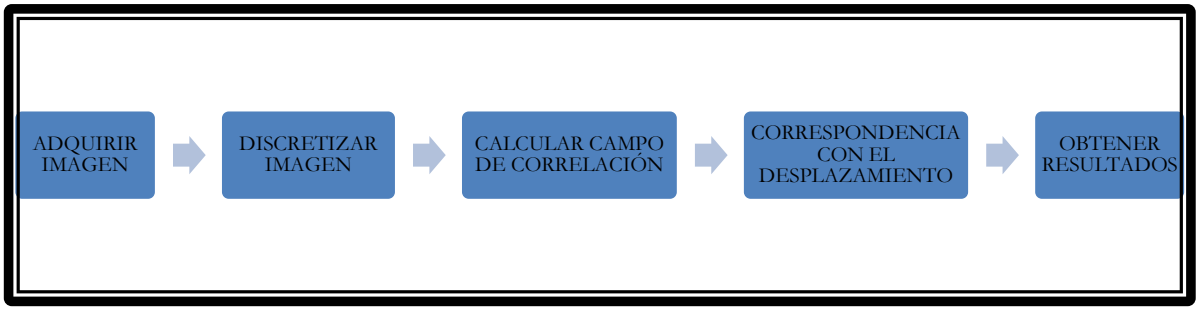
El proceso seguido por el algoritmo de correlación cruzada es el siguiente:

- Primero se obtiene una imagen por cuadrado de la cámara.

[12] Correlación y Orientaciones. (2008). UNSJ. Desde http://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCMQFjAA&url=ftp%3A%2F%2Fftp.unsj.edu.ar%2Fagrimensura%2FFotogrametria%2FUnidad6%2FTema_3_Fotogrametria_Digital_Correclacion_y_Orientaciones.doc&ei=aKqWU7vpGaXnsASftoD4BA&usg=AFQjCNFQIJ7mvjiNLxoCI5sRU2FJtcLVvw&bvm=bv.68445247,d.cWc

- Las imágenes obtenidas se discretizan en celdas más pequeñas, denominadas áreas de interrogación.
- Se calcula el campo de correlación para cada celda de la imagen, esto se lo puede realizar con la ayuda de la transformada rápida de Fourier (FFT).
- La correlación máxima se corresponde con el desplazamiento.
- Con el desplazamiento se obtiene la longitud del vector y la dirección de la ventana[13].

En la siguiente figura se presenta el esquema básico del algoritmo.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura II.29. Esquema algoritmo correlación cruzada

2.6.3. RECONOCIMIENTO DE CARACTERES (OCR)

El reconocimiento óptico de caracteres es un proceso que tiene como objetivo reconocer la parte textual de una imagen digitalizada. Dicho proceso identifica automáticamente símbolos o caracteres en una determinada imagen.

La imagen digitalizada es la entrada del OCR, dando como resultado un archivo de texto, el mismo puede ser usado para cualquier aplicación que lo necesite.

[13] http://www.analiticasa.com.ar/index_optica_sistemas_lavision.html

El reconocimiento óptico de caracteres trabaja con imágenes en una escala de grises, esto facilita el reconocimiento los caracteres, garantizando que los mismos van a ser reconocidos en su totalidad, y posteriormente poder crear el fichero de texto. Para realizar el reconocimiento de dichos caracteres, se realiza una comparación de los patrones o plantillas que contendrán todos los posibles caracteres con cada uno de los caracteres que compone la imagen.

Los problemas que se pueden presentar en el reconocimiento óptico de caracteres son los siguientes:

- Ruido en la imagen.
- La separación que existe entre caracteres.
- Al captar la imagen se pueden producir distorsiones, introduciendo niveles de grises que no pertenecen a la imagen real.
- La resolución de los dispositivos.
- La posición de la página en el escáner.

2.6.3.1. ALGORITMO BÁSICO DE RECONOCIMIENTO ÓPTICO DE CARACTERES (OCR)

Los algoritmos de reconocimiento óptico de caracteres tienen como objetivo diferenciar texto en una imagen cualquiera.

En todo sistema OCR se distinguen cuatro etapas:

- ♣ Binarización
- ♣ Segmentación
- ♣ Adelgazamiento de las componentes
- ♣ Comparación con patrones

A continuación se detallan cada una de las etapas.

1. Binarización

Para trabajar con el algoritmo OCR se debe partir de una imagen binaria, es decir una imagen de dos colores. Cualquier imagen ya sea a color o en escala de grises, puede ser binarizada, esto quiere decir que se puede convertir en una imagen de blanco y negro, ya que en este tipo de imágenes es más fácil resaltar los símbolos y contornos de caracteres y de esta manera aislar las partes de la imagen donde haya texto.

Fuente: http://multitouchvisual.blogspot.com/2010/09/tratando-imagenes.html

Figura II.30. Binarización de la imagen

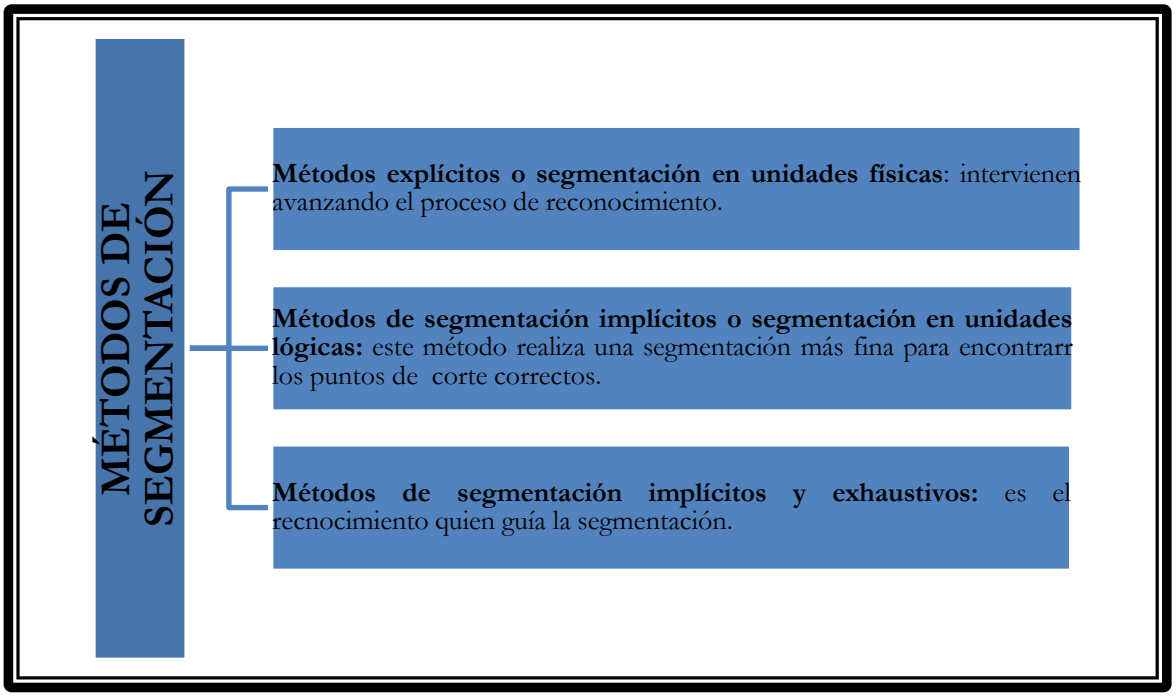
2. *Segmentación*

La segmentación tiene como objetivo detectar los contornos de los símbolos o caracteres de la imagen, basándose en la información de intensidad de la misma.

Además la segmentación permite descomponer el texto en diferentes entidades lógicas, dichas entidades deben ser lo suficientemente invariables, ya que con ello no se depende del estilo de escritura, y también deben ser lo suficientemente significativas para facilitar el reconocimiento.

Con la segmentación se logra ubicar las zonas de interés y separar la imagen por dichas zonas, estas zonas de interés tienen una serie de atributos como inclinación, dimensión, densidad, longitud del trazo, superficie, etc.

En base al grado de asociación entre las operaciones de reconocimiento y las de segmentación, se pueden distinguir tres tipos de métodos para la segmentación.



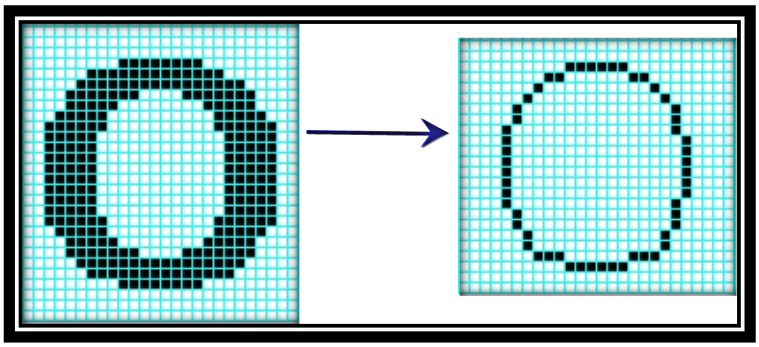
Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura II.31. Métodos de Segmentación

3. Adelgazamiento de las componentes

Después de detectar los contornos de los caracteres, se realiza el procedimiento de adelgazamiento de las componentes, el cual consiste en ir borrando los puntos de los contornos de forma, de manera que se mantenga su tipología.

Para el borrado de puntos se debe seguir un esquema de barridos sucesivos, esto se lo realiza para que la imagen no se deforme a lo largo del proceso y conserve las mismas proporciones que la original. Se debe hacer un barrido en paralelo, es decir, señalar los píxeles que se van a borrar, para eliminarlos todos a la vez.



Fuente: http://grupo.us.es/gtocoma/pid/pid10/OCR.htm

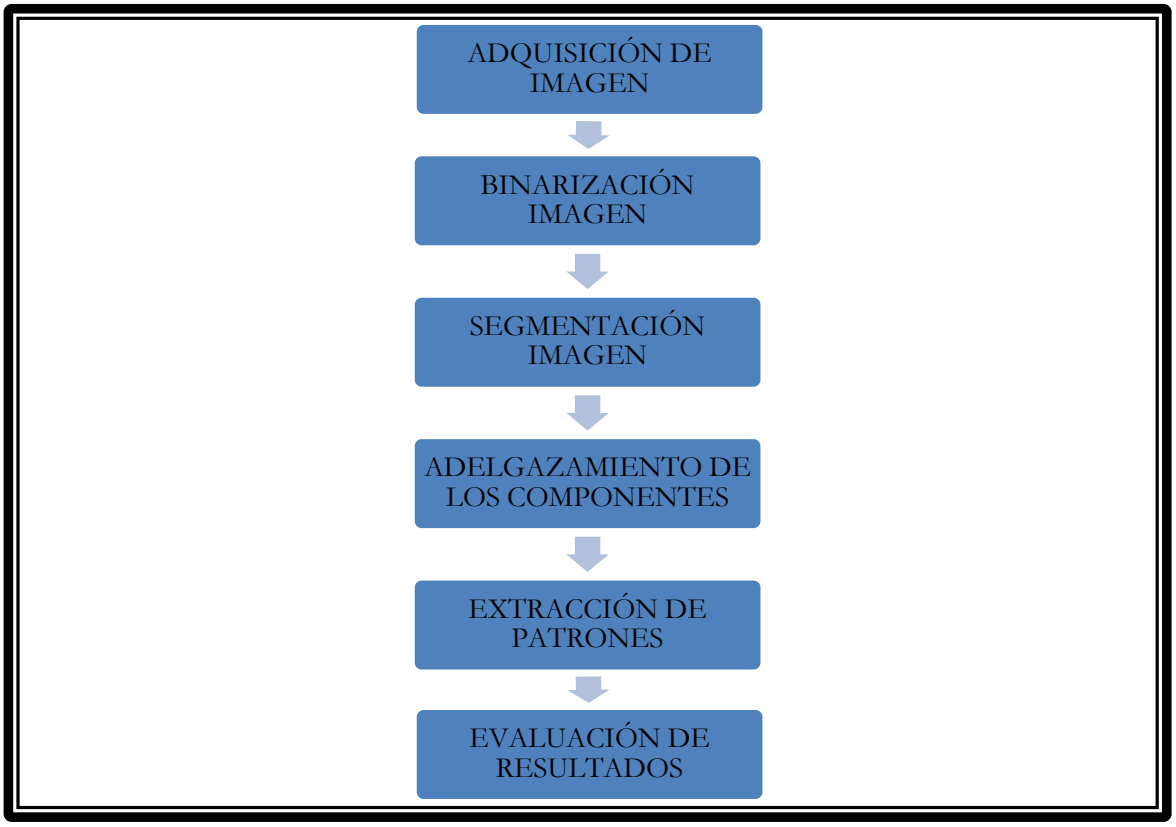
Figura II.32. Adelgazamiento de imágenes

4. Comparación con patrones

En esta etapa se comparan los caracteres obtenidos en el adelgazamiento de las componentes, con patrones o plantillas, los mismos que son almacenados en una base de datos.

Para un buen funcionamiento del OCR, se debe asegurar éxito en esta etapa.

En la presente figura se muestra el esquema del algoritmo.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura II.33. Esquema algoritmo OCR

2.6.3.2. ALGORITMO KNN

KNN (K vecinos más cercanos), es un método de clasificación supervisada, en reconocimiento de patrones es utilizado para clasificar elementos en un conjunto de prototipos cercanos al elemento de búsqueda.

Para el reconocimiento de patrones se debe tener en cuenta que los patrones son regiones de imágenes.

El algoritmo KNN sigue el siguiente proceso:

- Primero se debe convertir la región de imágenes en un vector de características, dicho vector es el prototipo o patrón.

- Clasificar manualmente los prototipos y definir una clase para cada uno de ellos.

- Crear un conjunto de referencia (de tamaño N), esto se refiere a almacenar en una tabla los prototipos clasificados anteriormente.

- Para clasificar un nuevo prototipo se debe calcular la distancia euclídea a los N prototipos del conjunto de referencia.

- Considerar los k vecinos más cercanos que existen en el conjunto de referencia.

- Contabilizar las clases a las que pertenecen los k prototipos, con lo cual el nuevo patrón se clasifica con la clase mayoritaria.

2.6.4. CUADRO COMPARATIVO DE ALGORITMOS

En la tabla II.2 se realiza un breve resumen comparativo de los algoritmos que se pueden emplear para el reconocimiento de patrones.

Tabla II.2. Cuadro comparativo de algoritmos

ALGORITMO COINCIDENCIA DE COLORES	ALGORITMO CORRELACIÓN CRUZADA	ALGORITMO RECONOCIEMIENTO ÓPTICO DE CARACTERES (OCR)
- Trabaja con imágenes RGB (Red, Green, Blue). - El reconocimiento	- Trabaja con imágenes binarias (imágenes en blanco y negro). - El reconocimiento	- Trabaja con imágenes de intensidad (imágenes en escala de grises). - El reconocimiento

de los patrones se realiza en base al color. - Posee un alto porcentaje de confiabilidad en los resultados. - Los resultados varían dependiendo de la fuente de luz y de la distancia a la que se encuentre ubicado el dispositivo para la captura de la imagen.	de los patrones se realiza en base a formas. - Los resultados obtenidos son poco confiables. - Los resultados varían dependiendo de la fuente de luz y de la distancia a la que se encuentre ubicado el dispositivo para la captura de la imagen.	de los patrones se realiza en base a caracteres. - Existe una gran probabilidad de error en los resultados obtenidos. - Los resultados varían dependiendo de la fuente de luz y de la distancia a la que se encuentre ubicado el dispositivo para la captura de la imagen.

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

2.7. FPGAs

FPGAs (Arreglos de compuertas programables en campo), son chips de silicio reprogramables. Para programar una FPGA no es necesario desarrollar una aplicación de software, ya que esta reconecta el propio chip para implementar su funcionalidad.

La National Instruments (NI) conjuntamente con Xilinx, ofrecen tecnología FPGA de última generación, la misma es compatible con una variedad de plataformas de hardware.

2.7.1. HISTORIA

Ross Freeman, el cofundador de Xilinx, inventó la primera FPGA en 1985. El primer modelo diseñado fue el XC2064, este tenía puertas programables e interconexiones entre las pruebas, con lo cual se marcó el comienzo de una nueva tecnología para el mercado. Debido al éxito que tuvo y a su rápido crecimiento, para el año de 1993 Actel ya abarcaba el 18% del mercado.

En la década de 1990 las FPGAs tuvieron un gran avance, estas contaban con más sofisticación, motivo por el cual se aumentó el volumen de producción. En dicha década las FPGAs ya abarcaban varios campos de aplicación, especialmente se utilizaban en telecomunicaciones y redes. A finales de la década ya eran aplicadas en otros campos como la automotriz e industrial.

2.7.2. LENGUAJES DE PROGRAMACIÓN

El programador cuenta con la ayuda de ambientes de desarrollo especializados para el diseño de sistemas que serán implementados en una FPGA. Un diseño puede ser desarrollado de dos maneras, con un diagrama esquemático, o utilizando lenguaje de programación especial.

VHDL, ABEL, LabView y Verilog son los más utilizados dentro de los lenguajes de programación especializados, dichos lenguajes se los conoce como HDL (Hardware Description Language).

VHDL

Es un lenguaje de alto nivel, utilizado para la descripción del hardware de los sistemas digitales, su acrónimo representa la combinación

de VHSIC (Very High Speed Integrated Circuit) y HDL, fue definido por el IEEE (ANSI/IEEE 1076-1993).

En la actualidad este programa aumentado notablemente su número de usuarios, esto se debe a la popularidad que adquirido.

VERILOG

También llamado Verilog HDL, se creó para soportar el diseño, prueba e implementación de circuitos analógicos y digitales, así como también los de señal mixta de diferentes niveles de abstracción[14].

Este lenguaje cuenta con una serie de características, entre las principales tenemos:

- Verilog es un lenguaje más compacto que VHDL.
- Su programación es similar al lenguaje C.
- No necesita de estructuras, apuntadores o funciones recursivas.
- No es necesario establecer tiempo, ya que este término no se encuentra definido en el lenguaje HDL.
- La ejecución de las sentencias en Verilog se lo realiza en base a una jerarquía de módulos.

ABEL

Este lenguaje fue creado para programar dispositivos lógicos programables (PLDs), cuenta con un conjunto de herramientas de diseño, que sirve para

[14] http://reinaldo-noelia-fpga.blogspot.com/2007/12/fpga_4670.html

la descripción de hardware. ABEL es el acrónico de Advanced Boolean Expression Language.

Características

- Para los diseños desarrollados este lenguaje utiliza tablas de verdad o ecuaciones lógicas.
- Para la programación secuencial utiliza máquina de estados.
- Permite definir patrones de entradas y salidas, así como también vectores de prueba, éstos pueden ser programados en el hardware.

LABVIEW

La plataforma de programación gráfica LabView actualmente cuenta con un módulo para desarrollo de proyectos en FPGAs, lo cual facilita la programación e implementación de las distintas aplicaciones.

Antes para programar un PLDs se utilizaban lenguajes de programación especializados. Para diseñar circuitos otros métodos utilizados son la captura de esquemas con herramientas CAD y los diagramas de bloques, sin embargo estos métodos no son aconsejables para diseños complejos.

2.7.3. TIPOS DE FPGAs

Actualmente existe una gran variedad de fabricantes de FPGAs, existen competidores que las fabrican de propósito general, mientras que otros optan por fabricarlas con capacidades únicas.

2.7.3.1. XILINX

Entre las principales series de FPGA Xilinx tenemos:

- **Series Virtex:** Esta serie fue creada para reemplazar ASICs en múltiples aplicaciones.

  Entre las principales aplicaciones en las que se utiliza Virtex tenemos:

  - Redes alámbricas e inalámbricas
  - Servidores
  - Telecomunicaciones
  - Imagen
  - Almacenamiento
  - Computación
  - Video

  Dentro de las versiones fabricadas tenemos FPGA Virtex-5, Virtex-4, Virtex-II Pro, Virtex-II, y Virtex-E[15].

- **Series Spartan:** en la presente tabla se indican las características de los dispositivos fabricados con esta serie.

Tabla II.3. Características dispositivos series Spartan

CARACTERÍSTICAS	
Compuertas	5 millones
Puertos entrada/salida	784

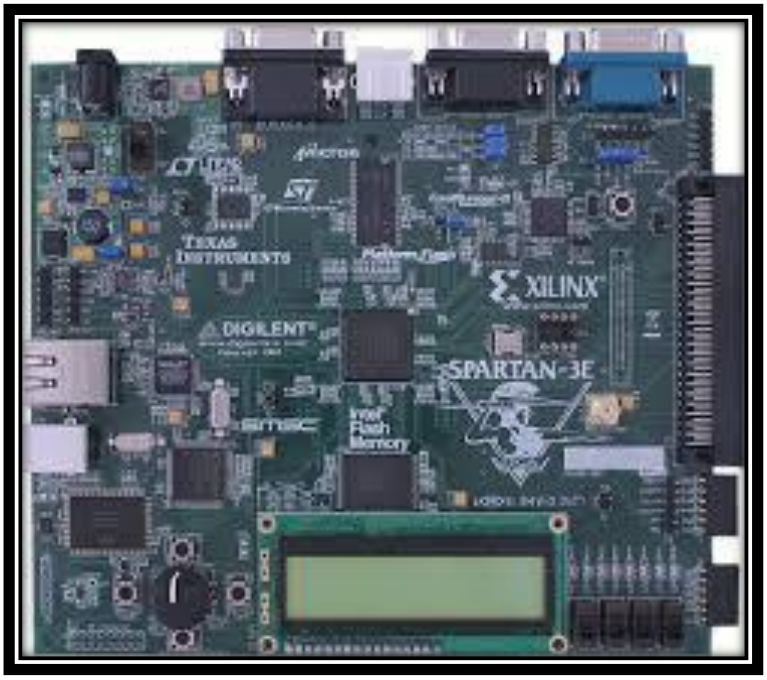
[15] http://reinaldo-noelia-fpga.blogspot.com/2007/12/fpga_4670.html

Entrada/salida diferencial	344 pares
Tecnología	XCITE

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Dentro de las familias de FPGA para esta serie tenemos: Spartan-3A DSP, Spartan-3AN, Spartan-3A, Spartan-3E, Spartan-3, Spartan-IIE, Spartan-II, Spartan/XL.

Entre las aplicaciones para la familia de esta serie podemos destacar el procesamiento digital de señales y memoria no volátil.



Fuente: http://digilentinc.com/Products/Detail.cfm?Prod=S3EBOARD

Figura II.34. Xilinx Spartan-3E

2.7.3.2. ALTERA

- **Serie Cyclone**

CARACTERÍSTICAS

- Poseen hasta 120 mil elementos lógicos
- 535 pines de entrada/salida
- Baja potencia
- Alta funcionalidad
- Bajo costo

FAMILIAS

- Cyclone III
- Cyclone II
- Cyclone

APLICACIONES

- Automotriz
- Despliegue y procesamiento de imágenes

- **Serie Stratix**

CARACTERÍSTICAS

- Poseen hasta 79 mil compuertas
- 1200 pines de entrada/salida
- 7 Mbits de memoria RAM
- 22 bloques de DSP
- 176 multiplexores embebidos

FAMILIAS

- Stratix III
- Stratix II GX
- Stratix II

APLICACIONES

- Aeroespaciales
- Militares

Fuente: http://www.altera.com/corporate/news_room/releases/2009/products/nr-ciii_ls_dev_kit.html

Figura II.35. Altera Cyclone III

2.7.3.3. LATTICE

Entre las principales series de este fabricante se encuentran:

- **Serie SC**

  CARACTERÍSTICAS

  - Equipada con la memoria embebida
  - Tiene hasta 12 bloques ASIC embebidos por dispositivo

- **Serie XP**

  CARACTERÍSTICAS

  - Sus dispositivos utilizan la combinación de celdas FLASH no volátiles.
  - Utiliza la tecnología SRAM.

- **Serie XP2**

  Esta serie cuenta con dispositivos que combinan un Look up Table (LUT), basado en la fabricación de FPGA con celdas de Flash no volátil en una arquitectura denominada flexiFLASH[16].

- **Serie ECP2**

  Los dispositivos de esta serie cuentan con las mismas características y capacidades que las FPGAs de alto costo.

[16] http://reinaldo-noelia-fpga.blogspot.com/2007/12/fpga_4670.html

2.7.3.4. ACTEL

Este fabricante ofrece un tipo de FPGA que se basa en la tecnología Flash Reprogramable.

Otro tipo de FPGA incluye mezcladores de señales basados en Flash.

Fuente: http://www.eetimes.com/document.asp?doc_id=1316281

Figura II.36. Actel flash process

2.7.3.5. QUICKLOGIC

Dentro de los dispositivos fabricados por Quicklogic se destaca el PolarPro tecnología FPGA, este fue diseñado con el propósito de satisfacer la lógica del sistema de interconexión y las aplicaciones portátiles.

Este dispositivo cuenta con una nueva e innovadora arquitectura lógica de celdas, posee memoria embebida y control de la lógica avanzada del reloj de control de la gestión de unidades.

La arquitectura PolarPro es de síntesis fácil, de modo que el costo es ultra efectivo para los nuevos diseños[17].

[17] http://reinaldo-noelia-fpga.blogspot.com/2007/12/fpga_4670.html

Fuente: http://www.eetimes.com/document.asp?doc_id=1301922

Figura II.37. QuickLogic – PolarPro

2.7.4. USO ACTUAL DE LAS FPGAs

Las FPGAs tienen múltiples aplicaciones, actualmente las mismas se encuentran orientadas al procesamiento digital de señales, procesado de datos, entre otras.

A continuación se detallan las principales aplicaciones en las que se utilizan las FPGAs.

- **Sistemas de visión artificial:** dentro del campo de la visión artificial puede ser utilizado en el desarrollo de programas que sirven para el reconocimiento de rostros, reconocimiento y determinación de la posición de los objetos de un entorno, etc.

- **Sistemas de imágenes médicas:** se están empleando las FPGAs para el tratamiento de imágenes biomédicas obtenidas mediante procesos de PET, escáner CT, rayos X, imágenes tridimensionales, etc[18].

[18] http://www.generatecnologias.es/aplicaciones_fpga.html

- **Codificación y encriptación:** para llevar a cabo el proceso se encriptación de mensajes se utiliza las FPGAs, ya que esta aporta seguridad al sistema.

- **Reconocimiento de voz:** la FPGA facilita el proceso para el reconocimiento de voz, se realiza la comparación de la voz de una persona con patrones previamente almacenados en una base de datos. Esta técnica es empleada en seguridad, especialmente en los sistemas de recuperación de información.

2.8. ARDUINO

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar[19].

Fuente: http://arduino.cc/en/Main/arduinoBoardUno

Figura II.38. Arduino Uno

[19] http://www.arduino.cc/es/

El Arduino UNO se basa en el microprocesador Atmega328, esta placa electrónica cuenta con los siguientes elementos:

- 14 pines digitales de entrada / salida (6 de ellos pueden ser utilizados como salidas PWM).
- 6 entradas analógicas
- Un resonador cerámico 16 MHz
- Una conexión USB
- Un conector de alimentación
- Un header ICSP
- Un botón de reinicio

CARACTERÍSTICAS

Tabla II.4. Características Arduino Uno

MICROCONTROLADORES	ATMEGA328
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límites)	6-20V
Digital I / O Pins	14 (de los cuales 6 proporcionan PWM)
Pines de entrada analógica	6
Corriente continua para las E / S Pin	40 mA
Corriente de la CC para Pin 3.3V	50 mA
Memoria Flash	32 KB (ATmega328) de los cuales 0,5 KB utilizado por el gestor de arranque
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)

Velocidad del reloj	16 MHz

Fuente: http://arduino.cc/en/Main/arduinoBoardUno

El arduino UNO se puede alimentar de dos maneras, mediante conexión USB o con una fuente de alimentación externa.

La programación se la realiza a través de un programa gratuito que se puede descargar de la página web de Arduino, a través de esta placa electrónica se transfiere el programa que se desarrolla desde el ordenador a la placa, estos programas utilizan un lenguaje de programación propio de Arduino basado en Wiring.

Arduino puede ser utilizado con otros programas, como por ejemplo, simulink de Matlab, LabView proporciona un sketch que interacciona con la tarjeta y la vuelve una DAG para LabView.

CAPÍTULO III

SELECCIÓN DEL ALGORITMO MÁS ÓPTIMO PARA EL RECONOCIMIENTO DE PATRONES

3.1. INTRODUCCIÓN

En este capítulo se seleccionará el algoritmo más óptimo para el reconocimiento de patrones, los algoritmos propuestos para la selección son:

- Algoritmo de Coincidencia de Colores
- Algoritmo de Correlación Cruzada
- Algoritmo de Reconocimiento Óptico de Caracteres (OCR)

Los algoritmos mencionados anteriormente son los más utilizados en el campo de la visión artificial en base a estudios investigados, dentro de los cuales podemos mencionar:

- UNICEN, PLADEMA-ISISTAN, "PROCESAMIENTO DE IMÁGENES PARA LA CLASIFICACIÓN MASIVA DE FRUTOS BASADO EN EL COLOR", realizado por D'Amato, Juan Pablo; García Bauza, Cristian; Vénere, Marcelo; Clausse, Alejandro.

- Escuela Superior Politécnica de Litoral (ESPOL), "SISTEMA DE CONTROL VEHICULAR UTILIZANDO RECONOCIMIENTO ÓPTICO DE CARACTERES", realizado por Andrade Miranda Gustavo, López Encalada José, Chávez Burbano Patricia.

- Universidad Carlos III, "RECONOCIMIENTO ÓPTICO DE CARACTERES (OCR)", realizado por Carlos Javier Sánchez Fernández, Víctor Sandonís Consuegra.

Para comprobar la hipótesis, se utilizó el método estadístico completamente al azar (DCA), el cual selecciona el algoritmo más óptimo para la implementación de un prototipo de semáforo inteligente.

3.2. MÉTODOS UTILIZADOS

Para el análisis de resultados se utilizó el Método Estadístico de Diseño Completamente al Azar, en el cual el autor asigna las unidades experimentales a los tratamientos al azar, su única limitación es que el número de observaciones recolectadas en cada tratamiento deben ser iguales y mínimo 10. Este método es el más sencillo y se origina por la asignación aleatoria de tratamientos a un conjunto de unidades experimentales[20].

El DCA se utiliza cuando las condiciones del lugar donde se va a llevar a cabo el experimento son totalmente homogéneas.

[20] http://reyesestadistica.blogspot.com/2011/07/diseno-de-experimentos-al-completo-azar.html

Los pasos que sigue este método para realizar el estudio comparativo son los siguientes:

a. Establecer el número de unidades experimentales (n) y numerarlas. Para obtener “n” se multiplica el número de tratamientos por el número de repeticiones.

$$n = (t * r)$$

b. Realizar el modelo estadístico, para lo cual se utiliza la siguiente fórmula:

$$Y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

Dónde:

Y_{ij} es la variable de respuesta de la ij - ésima unidad experimental.

μ es el efecto de la media general.

α_i es el efecto del i - ésimo tratamiento.

ε_{ij} es el efecto del error experimental asociado a la i - ésima unidad experimental.

c. Análisis de varianza

En este paso se establece la hipótesis nula y la hipótesis alterna, ya que en base a los resultados obtenidos en el análisis de datos una de las dos hipótesis se rechaza.

d. Fuentes de variación y grados de libertad

Para este análisis se construye la tabla respectiva de análisis de varianza y se completan los datos.

Tabla III.5. Tabla de análisis de varianza

FUENTES DE VARIACIÓN	GRADOS DE LIBERTAD	SUMA DE CUADRADOS	CUADRADOS MEDIOS	Fc CALCULADA	Ft REQUERIDA
Tratamientos	$t-1$	$\Sigma\left[\left(Y^2../rY^{2..}\right)/(tr)\right]$	$Sctrat/gltotal$	$Scmedios/Cmerror$	
Error	$t(r-1)$	$Sctotal-Sctrat$	$Scerror/glerror$		
Total	$tr-1$	$\Sigma\Sigma\left[\left(Yij-Y^2..\right)/(tr)\right]$			

Fuente: http://reyesestadistica.blogspot.com/2011/07/diseno-de-experimentos-al-completo-azar.html

Los grados de libertad son uno menos que el número de observaciones para cada fuente de variación, existen diferentes grados de libertad, a continuación se detalla las fórmulas utilizadas para cada caso.

- Grados de libertad de tratamientos

$$(Gltrat)=t-1$$

- Grados de libertad total

$$(Gltot)=r*t-1$$

- Grados de libertad del error

$$(Gle)=Gltot-Gltrat$$

e. Cuadro de ANDEVA Ó ANOVA

Esta tabla nos permite rechazar o no la hipótesis nula o la hipótesis alternativa.

f. Interpretación de datos

❖ **POBLACIÓN Y MUESTRA**

Población: para la presente investigación, la población son los tres algoritmos propuestos.

Muestra: para la selección del algoritmo más óptimo para el reconocimiento de patrones la muestra estuvo formada por dos parámetros que son:

- **PRECISIÓN**

  La precisión se considera al porcentaje equivalente al número de aciertos obtenidos por el algoritmo al realizar el reconocimiento de los patrones.

  Para realizar un reconocimiento idóneo de patrones es necesario probar el éxito de cada uno de los algoritmos, para lo cual consideramos que el 100% corresponde al número de patrones que fueron tomados como muestra.

$$\%\text{ de Precisión} = \frac{\text{Número de patrones reconocidos}}{\text{Total de patrones evaluados}} x100\%$$

- **TIEMPO DE RESPUESTA**

  El tiempo de respuesta se refiere al tiempo que tarda el algoritmo en reconocer los patrones.

3.3. DESARROLLO DE LAS APLICACIONES

Para seleccionar el algoritmo más adecuado se desarrolló una aplicación en LabView para cada algoritmo propuesto, dicha aplicación tiene como objetivo reconocer el patrón de búsqueda.

El esquema diseñado para el desarrollo de cada una de las aplicaciones es el siguiente:

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.39. Esquema de las aplicaciones

- ***Adquisición de la imagen:*** mediante una cámara se obtiene la imagen a ser procesada.

- ***Algoritmo/ Procesamiento de la imagen:*** en esta etapa el algoritmo procesa la imagen adquirida para su posterior análisis.

- ***Reconocimiento de patrones:*** realiza el análisis cuantitativo y cualitativo de la imagen para el reconocimiento de patrones, este puede ser a nivel de píxel, bordes, color, carácter, etc.

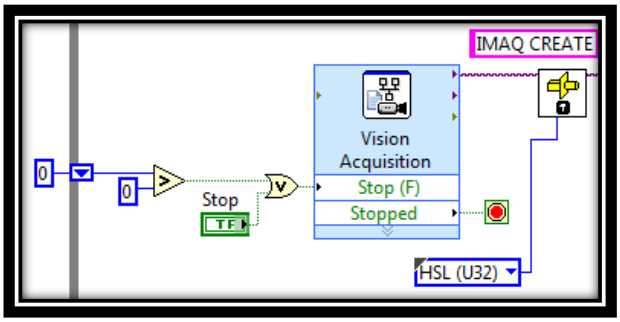
3.3.1. PRUEBA ALGORITMO DE COINCIDENCIA DE COLORES

El algoritmo de coincidencia de colores tiene como objetivo reconocer patrones en función al color, dicho patrón es establecido previamente por el usuario.

A continuación se describe la aplicación desarrollada para dicho algoritmo.

- ♣ Como primer paso abrimos la cámara.

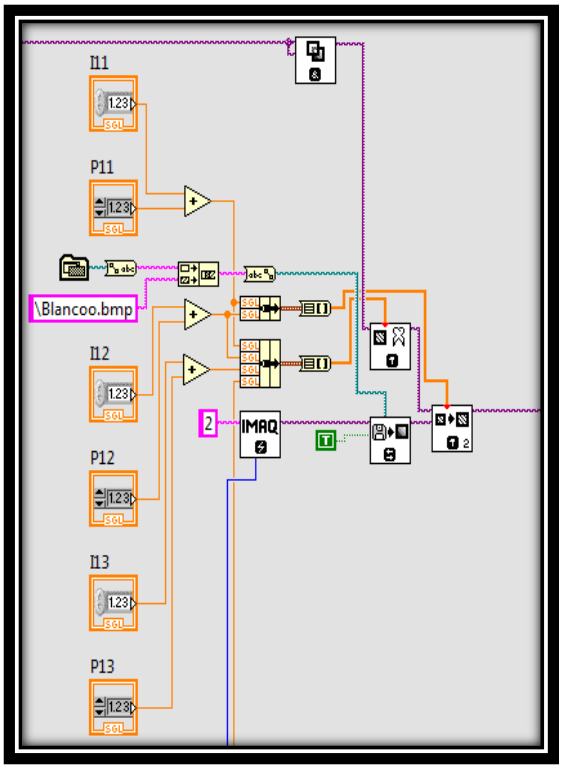
- Seguidamente seleccionamos el módulo Vision Acquisition como se puede observar en la figura III.40, el mismo tiene como función adquirir la imagen, y con el módulo IMAQ create seleccionamos el tipo de imagen.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.40. Módulo Vision Acquisition

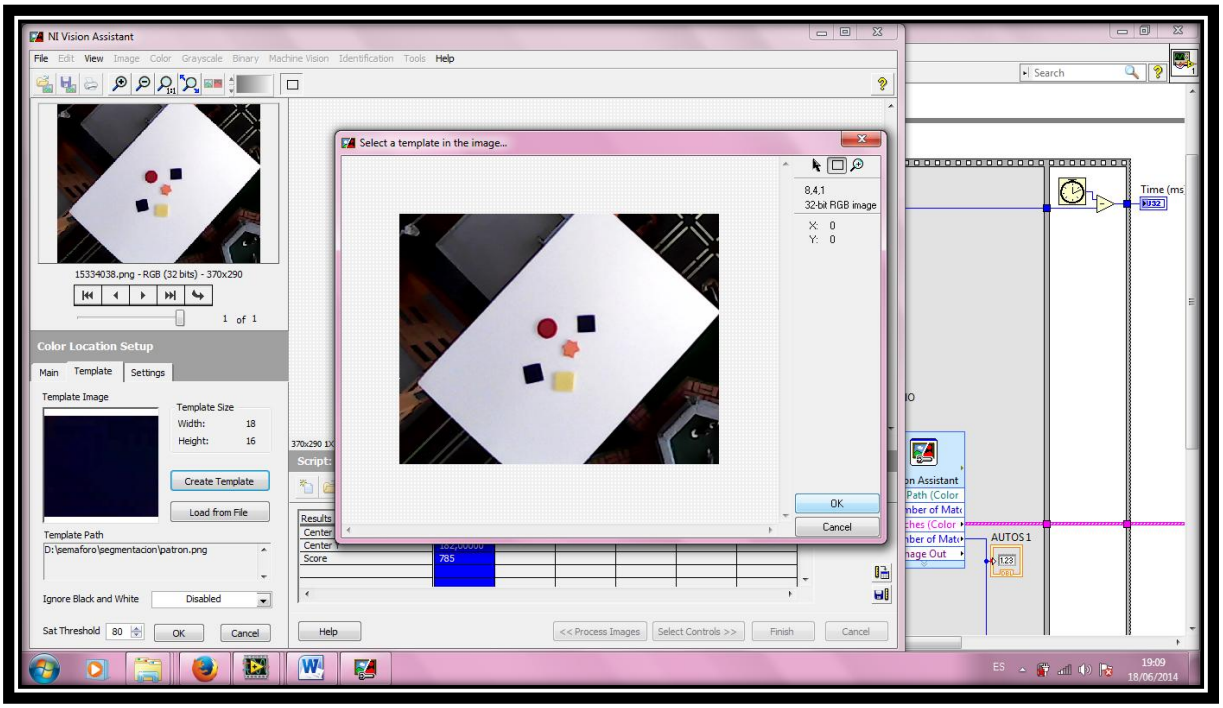
- Se procede a segmentar la imagen adquirida, como se observa en la figura III.41 para poder definir una región de interés.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.41. Segmentación de la imagen

- Se procede a cargar el patrón, para lo cual se utiliza la herramienta Select Template Image del módulo Vision Assistant, tal como se observa en la figura III.42.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

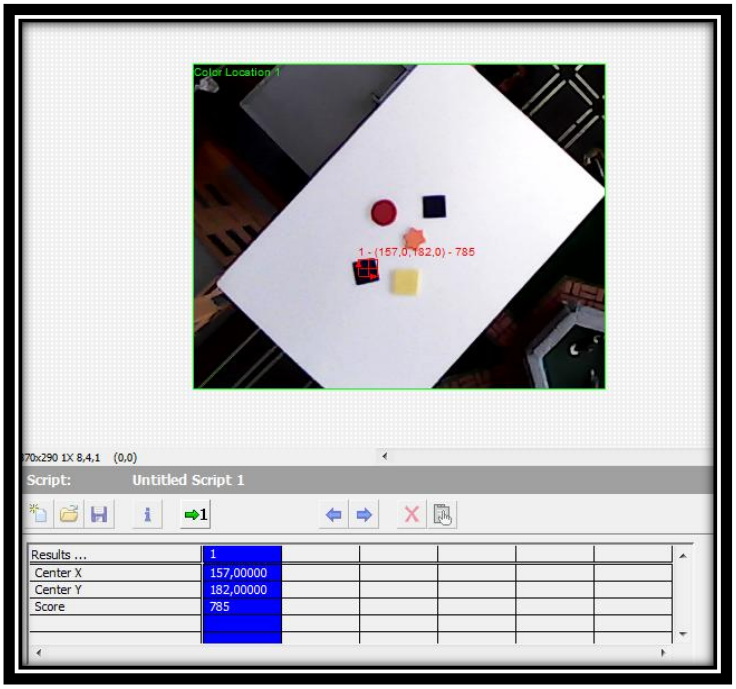
Figura III.42. Módulo Vision Assistant – Select Template

- Una vez cargado el patrón, la imagen a procesar se envía al localizador de color como se puede observar en la figura III. 43, y se realiza la diferencia absoluta de intensidad.

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.43. Color Location

- Luego de realizar la diferencia absoluta de intensidad, el programa nos entrega las coordenadas donde se encuentra ubicado el patrón y la cantidad de objetos similares detectados en el escenario de estudio, esto se puede evidenciar en la figura III.44.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.44. Coordenadas de ubicación del patrón

- Por último se realiza el cálculo de tiempo de respuesta, como se indica en la figura III.45, el cual se obtiene mediante la diferencia entre el tiempo de inicio y el tiempo final.

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.45. Cálculo tiempo de respuesta

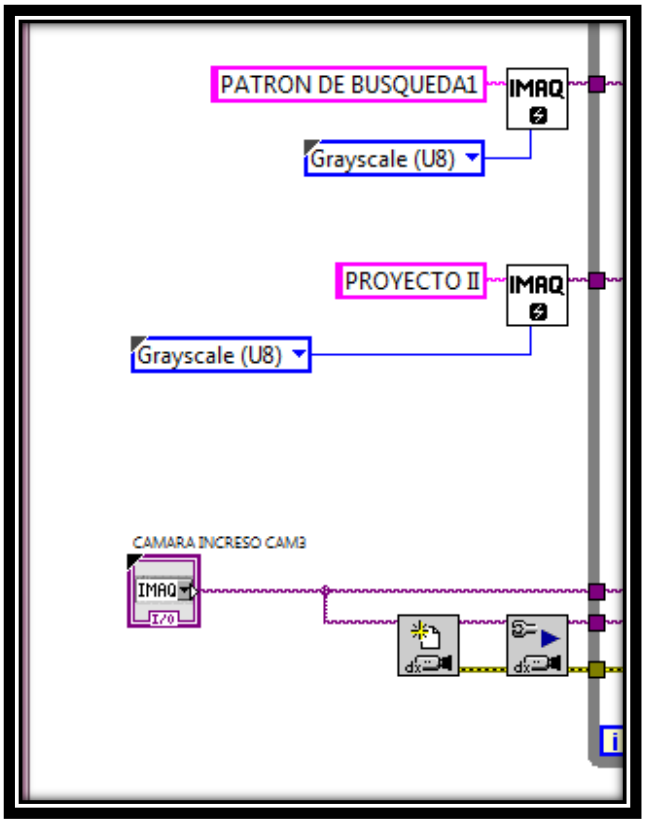
3.3.2. PRUEBA ALGORITMO DE CORRELACIÓN CRUZADA

El algoritmo de correlación cruzada tiene como objetivo detectar patrones en base a su forma.

A continuación se describe la aplicación desarrollada para dicho algoritmo.

- ♣ Como primer paso abrimos la cámara.

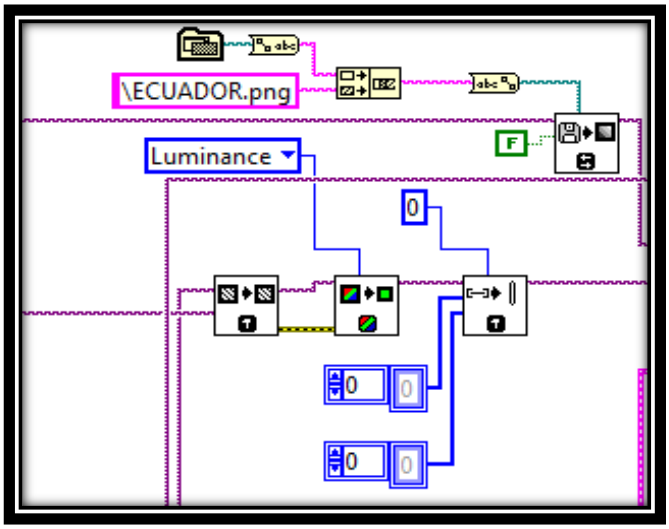
- ♣ Se asigna el espacio de memoria reservado por el IMAQ create que se puede observar en la figura III.46 y se selecciona el tipo de imagen.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.46. Asignación del espacio de memoria y selección del tipo de imagen

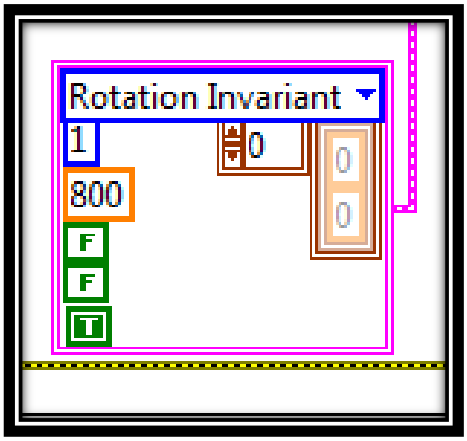
- Seleccionar el patrón de búsqueda, para lo cual se utiliza el módulo IMAQ ReadFile VI, este proceso se lo puede ver en la figura III.47.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.47. Selección del patrón

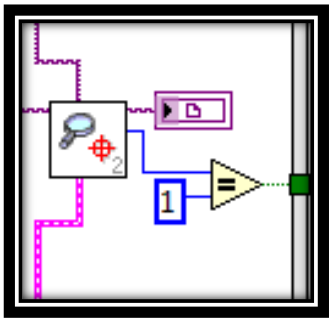
- El módulo que se puede observar en la figura III.48, se utiliza para rotar la imagen, con lo cual se facilita la detección del patrón.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.48. Módulo para la rotación de la imagen

- Mediante el módulo IMAQ Find Pattern 2 VI que se puede observar en la figura III.49, se realiza la búsqueda del patrón previamente definido.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.49. Módulo IMAQ Find Pattern 2 VI

- Por último en la figura III.50, se realiza el cálculo de tiempo de respuesta, el cual se obtiene mediante la diferencia entre el tiempo de inicio y el tiempo final.

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.50. Cálculo tiempo de respuesta

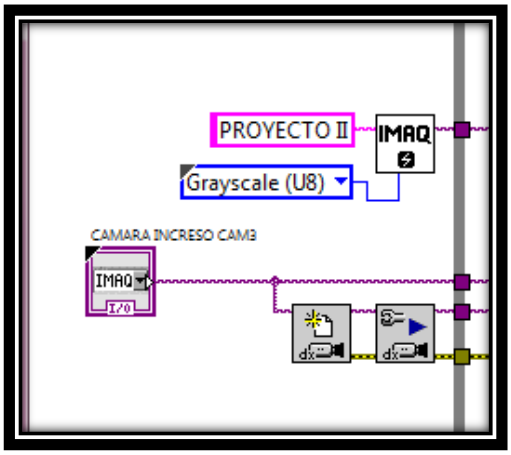
3.3.3. PRUEBA ALGORITMO DE RECONOCIMIENTO ÓPTICO DE CARACTERES (OCR)

El algoritmo de reconocimiento óptico de caracteres tiene como objetivo detectar patrones en base a caracteres, estos pueden ser letras o números.

A continuación se describe la aplicación desarrollada para dicho algoritmo.

- Como primer paso abrimos la cámara.

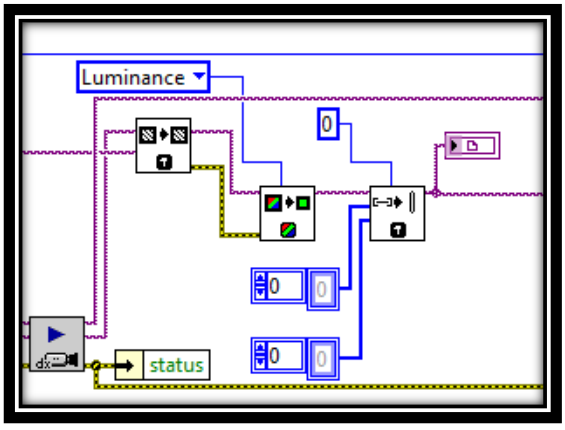
- Se asigna el espacio de memoria reservado por el IMAQ create, como se muestra en la figura III.51 y se selecciona el tipo de imagen, para esta aplicación se trabaja sólo con la imagen real.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.51. Asignación del espacio de memoria y selección del tipo de imagen

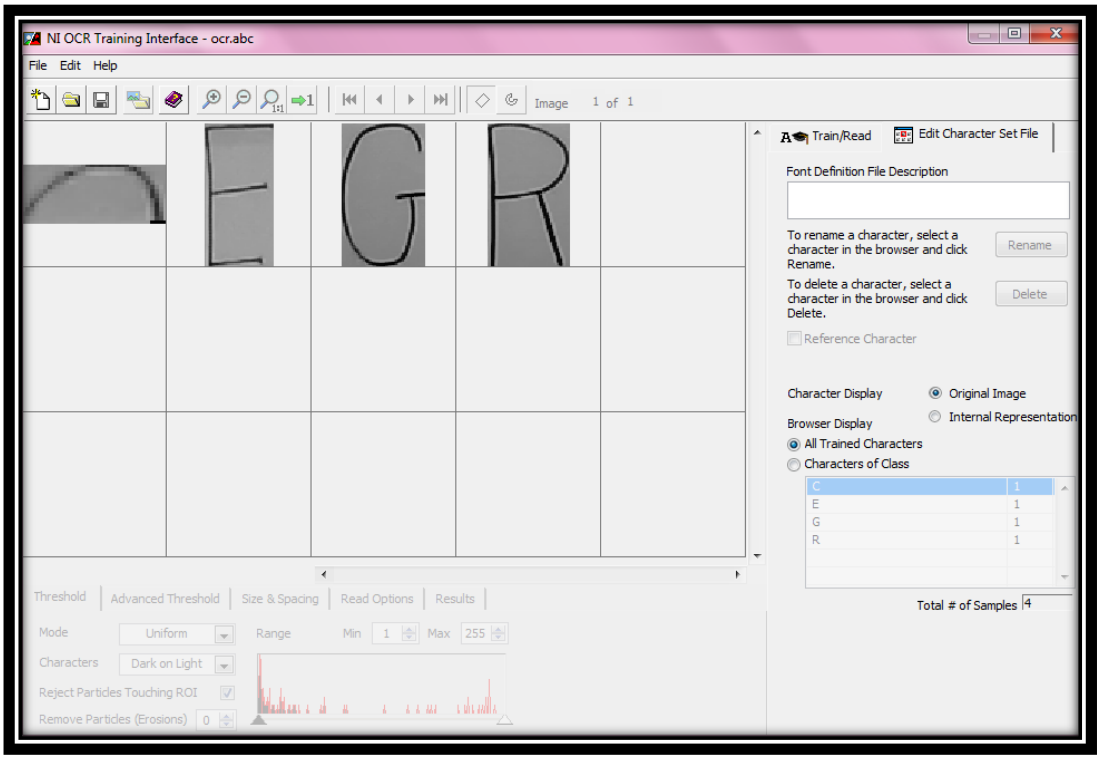
- Posteriormente se procesa la imagen para poder diferenciar sus contornos, esto se evidencia en la figura III.52.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.52. Procesamiento de la imagen

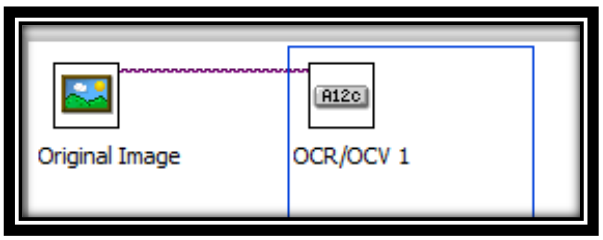
- Se crea una base de datos de los caracteres de interés con la ayuda del NI OCR Training Interface de Vision Assistant, tal como se indica en la figura III.53.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.53. NI OCR Training Interface

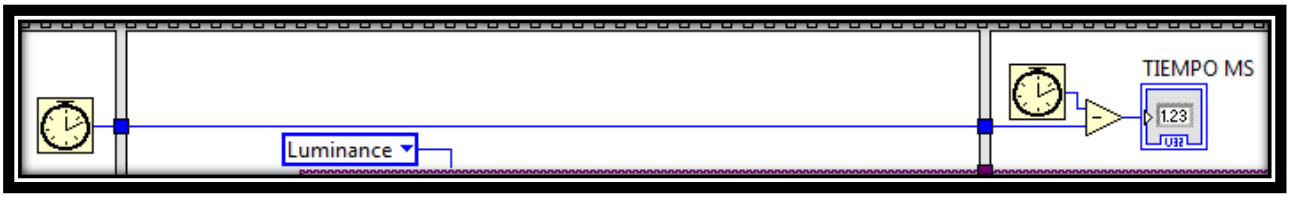
- La imagen procesada es enviada al OCR/OCV de Vision Assistant que se indica en la figura III.54, el cual busca caracteres en una región de la imagen mediante la base de datos previamente creada.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.54. OCR/OCV

- ♣ Por último se realiza el cálculo de tiempo de respuesta, el cual se obtiene mediante la diferencia entre el tiempo de inicio y el tiempo final, esto se observa en la figura III.55.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.55. Cálculo tiempo de respuesta

3.4. ESTUDIO COMPARATIVO

Para seleccionar el algoritmo más adecuado para el reconocimiento de patrones se utilizó el método de diseño completamente al azar (DCA), con un nivel de confianza del 99%, mientras que para los cálculos estadísticos se utilizó el software Minitab.

Se realizó 10 tomas de muestras de cada parámetro con los tres algoritmos propuestos para el estudio.

Para calcular n cada algoritmo propuesto representa un tratamiento y cada toma de muestra equivale a una repetición.

$$n = (t * r)$$
$$n = (3 * 10)$$
$$n = 30$$

Para tomar las muestras respectivas con cada uno de los algoritmos propuestos se debe tener en cuenta las siguientes consideraciones:

- ***Algoritmo de coincidencia de colores***

Las pruebas para el algoritmo de coincidencia de colores se realizaron en un rango de uno a cinco patrones, los mismos fueron aumentando o disminuyendo de manera aleatoria para cada caso (1, 4, 2, 5, 3).

El color elegido para el patrón de búsqueda fue el azul.

Los colores utilizados para las pruebas se pueden observar en la tabla III.6.

Tabla III.6. Combinaciones de colores

No. Prueba	No. Patrones	Colores
1	1	Amarillo, Rojo, **Azul**, Tomate, Celeste
2	3	Tomate, **Azul,** Amarillo, **Azul, Azul**
3	5	**Azul, Azul, Azul, Azul, Azul**
4	4	Celeste, **Azul, Azul**, **Azul**, **Azul**
5	2	**Azul**, Tomate, Rojo, Tomate, **Azul**
6	5	**Azul, Azul, Azul, Azul, Azul**
7	3	**Azul**, Amarillo, **Azul**, Celeste, **Azul**
8	1	Tomate, **Azul**, Rojo, Celeste, Amarillo
9	4	**Azul**, **Azul**, Celeste, **Azul**, **Azul**
10	2	Amarillo, **Azul**, Rojo, **Azul,** Tomate

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

La cámara se fijó a una distancia de 90 cm.

En la tabla III.7 se indica la información recabada por este algoritmo.

Tabla III.7. Muestreo de datos

No. PRUEBA	No. PATRONES DE BÚSQUEDA	No. ACIERTOS	PRECISIÓN (%)	TIEMPO RESPUESTA (ms)
1	1	1	100	2
2	3	3	100	3
3	5	5	100	3
4	4	4	100	4
5	2	2	100	3
6	5	5	100	4
7	3	3	100	3
8	1	1	100	3
9	4	4	100	2
10	2	2	100	3

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

- ***Algoritmo de correlación cruzada***

Igual que en el caso anterior las pruebas se realizaron con un mínimo de un patrón y un máximo de cinco patrones, los mismos fueron aumentando o disminuyendo de manera aleatoria para cada caso (1, 4, 2, 5, 3).

La forma elegida para el patrón de búsqueda fue el cuadrado.

En la tabla III.8 se puede observar las figuras utilizadas para realizar las respectivas pruebas.

Tabla III.8. Combinaciones de figuras

No. Prueba	No. Patrones	Figuras
1	1	
2	3	
3	5	
4	4	

5	2	
6	5	
7	3	
8	1	
9	4	
10	2	

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Al igual que en el algoritmo de coincidencia de colores la cámara se fijó a una distancia de 90 cm.

En la tabla III.9 se indica la información recabada por este algoritmo.

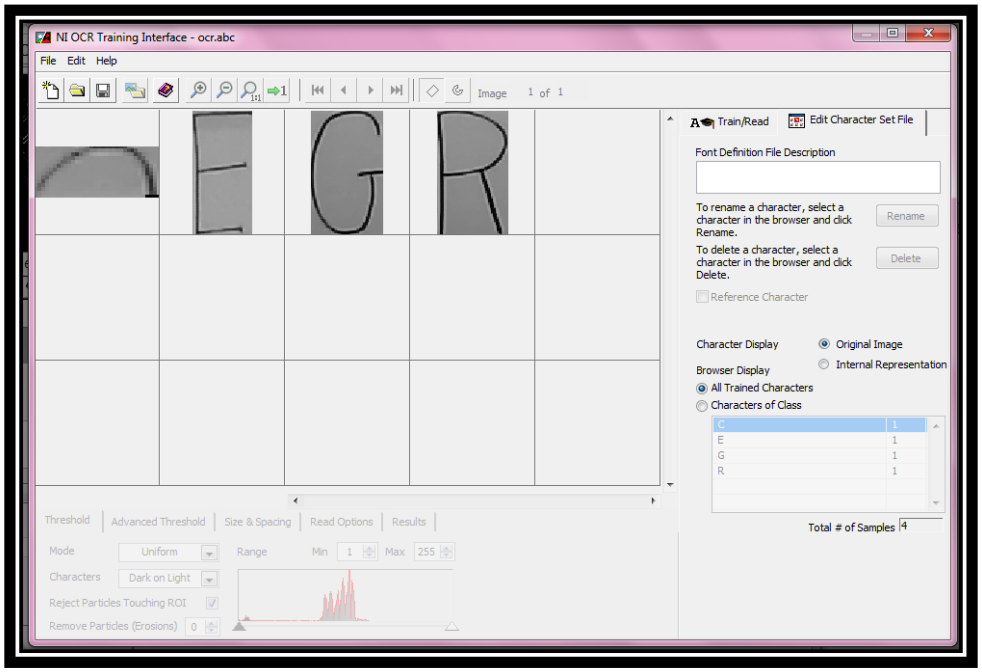
Tabla III.9. Muestreo de datos

No. PRUEBA	No. FIGURAS DE BÚSQUEDA	No. ACIERTOS	PRECISIÓN (%)	TIEMPO RESPUESTA (ms)
1	1	1	100	367
2	3	1	33,33	382
3	5	1	20	395
4	4	1	25	385
5	2	1	50	374
6	5	1	20	394
7	3	1	33,33	369
8	1	1	100	365
9	4	1	25	374
10	2	1	50	368

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

- ***Algoritmo óptico de caracteres (OCR)***

Para las pruebas realizadas con este algoritmo se creó previamente una base de datos en la cual se almacenaron los caracteres de búsqueda, esto se puede evidenciar en la figura III.56.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura III.56. Base de datos

Para esta aplicación se almacenaron en la base de datos los siguientes caracteres: C, E, G y R.

Para este algoritmo la cámara se fijó a una distancia de 90 cm.

En la tabla III.10 se indica la información recabada por este algoritmo.

Tabla III.10. Muestreo de datos

No. PRUEBA	No. CARACTERES	INFORMACIÓN RECABADA	No. ACIERTOS	PRECISIÓN (%)	TIEMPO RESPUESTA (ms)
1	CEGR	CERE	3	75	301
2	CEAB	C???	1	25	312
3	1CEC	?C?C	2	50	315
4	REGR	CE	1	25	324
5	CECR	CC	2	50	297
6	F2E4	?E?	1	25	300
7	ZRGE	??CE	1	25	320
8	123R	???R	1	25	310
9	DGAJ	G?	1	25	313
10	QNJG	??C	0	0	322

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

En la tabla III.11 se detalla las medias generales de las muestras tomadas.

Tabla III.11. Medias de los datos

	PRECISIÓN (%)	TIEMPO RESPUESTA (ms)
PROMEDIO ALGORITMO COINCIDENCIA COLORES	100	3
PROMEDIO ALGORITMO CORRELACIÓN CRUZADA	45,666	377,3
PROMEDIO ALGORITMO OCR	32,5	311,4

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

- **ANÁLISIS VARIANZA**

Luego de calcular las medias respectivas para cada parámetro de cada uno de los algoritmos, se realiza el análisis de la varianza.

- **PRECISIÓN (%)**

Planteamiento Hipótesis

Ho: La media de precisión para los tres algoritmos es igual.

$$H_o : t_i = 0$$

Ha: La media de precisión para los tres algoritmos no es igual.

$$H_o : t_i \neq 0$$

Cálculo Varianza

Tabla III.12. Varianza para precisión

ALGORITMOS	MUESTRAS	SUMA	PROMEDIO	VARIANZA
COINCIDENCIA COLORES	10	1000	100	0
CORRELACIÓN CRUZADA	10	456,66	45,666	935,3269156
OCR	10	325	32,5	423,6111111

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Análisis ANOVA

Tabla III.13. ANOVA de precisión

FUENTES DE VARIACIÓN	GRADOS DE LIBERTAD	SUMA DE CUADRADOS	PROMEDIO DE LOS CUADRADOS	F CALCULADA	F TABULADA
Entre tratamientos	2	25605,924	12802,962	28,263	5,49
Dentro de tratamientos	27	12230,664	452,988		
Total	29	37836,588			

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

INTERPRETACIÓN

Al terminar el análisis de varianza, el valor de F Calculada es mayor que el de F Tabulada, por lo tanto se rechaza la hipótesis nula y se decidió que el algoritmo de coincidencia de colores es más preciso.

♣ TIEMPO DE RESPUESTA (ms)

Planteamiento Hipótesis

Ho: La media de tiempo de respuesta para los tres algoritmos es igual.

$$H_o : t_i = 0$$

Ha: La media de tiempo de respuesta para los tres algoritmos no es igual.

$$H_a : t_i \neq 0$$

Cálculo Varianza

Tabla III.14. Varianza para tiempo de respuesta

ALGORITMOS	MUESTRAS	SUMA	PROMEDIO	VARIANZA
COINCIDENCIA COLORES	10	30	3	0,444444444
CORRELACIÓN CRUZADA	10	3773	377,3	123,1222222
OCR	10	3114	311,4	89,82222222

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Análisis ANOVA

Tabla III.15. ANOVA de tiempo de respuesta

FUENTES DE VARIACIÓN	GRADOS DE LIBERTAD	SUMA DE CUADRADOS	PROMEDIO DE LOS CUADRADOS	F CALCULADA	F TABULADA
Entre tratamientos	2	798512,867	399256,434	15,046	5,49
Dentro de tratamientos	27	716474,663	26536,099		
Total	29	1514987,53			

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

INTERPRETACIÓN

Al terminar el análisis de varianza, el valor de F Calculada es mayor que el de F Tabulada, por lo tanto se rechaza la hipótesis nula y se decidió que el algoritmo de coincidencia de colores tiene un menor tiempo de respuesta.

Mediante los resultados obtenidos en las tablas III.11 y III.13, en base a la varianza, se pudo comprobar que el estudio de algoritmos de reconocimiento de patrones permitió seleccionar el algoritmo de coincidencia de colores como el más adecuado para la implementación de un semáforo inteligente mediante FPGAs, para minimizar la congestión vehicular.

CAPÍTULO IV

DISEÑO E IMPLEMENTACIÓN

En este capítulo se detalla el diseño e implementación del prototipo de semáforo inteligente mediante FPGA, utilizando el algoritmo más adecuado para el reconocimiento de patrones, para lo cual se partió de los resultados obtenidos en el capítulo anterior.

Para el diseño del prototipo se ha divido el mismo en dos etapas, la primera etapa es el diseño del hardware en el cual vamos a determinar y seleccionar los materiales a utilizarse, la segunda etapa es el diseño del software en el cual se desarrolla la interfaz gráfica y electrónica del sistema.

4.1. DISEÑO DEL HARDWARE

El diseño del hardware tiene como objetivo controlar el prototipo implementado.

4.1.1. SELECCIÓN DE LA INFRAESTRUCTURA DEL PROTOTIPO

En esta sección se detalla los materiales utilizados que conforman el prototipo para la automatización de un semáforo inteligente mediante FPGAs.

4.1.1.1. SELECCIÓN DE PISTAS

Para la construcción de la pista se utilizó pistas eléctricas, ya que éstas al funcionar de manera automática, facilitan el control de los vehículos para el desarrollo del prototipo.

Dichas pistas requieren para su funcionamiento de una fuente de alimentación, la misma permite variar el voltaje, lo cual sirve para controlar la velocidad de los autos.

La electricidad llega al carro a través de los rieles de metal insertados en la pista y es captada por dos escobillas metálicas, las cuales se encuentran montadas en la parte delantera del carro.

Para mantener los autos sujetos a la pista los mismos se encuentran equipados con imanes.

4.1.1.2. SELECCIÓN DE LA CÁMARA

Debido a su fácil instalación, manejo y compatibilidad de drivers con LabView se ha utilizado la Genius FaceCam 320X, la misma que recepta

imágenes claras facilitando la manipulación de los datos adquiridos para el control de nuestro sistema.

Este tipo de cámaras trabajan con sensores CMOS, los cuales permiten adquirir la imagen en menor tiempo y con una buena calidad.

Fuente: http://www.geniusnet.com/Genius/wSite/ct?xItem=53585&ctNode=1304

Figura IV.57. Genius FaceCam 320X

Características de la cámara:

Tabla IV.16. Características de la Genius FaceCam 320X

Interface	USB 2.0
File format	MJPEG/WMV
Max. Still Image Resolution	640 x 480 pixels
Lens Type	Manual focus
Video resolution	VGA: Up to 30fps
Image Sensor	VGA pixel CMOS
IPM(Image Protection Mechanism)	YES
UVC (Plug & Play)	YES

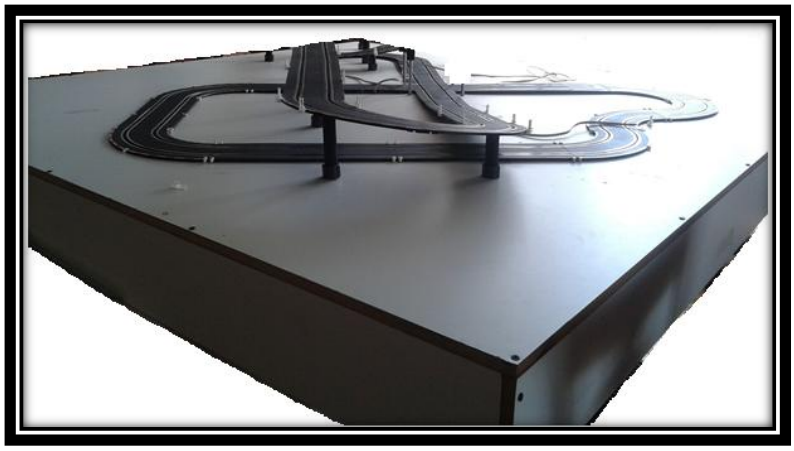
Fuente: http://www.geniusnet.com/Genius/wSite/ct?xItem=53585&ctNode=1304

4.1.1.3. SELECCIÓN DEL MATERIAL DE CONSTRUCCIÓN

Para el diseño e implementación de la maqueta se ha utilizado madera MDF laminada, ya que este es un material fácil de cortar, moldear y taladrar.

Tomando en cuenta los elementos a ser montados como la pista, placas, fuente de alimentación, entre otros, el diseño cuenta con un área de trabajo de 200 x 180 cm (largo x ancho).

A continuación se muestra el diseño de la base.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.58. Diseño de la base

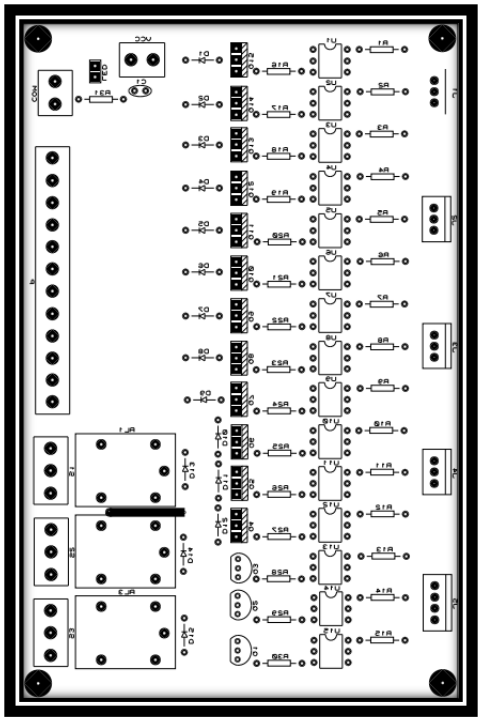
4.1.2. DISEÑO DE LA INFRAESTRUCTURA ELÉCTRICA

En esta sección se detalla la elaboración de la parte electrónica para la automatización de nuestro prototipo.

4.1.2.1. INTERFAZ DE POTENCIA

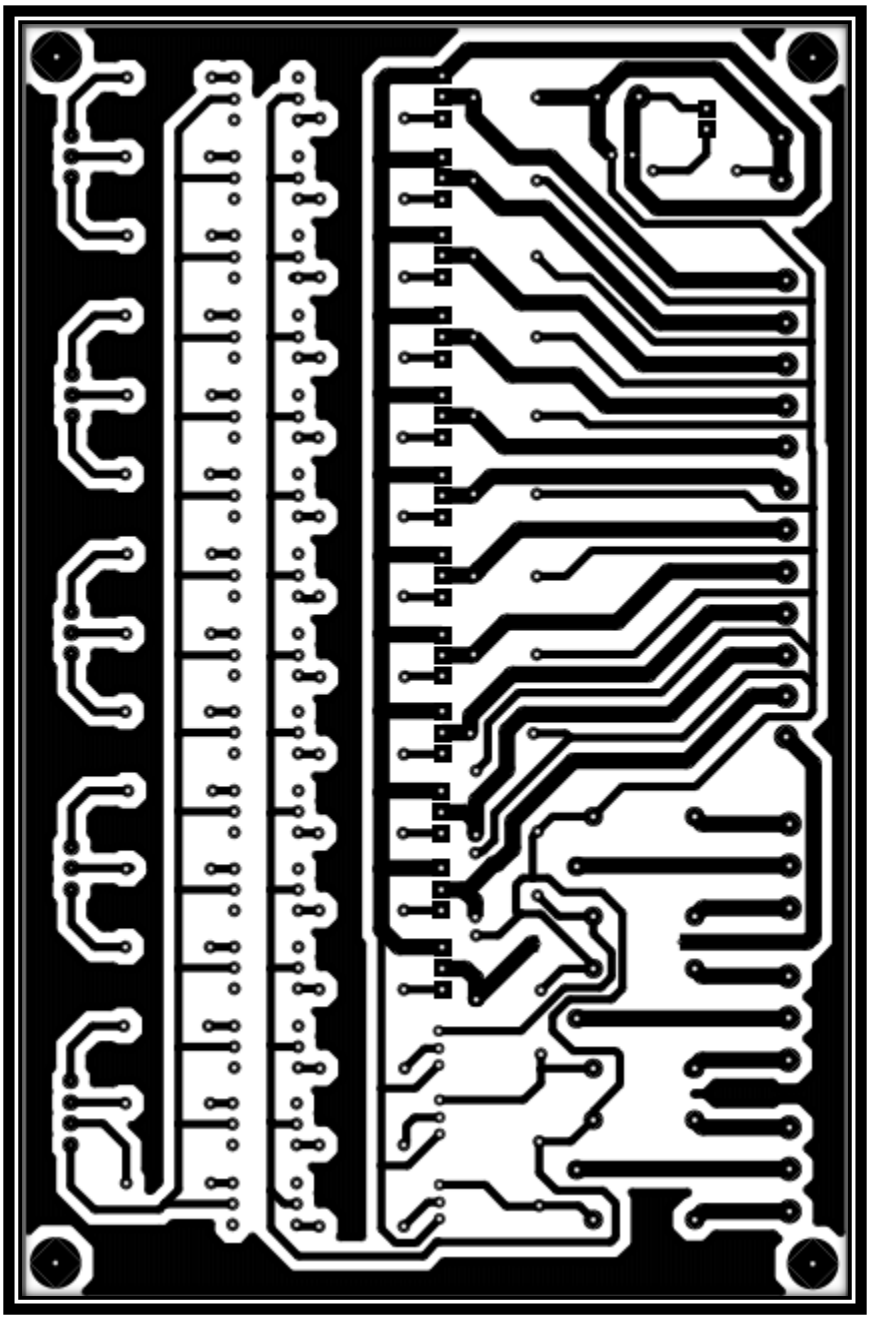
Para el diseño de la interfaz de potencia se elaboró cuatro placas, la primera controla el voltaje de 3.3 voltios que sirve para alimentar la FPGA; la segunda tiene como función controlar los cuatro semáforos que conforman la pista; la tercera es una interfaz para ordenar las líneas de los sensores de contacto (microswitch); por último la cuarta placa es un zócalo conector macho para el socket de la tarjeta SPARTAN 3E.

Para la primera placa se ha utilizado opto transistores 4N25, cuya función es aislar la FPGA y evitar daños en la misma, dichos menoscabos pueden darse por transientes producidos por los motores de los carros y corto circuitos en las pista, así como en los motores. Para controlar las pistas se ha utilizado el TIP 142, el cual nos permite controlar 10 A y 100 VDC. El diseño de esta placa se puede observar en la figura IV.59 y IV.60.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

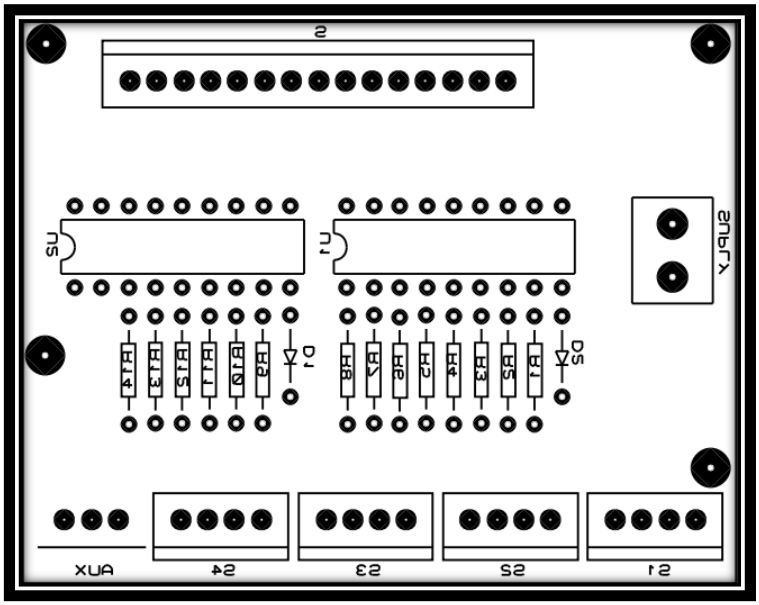
Figura IV.59. Diseño de máscara de la placa de potencia



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

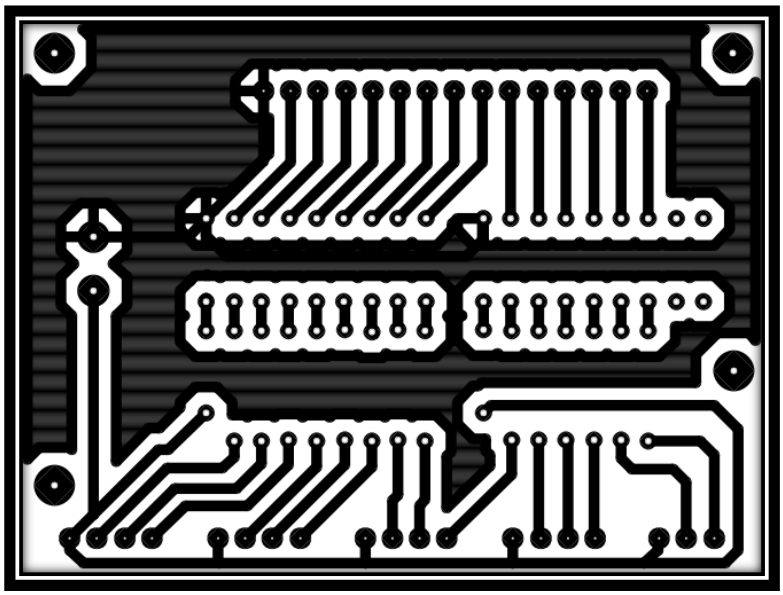
Figura IV.60. Diseño de las pistas de la placa de potencia

La segunda placa se implementó con integrados ULN 2803, este es un driver que empaqueta 8 transistores de arreglo Darlington, para controlar los semáforos. En las figuras IV.61 y IV.62 se puede observar el diseño de la placa.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

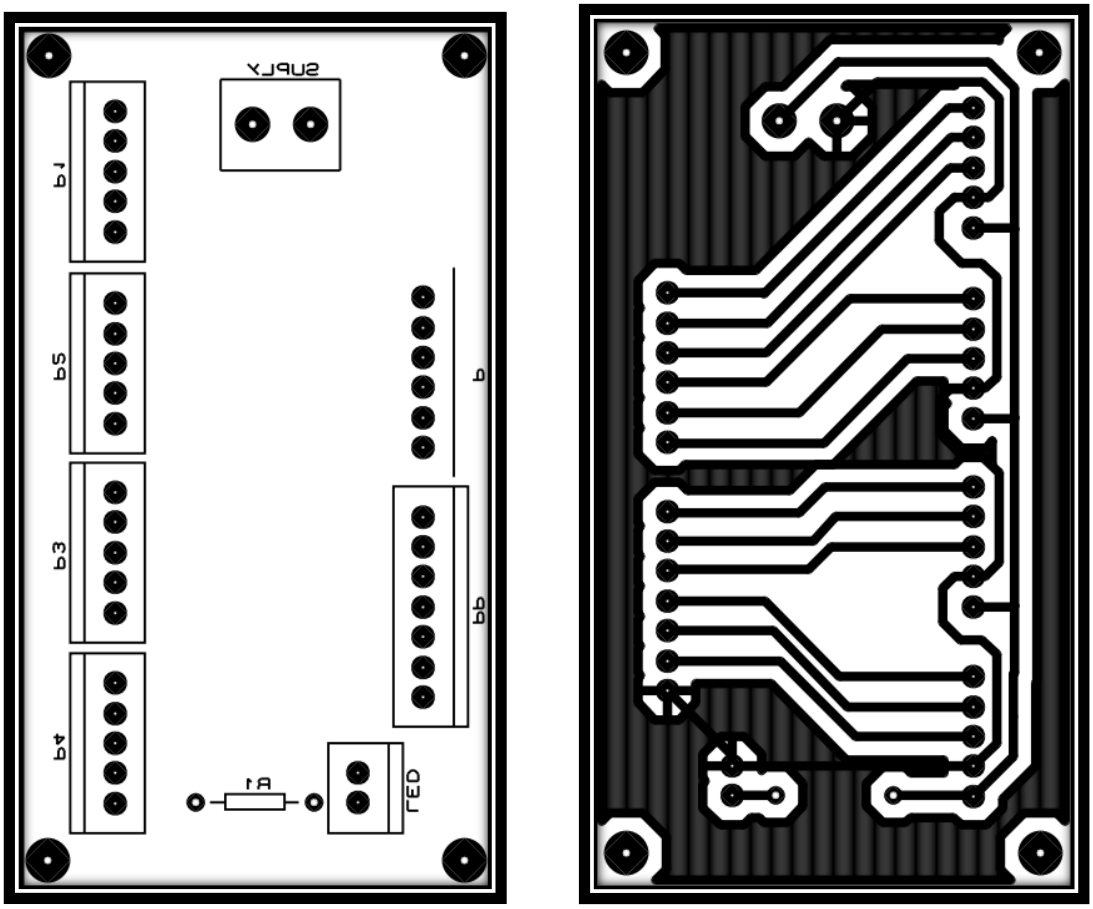
Figura IV.61. Diseño de la máscara de las placas de los semáforos



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.62. Diseño de las pistas de la placa de los semáforos

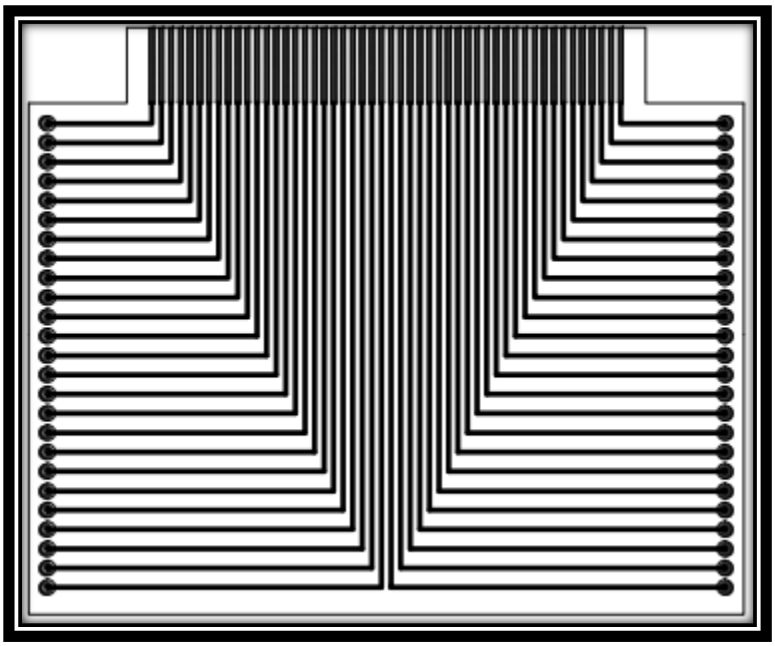
La tercera placa es un paso para realizar el control de los sensores (micro switch). La misma se puede observar el la figura IV.63.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.63. Diseño de la máscara y las pistas de las placas para los sensores

El diseño del zócalo se lo realizó en AUTOCAD, debido a la precisión en micrómetros que nos brinda este programa. El diseño del mismo se lo puede observar en la figura IV.64.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.64. Zócalo conector macho

4.1.2.2. FUENTE DE ALIMENTACIÓN

Para energizar a todo nuestro sistema es necesario una tensión mínima de 5 V, por lo cual se ha utilizado una fuente de computadora.

Fuente: http://www.monografias.com/trabajos62/arquitectura-computadores-guia-cuatro/arquitectura-computadores-guia-cuatro2.shtml

Figura IV.65. Fuente de computadora

Características principales de la fuente:

- Salida de voltaje de 5V a 12V.
- Voltaje de ingreso 127V o 240V.

4.2. DISEÑO DEL SOFTWARE

La parte del diseño del software tiene la misma importancia que el hardware, pues los dos en conjunto harán que el sistema funcione correctamente. La construcción del software para el control se lo dividirá en tres partes que son: el software de control diseñado en LabView, el software para la FPGA y el tercero realizaremos la unión de la visión artificial con la FPGA.

Se eligió la plataforma de programación LabView por las funciones con las que cuenta para procesar imágenes y sus herramientas de visualización para trabajar con visión artificial, además dicha plataforma cuenta con procesamiento de imágenes en escala de grises, a color y binarias.

Labview también cuenta con el módulo NI LabView FPGA, el cual nos permite desarrollar programas complejos de una manera más eficiente y efectiva.

La plataforma de programación Labview nos brinda la facilidad de unir visión artificial con la FPGA, lo cual simplifica la implementación del prototipo propuesto.

4.2.1. ELABORACIÓN DEL SOFTWARE EN LABVIEW

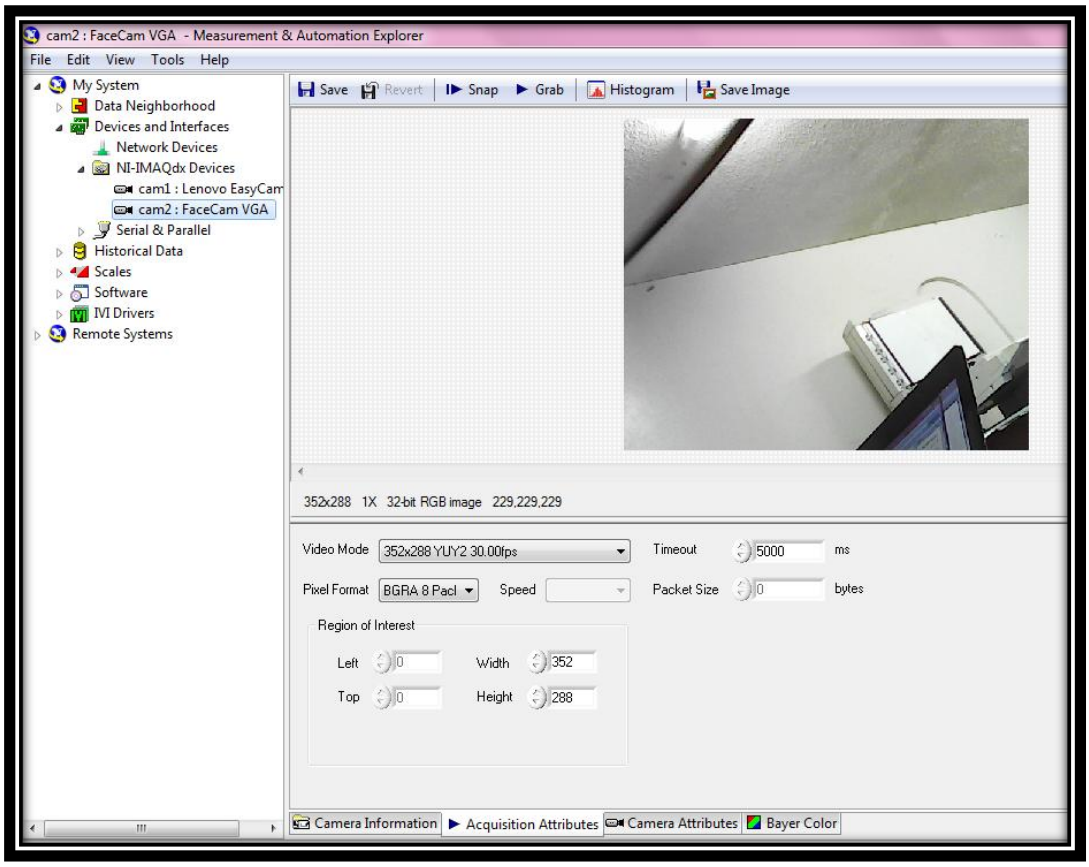
El software desarrollado en LabView controlará en su totalidad al sistema implementado, a continuación se indican las cosas que haremos con LabView.

- ♣ Comprobar la compatibilidad de los drivers de la cámara.

- ♣ Centrar la imagen, esto nos permitirá tener una buena captación de la imagen para los planos que se van a visualizar.
- ♣ Segmentar la imagen, esto nos ayudará a segmentar las áreas de las pistas asignadas dentro de la imagen general.
- ♣ Aplicar el algoritmo.
- ♣ Enviar datos a la FPGA.

A continuación se detallaran cada una de las partes mencionadas anteriormente.

- ***COMPROBAR LA COMPATIBILIDAD DE LOS DRIVERS DE LA CÁMARA***
  - Conectamos la cámara.
  - Abrimos el NI MAX.
  - Vamos a dispositivos e interfaces.
  - Esperamos que aparezca el módulo NI-IMAQdx.
  - Elegimos la cámara que se va abrir de las que se encuentran conectadas y son compatibles con LabView.
  - Luego se nos abre la siguiente ventana, la cual se puede observar en la figura IV.66.

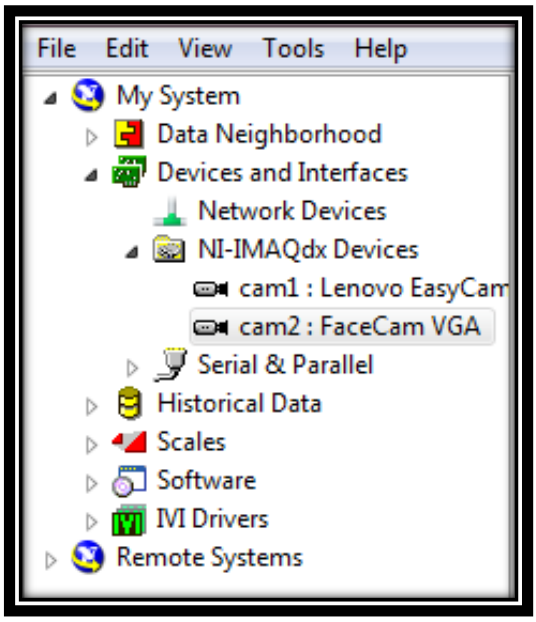


Fuente: NI MAX

Figura IV.66. Módulo NI- IMAQdx Devices

En la pantalla mostrada en la figura IV.66 podemos modificar diferentes parámetros como son modo de video, formato de la imagen, tamaño entre otras, de acuerdo a los requerimientos del usuario.

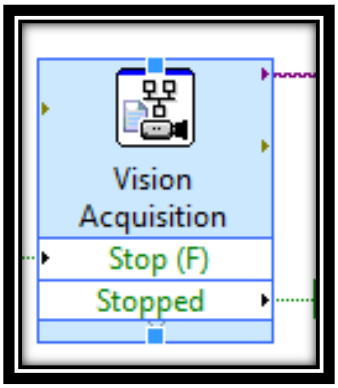
Si la cámara no es compatible con Labview no se puede visualizar en el módulo NI- IMAQdx Devices, que se puede observar en la figura IV.67.



Fuente: NI MAX

Figura IV.67. Módulo NI- IMAQdx Devices

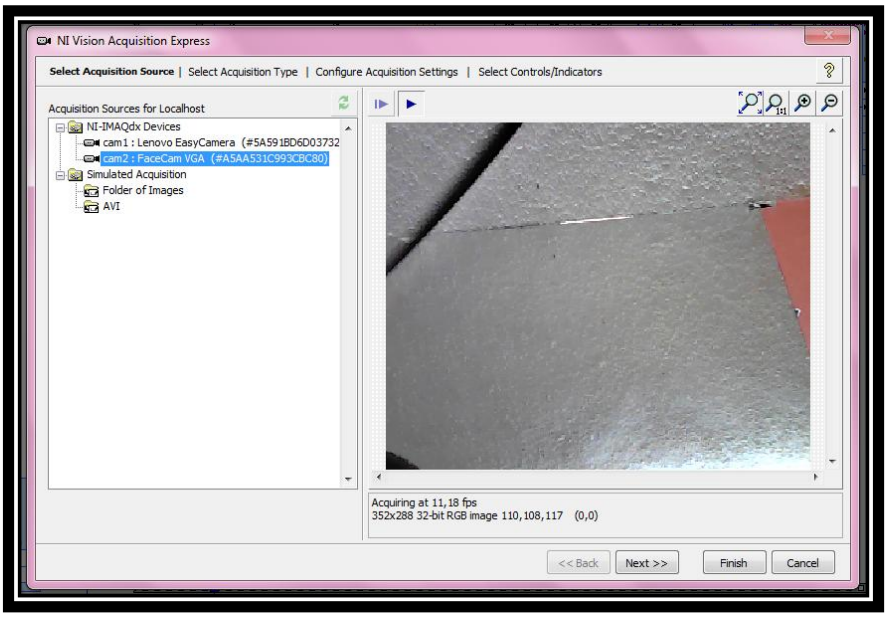
- Abrimos la cámara en el VI de Labview, para lo cual utilizamos el módulo Vision Acquisition que se observa en la figura IV.68.



Fuente: LabView 2012

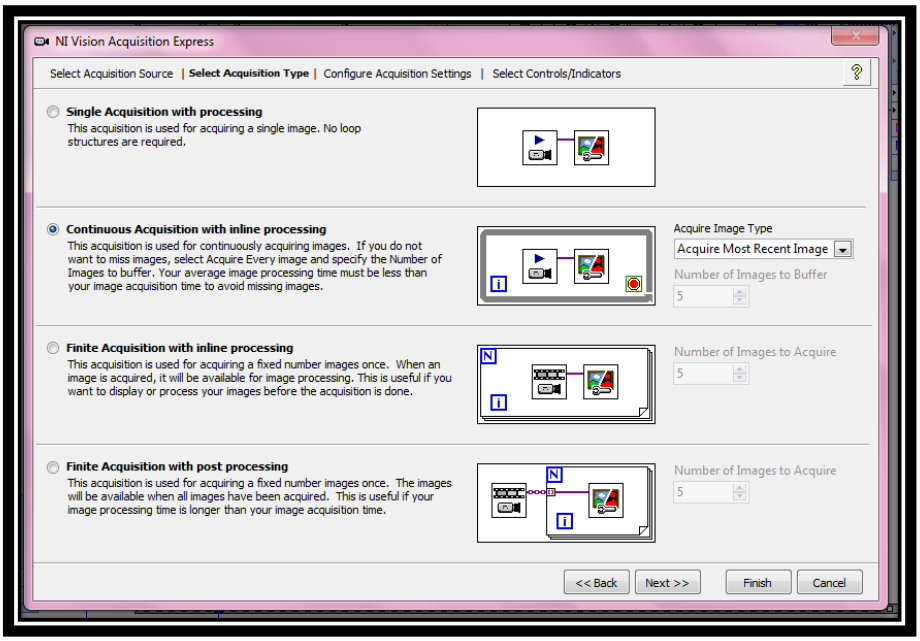
Figura IV.68. Módulo Vision Acquisition

Este módulo nos permite adquirir la imagen de una manera más fácil y rápida mediante configuraciones del mismo, dichas configuraciones se muestran en las figuras IV.69 y IV.70.



Fuente: NI Vision Acquisition Express- LabView 2012

Figura IV.69. Adquisición de la imagen



Fuente: NI Vision Acquisition Express- LabView 2012

Figura IV.70. Adquisición continua con procesamiento en línea de la imagen

Al finalizar esto ya podemos observar la imagen adquirida en el VI.

- **CENTRAR LA IMAGEN**

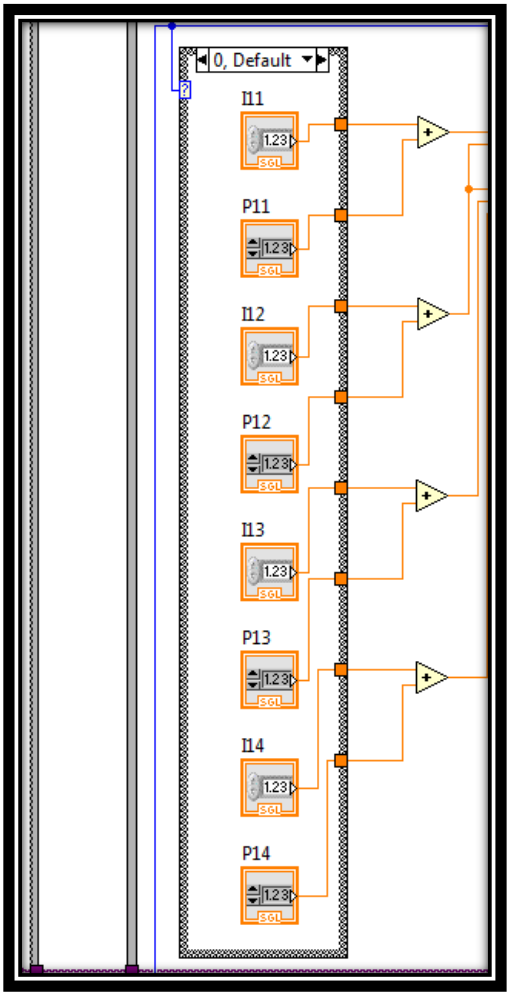
Para centrar la imagen la cámara debe enfocar todas las áreas que se va a segmentar, ya que las imágenes adquiridas son dinámicas el tiempo de lectura para este caso es de 200 ms por área segmentada, para esta aplicación son cuatro pistas las que se deben enfocar con claridad, estas se pueden observar en la figura IV.71.

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.71. Área a segmentar

- **SEGMENTAR LA IMAGEN**

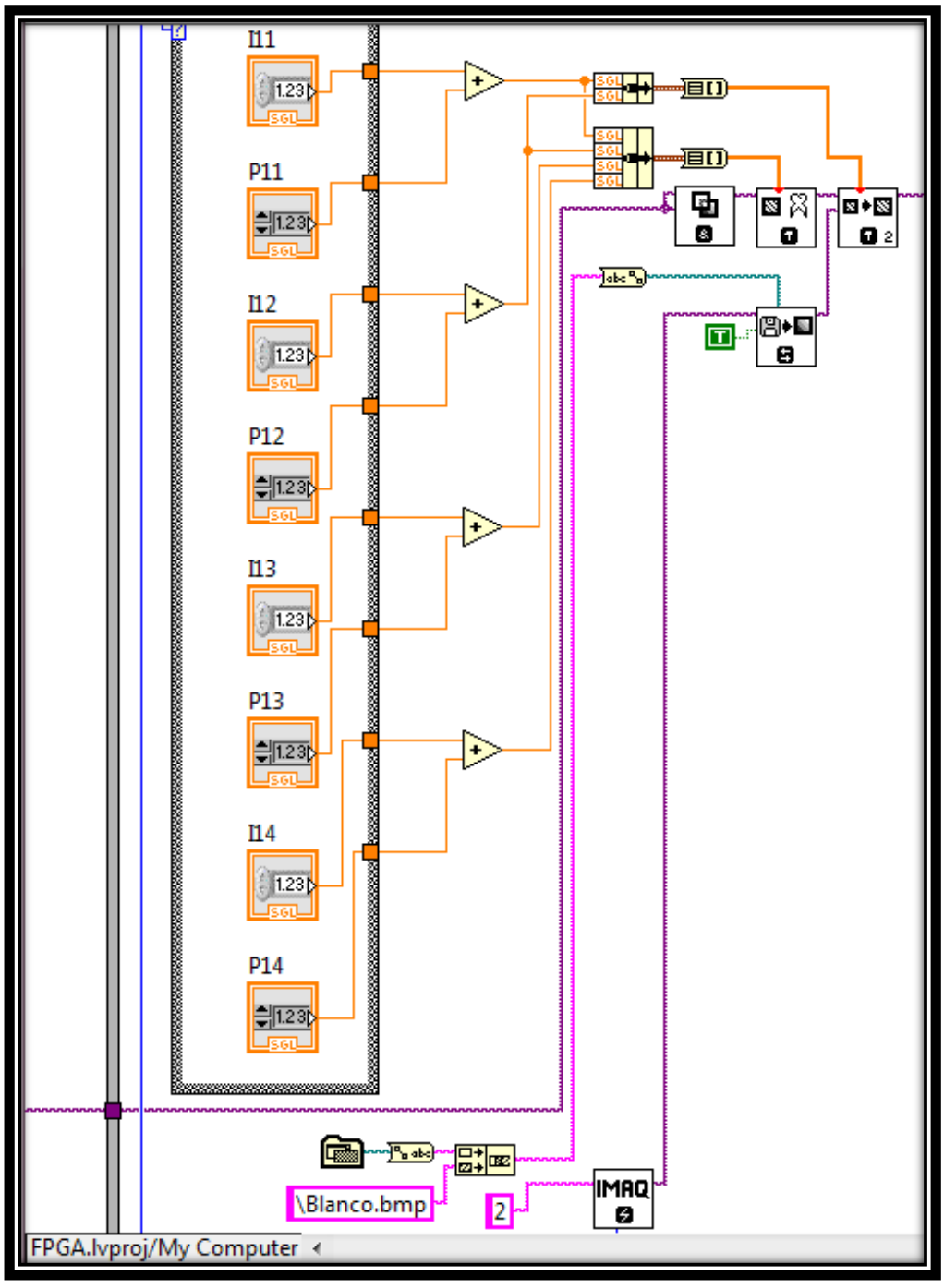
Una vez centrada la imagen segmentamos por tiempos nuestras regiones de interés, cada pista se va a segmentar cada 200 ms, para lo cual primero asignamos las coordenadas de corte en la imagen, este proceso se puede observar en la figura IV.72.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.72. Valores de las coordenadas para segmentar la imagen

Una vez que se designa las coordenadas, se procede a establecer un espacio de memoria y posteriormente a segmentar la imagen de interés, este proceso se puede observar en la figura IV.73.

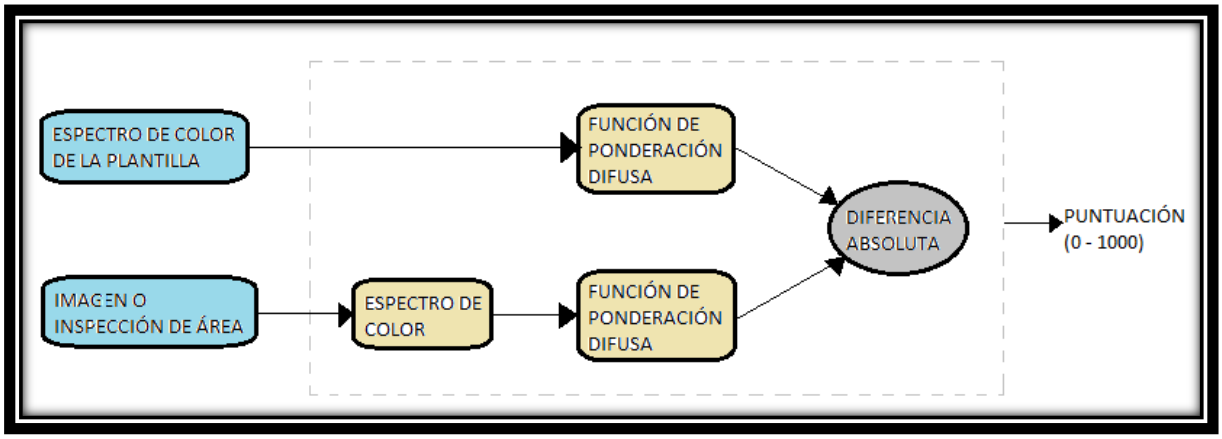


Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.73. Segmentación de la imagen con las coordenadas establecidas

Estos pasos se deben repetir cuatro veces cada 200 ms para esta aplicación.

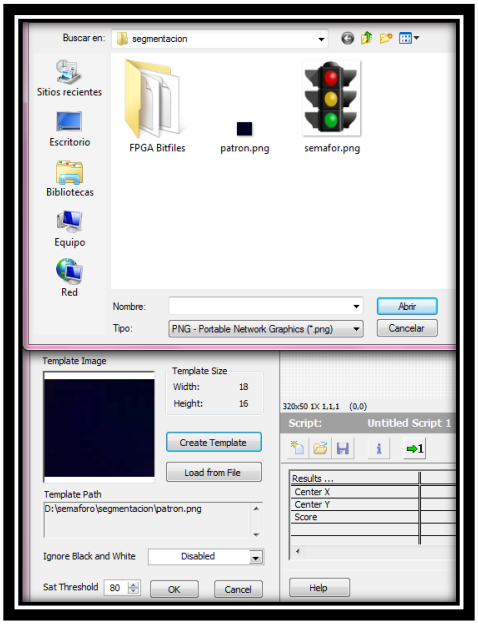
- **APLICACIÓN DEL ALGORITMO**



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.74. Esquema algoritmo de Coincidencia de Colores

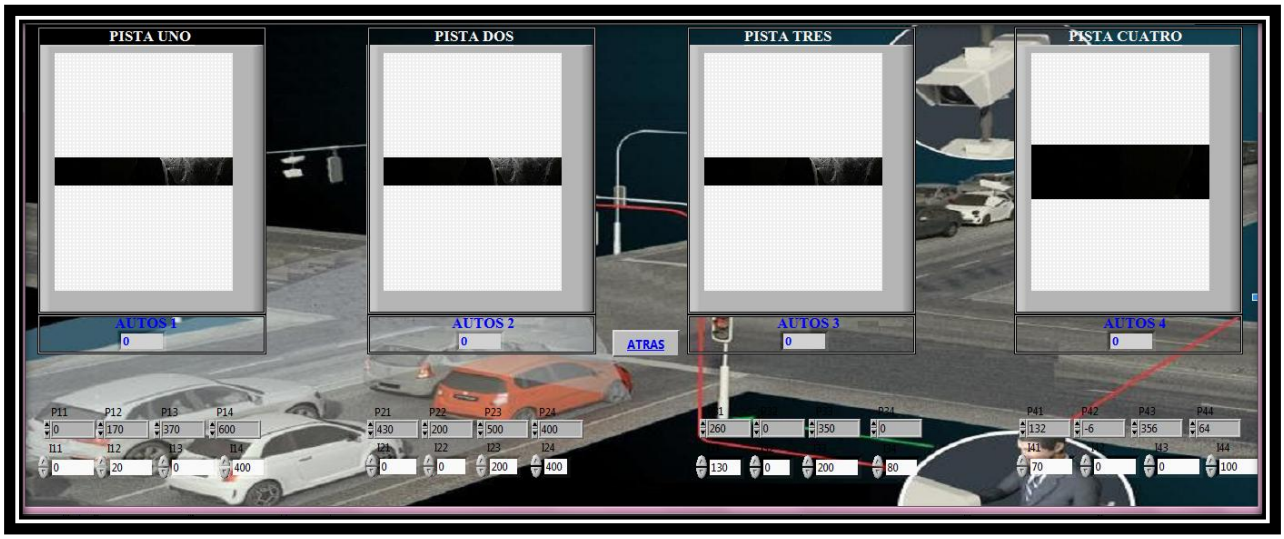
El algoritmo de la figura IV.74 funciona con lógica difusa para el reconocimiento de los patrones, el mismo lo primero que realiza es cargar el patrón de búsqueda, esto se puede observar en la figura IV.75, seguidamente busca el número de patrones existentes en el área segmentada.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.75. Asignación del patrón

Una vez asignado el patrón se procede a ejecutar el algoritmo cada 200 ms, esto se lo realiza con cada una de las imágenes segmentadas, para esta aplicación son cuatro imágenes segmentadas como se puede observar en la figura IV.76.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.76. Visualización de las pistas para la búsqueda del patrón

Como podemos observar en la figura IV.76 se aprecia las cuatro pistas segmentadas, el número de patrones encontrados en cada pista y sus coordenadas de segmentación.

- **ENVIO DE DATOS A LA FPGA**

Para determinar la máxima cantidad de autos en una determinada pista se toma como referencia el uno por características de diseño de la pista.

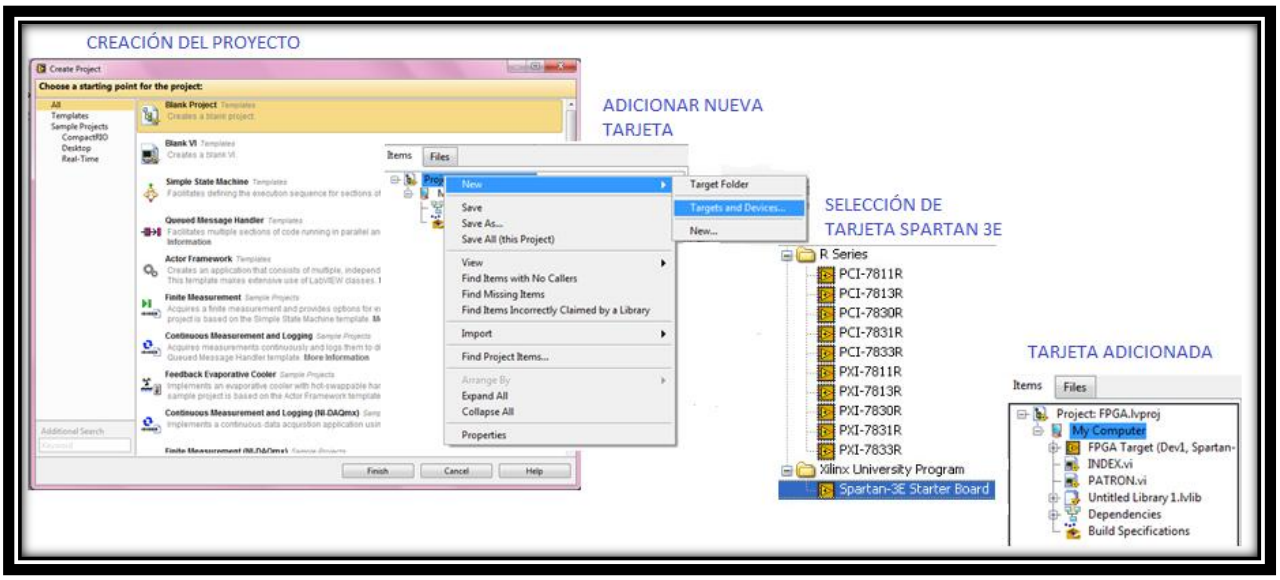
Para esta aplicación cuando la lectura de los patrones es mayor a uno seria el máximo de carros encontrados.

4.2.2. ELABORACIÓN DEL SOFTWARE PARA LA FPGA

Para elaborar el software de la FPGA primero se debe cargar los controladores de la tarjeta Spartan 3E, los mismos se pueden descargar de forma gratuita de la National Instruments ingresando al siguiente enlace:

http://digital.ni.com/public.nsf/allkb/949B93D25041EB2F86257626004293FD

Una vez cargado los drivers procedemos a crear un proyecto para FPGA de Xilinx, como se observa en la figura IV.77.

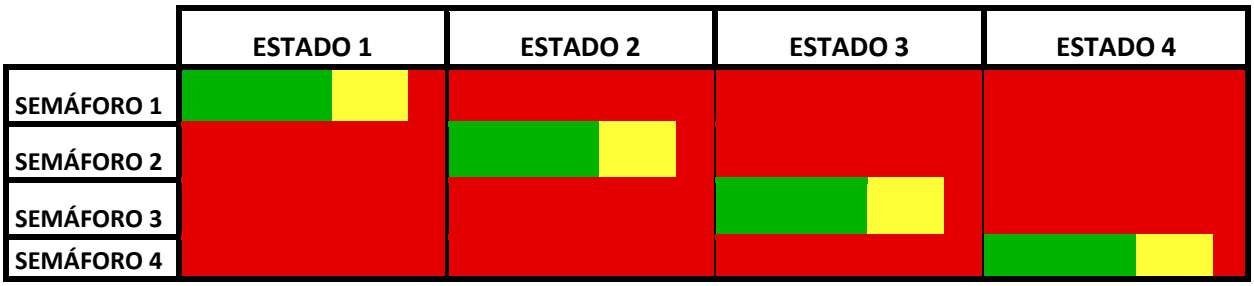


Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.77. Creación del proyecto para la FPGA

- **ANÁLISIS Y FUNCIONAMIENTO DEL SEMÁFORO**

  Antes de realizar el software debemos entender como funciona un semáforo y la forma como se va automatizar el mismo, para lo cual nos ayudaremos de una máquina de estados.

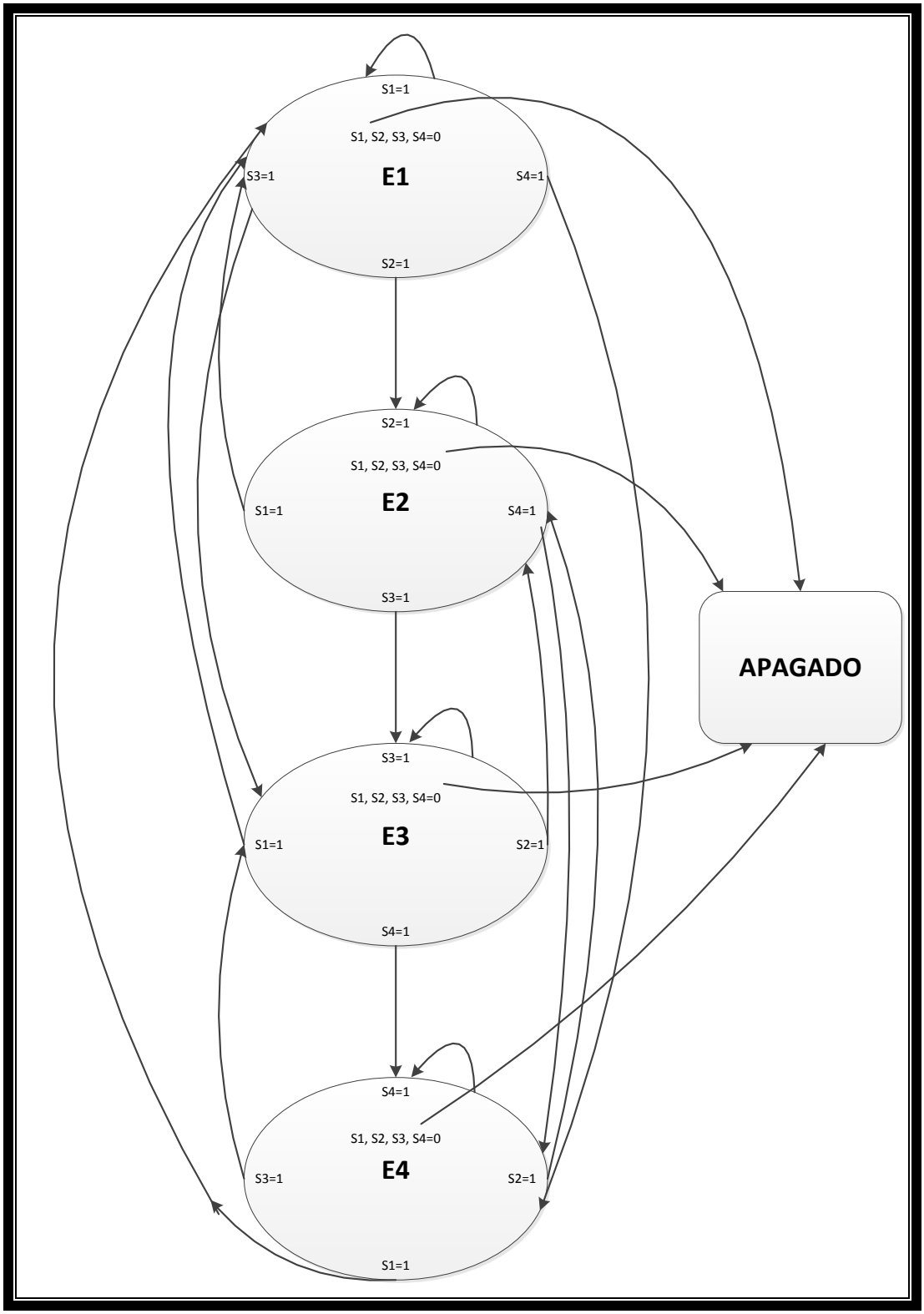


Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.78. Funcionamiento del semáforo y división en estados

Podemos observar en la figura IV.78 que en cada uno de los estados existe un tiempo en el cual los cuatro semáforos van a estar en rojo, esto es para evitar una colisión en el momento del intercambio de la luz roja a verde.

Para nuestra aplicación se han dividido los semáforos en estados, cada estado posee la información de los cuatro semáforos, estos estados se los va asociar a la electrónica digital básica que es una máquina de estados.

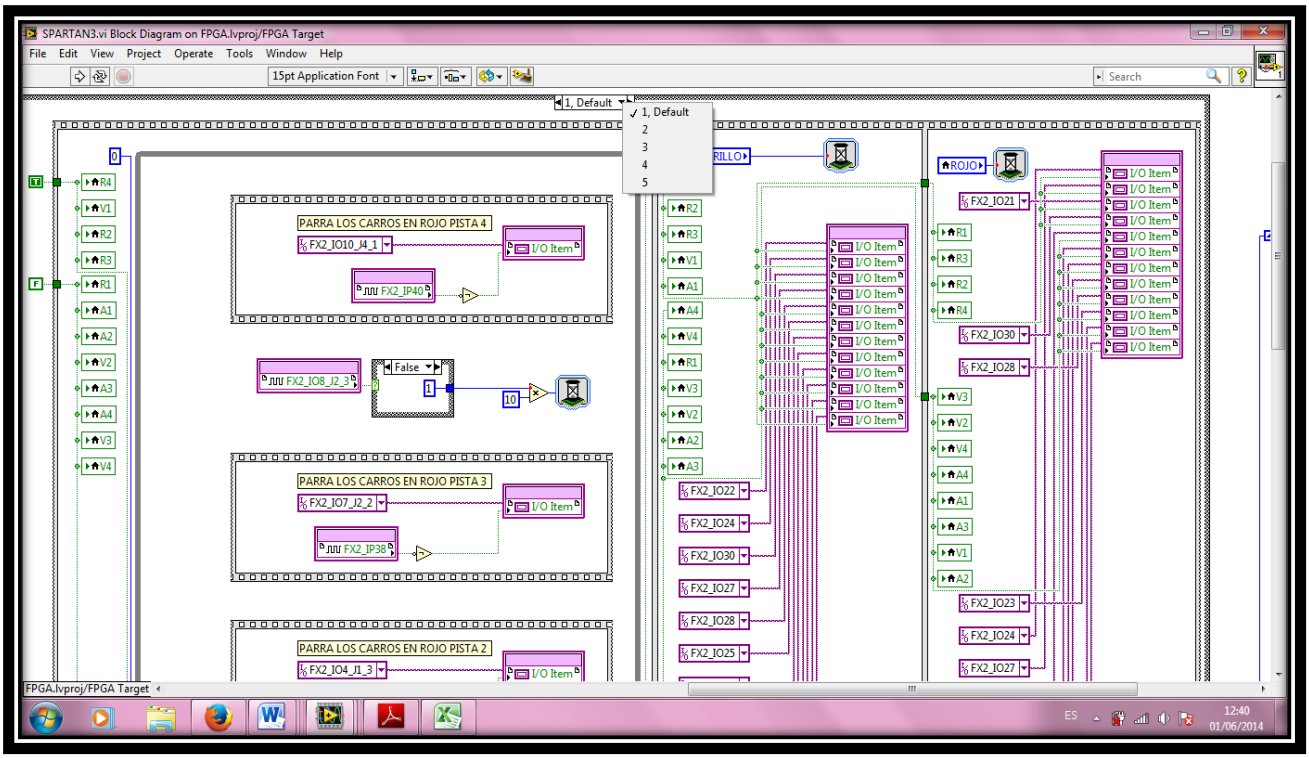


Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.79. Máquina de Estados

En la figura IV.79 se evidencia la máquina de estados implementada, la cual nos permite desplazarnos a cualquier estado en cualquier momento sin la necesidad de seguir una secuencia establecida. Cuando no tenemos ninguna señal en la S, nuestra máquina de estados se apaga, al apagarse la misma los últimos valores registrados son las luces rojas.

Una vez comprendido el funcionamiento y la automatización del semáforo procedemos a implementar la máquina de estados en Labview, como se observa en la figura IV.80.



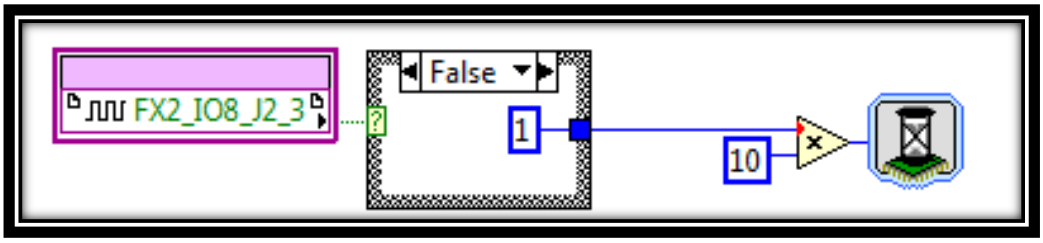
Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.80. Máquina de estados implementada en LabView

Para la implementación nos ayudamos de la estructura Flat Sequence, la misma tiene como función activar las diferentes luces de cada semáforo.

Se activó un control de tiempo, como se muestra en la figura IV.81,este trabajará en función a la cantidad de autos detectados en cada pista,

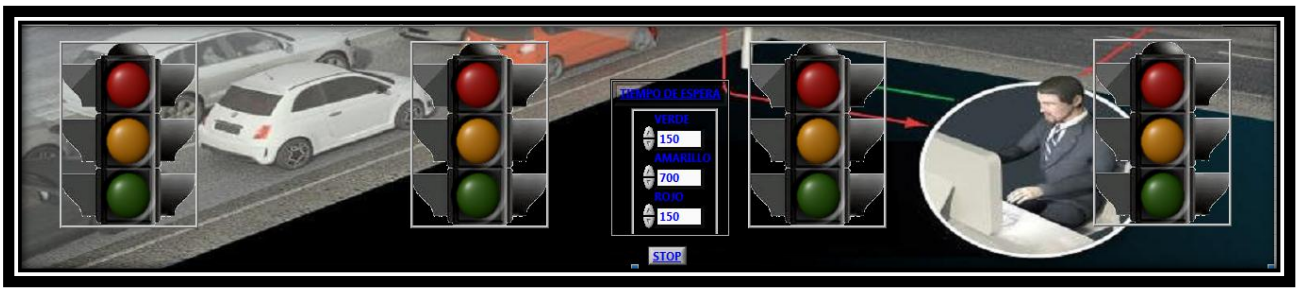
estableciendo un tiempo mínimo para el caso que exista un solo auto en la pista, y el doble de este tiempo para el máximo de autos.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.81. Asignación del tiempo por la cantidad de autos

Una vez desarrollado el código se procedió hacer la interfaz gráfica, la misma se puede observar en la figura IV.82.

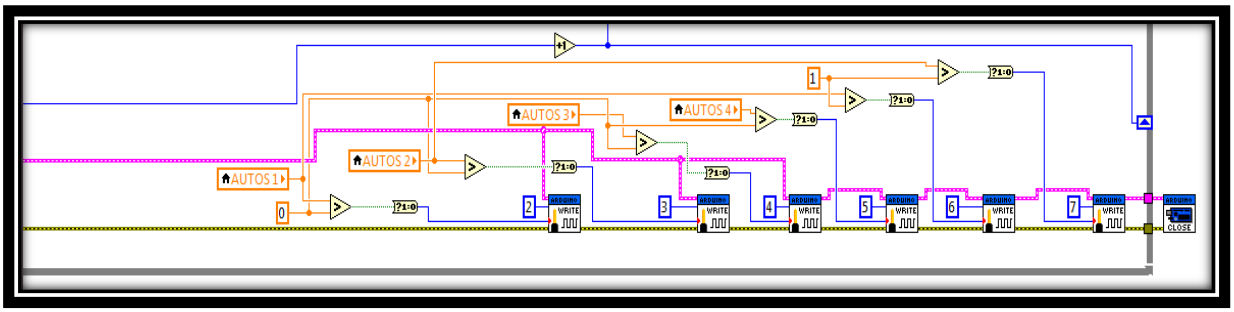


Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.82. Interfaz gráfica de los semáforos en la FPGA

4.2.3. UNIÓN DE LA VISIÓN ARTIFICIAL CON LA FPGA

Uno de los pro al desarrollar la aplicación es la transferencia de datos desde la aplicación de reconocimiento de patrones hacia la aplicación de la FPGA, en nuestro caso práctico por motivos de falta de memoria de la tarjeta Spartan 3E, no se procedió hacer una comunicación directa entre la computadora y la FPGA, esta conexión se la realizó mediante la ayuda de una tarjeta ARDUINO 1, la cual solamente envía bits, los mismos que la FPGA recibe como señales digitales. Dicho proceso se puede observar en la figura IV.83.

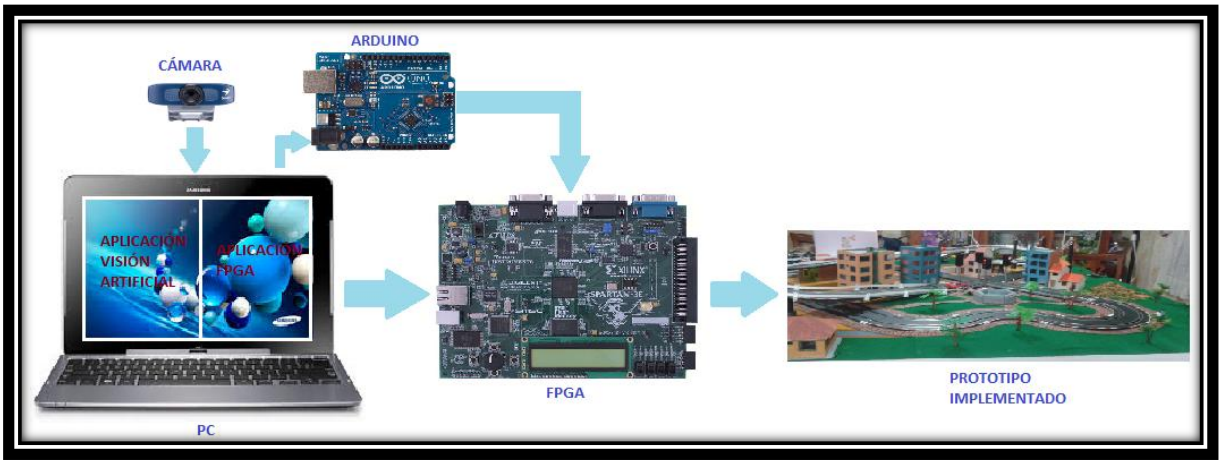


Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.83. Transmisión de datos desde la aplicación de reconocimiento de patrones hacia la FPGA mediante Arduino

4.2.4. ELABORACIÓN DEL ESQUEMA DE FUNCIONAMIENTO DEL PROTOTIPO Y LAS APLICACIONES

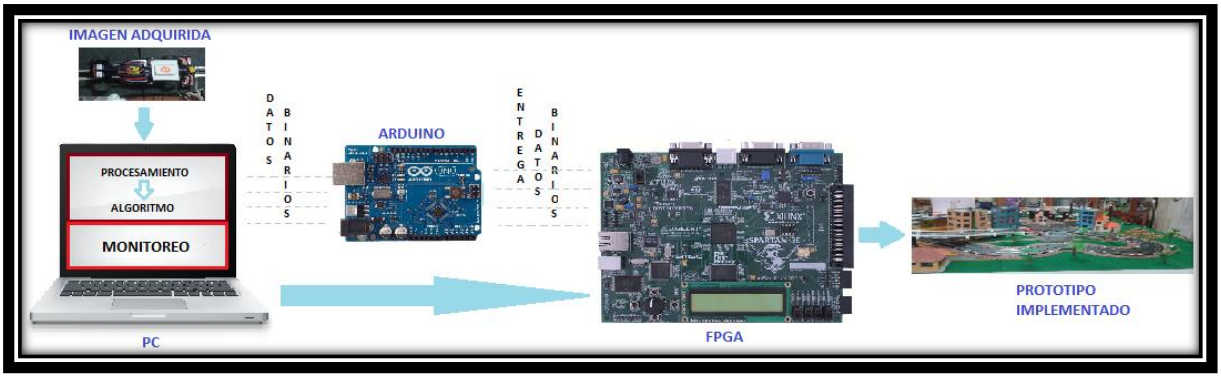
En la implementación nos encontramos con dos aplicaciones de diferentes tecnologías dentro de un mismo proyecto, que nos conllevan a una solución al finalizar el estudio comparativo de los algoritmos, para lo cual el flujo de los datos desde la adquisición hasta la proyección de los datos resultantes se detalla en la figura IV.84.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.84. Esquema de funcionamiento

Como podemos observar en la figura IV.85 primero se adquiere la imagen, la cual se procesa en el computador, y posteriormente los datos adquiridos de la imagen son entregados a la aplicación de la FPGA.



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura IV.85. Proceso para adquisición y entrega de datos

CAPÍTULO V

PRUEBAS Y RESULTADOS

Luego de implementar el prototipo se procede a realizar las respectivas pruebas para comprobar el funcionamiento del algoritmo implementado en LabView, para el control de un semáforo inteligente mediante FPGA, y minimizar la congestión vehicular.

El objetivo del diseño de un semáforo inteligente es crear un sistema que con poca o sin interacción humana se capaz de tomar decisiones de manera autónoma para controlar el tráfico vehicular.

5.1. FUNCIONAMIENTO DEL DISEÑO

El presente sistema por medio de una cámara y del algoritmo desarrollado detecta los autos presentes en cada una de las pistas y realiza el conteo de los mismos, la información obtenida es transmitida hacia la FPGA, la misma analiza los datos y toma decisiones para controlar los semáforos, los cuales indican el curso que debe tomar el tráfico.

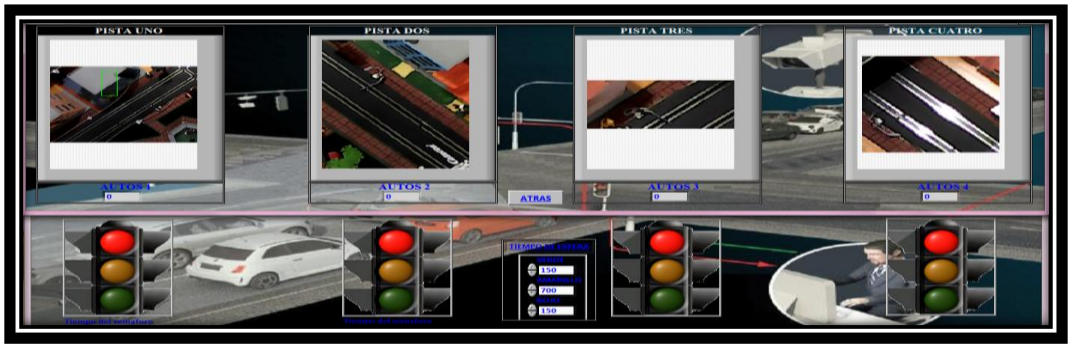
5.2. PRUEBAS DE SIMULACIÓN

Una de las primeras pruebas realizadas fue escoger el color del patrón a ser utilizado, se probó con una serie de colores y en base al método de prueba y error se seleccionó el color azul obscuro como patrón, debido a que éste no tiene similitud con algún otro color presente en el escenario.

5.2.1. SEMÁFORO INTELIGENTE

Para comprobar que el prototipo de semáforo implementado es inteligente se utilizó el método inductivo experimental, para lo cual se tomó 5 muestras de manera aleatoria.

PRUEBA 1

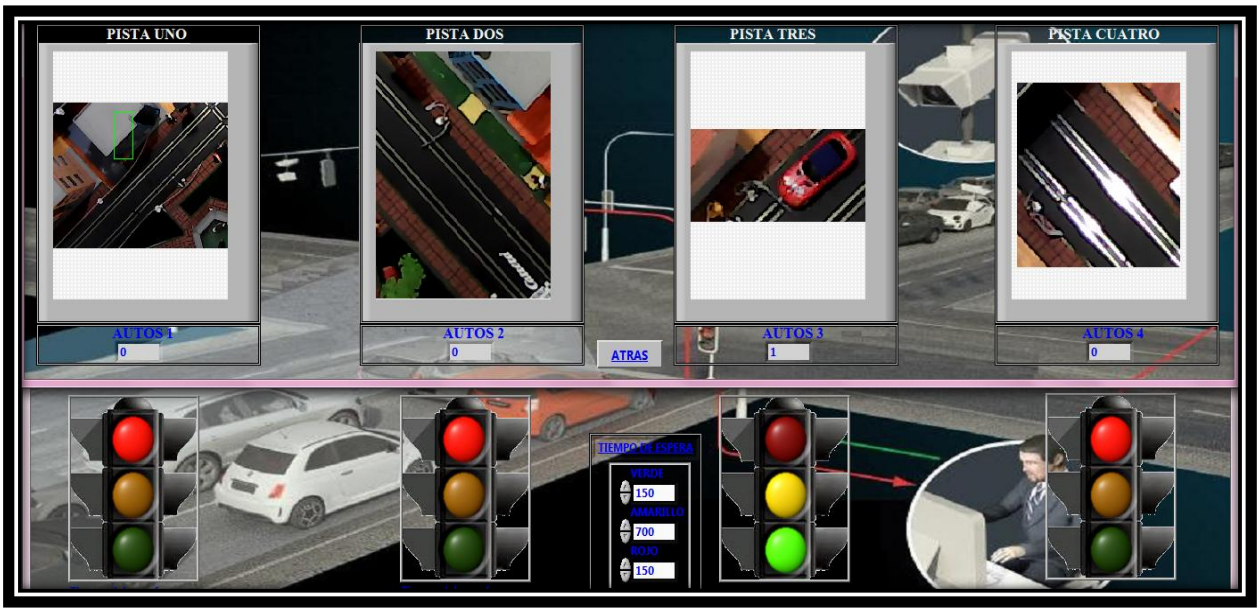


Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura V.86. Semáforos desactivados

El resultado del algoritmo no detecta ningún auto en las pistas, como se puede observar en la figura V.86 si no existe presencia de autos los semáforos no funcionan y se mantienen en el estado de luz roja.

PRUEBA 2

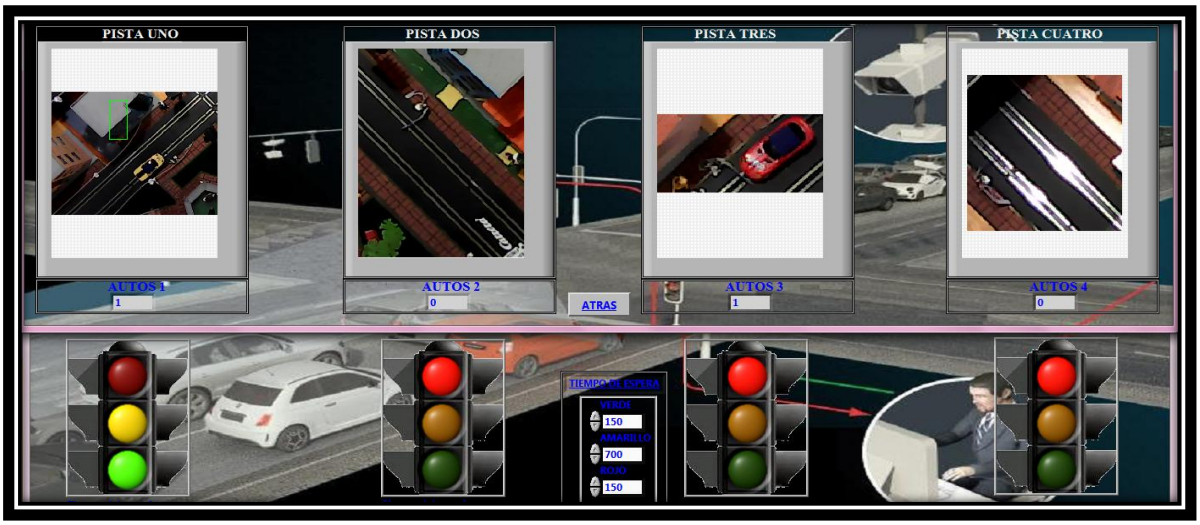


Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura V.87. Activación del semáforo 3

El resultado del algoritmo detecta un auto en la pista tres, como se puede observar en la figura V.87 cuando existe presencia de autos el semáforo se activa.

PRUEBA 3

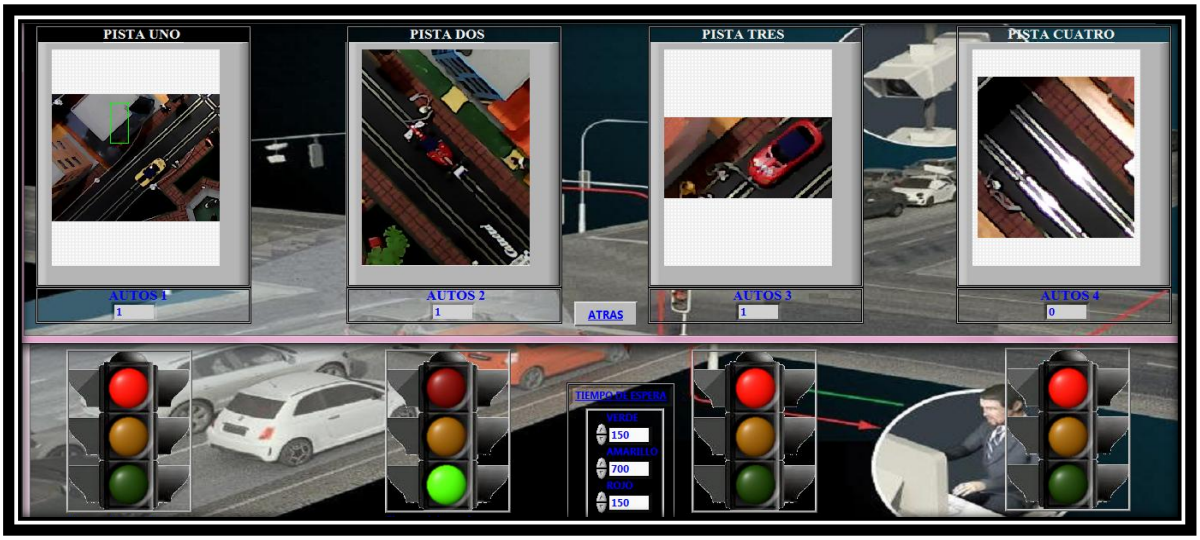


Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura V.88. Activación del semáforo uno

El resultado del algoritmo detecta presencia de autos en las pistas uno y tres, como se puede observar en la figura V.88 el semáforo tres ya se encuentra activado, por lo tanto se procede activar el semáforo uno.

PRUEBA 4

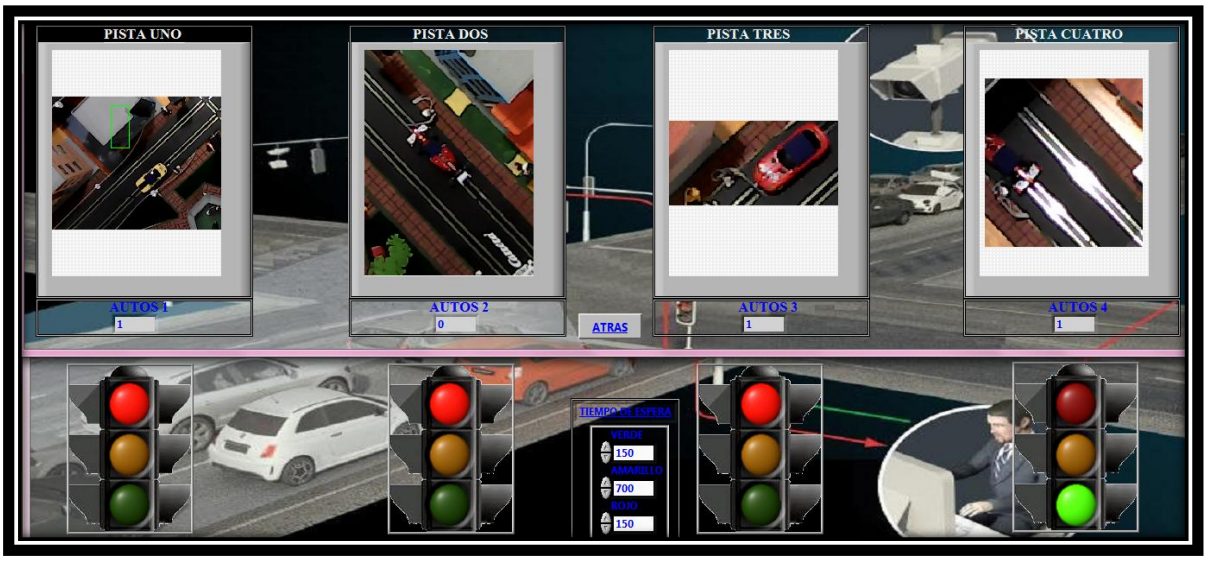


Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura V.89. Activación del semáforo dos

El resultado del algoritmo identifica tres autos, ubicados en las pistas uno, dos y tres. El algoritmo determina que se debe activar el semáforo dos como se observa en la figura V.89, el semáforo uno y tres ya se encuentran activados.

PRUEBA 5



Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Figura V.90. Activación del semáforo cuatro

El resultado del algoritmo identifica cuatro autos, ubicados en las pistas uno, dos, tres y cuatro. El algoritmo determina que se debe activar el semáforo cuatro como se observa en la figura V.90, ya que previamente los otros semáforos ya se activaron.

ANÁLISIS DE RESULTADOS

Mediante los resultados obtenidos, los cuales se detallan en la tabla V.17, se pudo comprobar que el prototipo de semáforo implementado es inteligente, si el algoritmo no detecta presencia de autos los semáforos se encuentran desactivados, esto quiere decir que permanecen en el estado de luz roja, mientras

que si el algoritmo detecta un auto en cualquiera de las pistas el semáforo correspondiente se activa.

Tabla V.17. Muestreo de datos

No. MUESTRA	PISTA 1		PISTA 2		PISTA 3		PISTA 4	
	SEMÁFORO 1	No. AUTOS	SEMÁFORO 2	No. AUTOS	SEMÁFORO 3	No. AUTOS	SEMÁFORO 4	No. AUTOS
1	ROJO	0	ROJO	0	ROJO	0	ROJO	0
2	ROJO	0	ROJO	0	VERDE	1	ROJO	0
3	VERDE	1	ROJO	0	ROJO	0	ROJO	0
4	ROJO	0	VERDE	1	ROJO	0	ROJO	0
5	ROJO	0	ROJO	0	ROJO	0	VERDE	1

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

5.2.2. MINIMIZAR EL TRÁFICO

Para comprobar que el prototipo implementado funciona y minimiza el tráfico, se utilizó el Método Estadístico de Diseño Completamente al Azar, en el cual el autor asigna las unidades experimentales a los tratamientos al azar, su única limitación es que el número de observaciones recolectadas en cada tratamiento deben ser iguales y mínimo 8, con un nivel de confianza del 99%.

- **POBLACIÓN Y MUESTRA**

Población: son los sistemas utilizados, sistema tradicional y sistema inteligente de semaforización.

Muestra: para minimizar el tráfico la muestra estuvo formada por dos parámetros que son:

- **NÚMERO DE AUTOS**

  El número de autos se refiere a la cantidad de autos detectados en cada pista por el algoritmo.

- **TIEMPO DE ESPERA**

  El tiempo de espera se refiere al tiempo que se mantiene el semáforo en verde, el cual se determina en función de la cantidad de autos.

Se realizó 8 tomas de muestras de cada parámetro.

Para calcular n cada sistema representa un tratamiento y cada toma de muestra equivale a una repetición.

$$n = (t * r)$$
$$n = (2 * 8)$$
$$n = 16$$

A continuación se detalla la tabla de muestras tomadas completamente al azar de los dos sistemas utilizados.

SISTEMA TRADICIONAL

Tabla V.18. Muestreo de datos

No. PRUEBA	SEMÁFORO 1		SEMÁFORO 2		SEMÁFORO 3		SEMÁFORO 4	
	No. AUTOS	TIEMPO ESPERA (ms)	No. AUTOS	TIEMPO ESPERA (ms)	No. AUTOS	TIEMPO ESPERA (ms)	No. AUTOS	TIEMPO ESPERA (ms)
1	2	60	2	60	2	60	1	60
2	1	60	1	60	1	60	1	60
3	3	60	0	60	0	60	0	60
4	2	60	2	60	0	60	1	60
5	3	60	1	60	2	60	0	60
6	1	60	1	60	1	60	1	60

7	1	60	0	60	0	60	0	60
8	2	60	3	60	1	60	1	60

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

SISTEMA INTELIGENTE

Tabla V.19. Muestreo de datos

No. PRUEBA	SEMÁFORO 1		SEMÁFORO 2		SEMÁFORO 3		SEMÁFORO 4	
	No. AUTOS	TIEMPO ESPERA (ms)	No. AUTOS	TIEMPO ESPERA (ms)	No. AUTOS	TIEMPO ESPERA (ms)	No. AUTOS	TIEMPO ESPERA (ms)
1	1	10	0	0	0	0	0	0
2	1	10	1	10	1	10	0	0
3	2	20	1	10	0	0	1	10
4	3	30	3	30	0	0	0	0
5	2	20	1	10	1	10	1	10
6	2	20	1	10	2	20	1	10
7	3	30	1	10	2	20	1	10
8	1	10	2	20	1	10	1	10

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

En la presente tabla se detalla las medias generales de las muestras tomadas.

Tabla V.20. Medias de los datos

	SEMÁFORO 1		SEMÁFORO 2		SEMÁFORO 3		SEMÁFORO 4	
	No. AUTOS	TIEMPO ESPERA (ms)	No. AUTOS	TIEMPO ESPERA (ms)	No. AUTOS	TIEMPO ESPERA (ms)	No. AUTOS	TIEMPO ESPERA (ms)
PROMEDIO SISTEMA TRADICIONAL	15	60	10	60	7	60	5	60
PROMEDIO SISTEMA INTELIGENTE	15	18,75	10	12,5	7	8,75	5	6,25

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

❖ ANÁLISIS VARIANZA

Luego de calcular las medias respectivas para cada parámetro de cada uno de los sistemas, se realiza el análisis de la varianza, este análisis se lo realizó solo para un semáforo, ya que para los otros casos el procedimiento va a ser repetitivo.

♣ NÚMERO DE AUTOS

Planteamiento Hipótesis

Ho: La media de número de autos en el sistema tradicional es igual a la media de número de autos en el sistema inteligente.

$$H_o : \mu_1 = \mu_2$$

Ha: La media de número de autos en el sistema tradicional no es igual a la media de número de autos en el sistema inteligente.

$$H_a : \mu_1 \neq \mu_2$$

Cálculo Varianza

Tabla V.21. Varianza para número de autos

SISTEMAS	MUESTRAS	SUMA	PROMEDIO	VARIANZA
TRADICIONAL	8	15	1,875	0,6964286
INTELIGENTE	8	15	1,875	0,6964286

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Análisis ANOVA

Tabla V.22. ANOVA de precisión

FUENTES DE VARIACIÓN	GRADOS DE LIBERTAD	SUMA DE CUADRADOS	PROMEDIO DE LOS CUADRADOS	F CALCULADA	F TABULADA
Entre tratamientos	1	0	0	0	8,86
Dentro de tratamientos	14	28,125	2,009		
Total	15	28,125			

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

INTERPRETACIÓN

Al terminar el análisis de varianza, el valor de F Calculada es menor que el de F Tabulada, por lo tanto se acepta la hipótesis nula, el número de autos es igual en los dos sistemas.

♣ **TIEMPO DE ESPERA (ms)**

Planteamiento Hipótesis

Ho: La media de tiempo de espera en el sistema tradicional es igual a la media de tiempo de espera en el sistema inteligente.

$$H_o : \mu_1 = \mu_2$$

Ha: La media de tiempo de espera en el sistema tradicional no es igual a la media de tiempo de espera en el sistema inteligente.

$$H_a : \mu_1 \neq \mu_2$$

Cálculo Varianza

Tabla V.23. Varianza para tiempo de espera

SISTEMAS	MUESTRAS	SUMA	PROMEDIO	VARIANZA
TRADICIONAL	8	480	60	0
INTELIGENTE	8	150	18,75	69,642857

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Análisis ANOVA

Tabla V.24. ANOVA de tiempo de espera

FUENTES DE VARIACIÓN	GRADOS DE LIBERTAD	SUMA DE CUADRADOS	PROMEDIO DE LOS CUADRADOS	F CALCULADA	F TABULADA
Entre tratamientos	1	6806,25	6806,25	43,436	8,86
Dentro de tratamientos	14	2193,75	156, 696		
Total	15	9000			

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

INTERPRETACIÓN

Al terminar el análisis de varianza, el valor de F Calculada es mayor que el de F Tabulada, por lo tanto se rechaza la hipótesis nula y se decidió que el tiempo de espera en el sistema inteligente es menor.

A través de los resultados obtenidos se realizó un cuadro comparativo en el cual se describe las ventajas del sistema inteligente sobre el tradicional.

Tabla V.25. Cuadro comparativo

SISTEMA TRADICIONAL	SISTEMA INTELIGENTE
Tiempo de espera fijo.	•Tiempo de espera variable, dependiendo el caso dicho tiempo se reduce, ya que este actúa en función de la cantidad de autos.
Funcionan en base a una secuencia, por lo cual las luces siempre se encuentran en funcionamiento sin tomar en cuenta el tráfico.	•Si no existe presencia de autos las luces siempre estarán en rojo, si se detecta presencia de autos el semáforo empieza a funcionar en base al tráfico existente.
No realiza conteo ni detección de los autos.	•Detecta y realiza el conteo de los autos que transitan cada una de las vías, a través de esta informacón el sistema puede tomar decisiones para reducir el congestionamiento.

Fuente: Gabriela M. Mafla M., Aldiniver J. Ortiz E. (Autores)

Al examinar los resultados obtenidos de cada sistema como se muestra en la tabla V.20, se pudo comprobar que el sistema inteligente mediante parámetros de evaluación (número de autos y tiempo de espera), minimizó la congestión vehicular. Este sistema mediante el algoritmo de coincidencia de colores actúa de manera inteligente, reduciendo notablemente los tiempos de espera.

CONCLUSIONES

- Al realizar el estudio comparativo de los algoritmos de reconocimiento de patrones propuestos, mediante parámetros de evaluación, se obtuvo como resultado que el algoritmo adecuado para implementar un prototipo de semáforo inteligente mediante FPGAs es el algoritmo de coincidencia de colores que cuenta con una precisión del 100% y un tiempo de respuesta de 3ms.

- No todos los algoritmos para el reconocimiento de patrones se utilizan para aplicaciones específicas, sino ciertos algoritmos que cumplen con especificaciones del diseñador, esto es debido a las características de los patrones y del tipo de reconocimiento del algoritmo que hace ineficientes a ciertos algoritmos en determinadas aplicaciones.

- La ejecución de las operaciones en la FPGA se realizan de forma paralela, esto permite aumentar la velocidad de procesamiento de datos.

- Al momento del reconocimiento de los patrones debemos tener en cuenta las características de la cámara y la intensidad lumínica, debido a que son factores predominantes que afectan directamente en el proceso, los cuales si no cumplen con los requerimientos necesarios pueden dar resultados erróneos y nulos.

- La eficiencia de la tarjeta Spartan 3E depende mucho de la aplicación, el tipo de software a utilizar para la programación de la misma, y el tipo de controlador que provea el distribuidor o los distribuidores asociados a la tarjeta.

- Podemos dar soluciones a procesos complejos o industriales mediante el uso de conceptos de electrónica básica como son las máquinas de estado, que nos permitió dar solución de una manera eficiente y rápida a esta aplicación.

- El resultado del presente trabajo de investigación fue la creación de un sistema inteligente para el control de los semáforos, este actúa en función a la cantidad de autos, permitiendo de esta manera minimizar los tiempos de espera y reducir el tráfico vehicular.

RECOMENDACIONES

- Los algoritmos seleccionados para el estudio comparativo están sujetos a la desventaja de la complejidad computacional, convirtiéndose en un punto de gran importancia el tiempo de cómputo en las máquinas convencionales. Por lo cual se debe mejorar la complejidad de los algoritmos o conseguir una mayor capacidad computacional.

- Para trabajar con tarjetas FPGA y cámaras en Labview, se debe verificar que estas sean compatibles con dicho software.

- Para aplicaciones con FPGA se debe desarrollar entornos gráficos lo más simples posibles, o en su caso solo código para no saturar la memoria y evitar errores al momento de compilar por falta de memoria.

- Al comunicar dos tipos de tarjetas mediante bits o unos y ceros, verificar el voltaje que soportan y suministran las mismas, para evitar averías.

- Para sustituir una cámara de una aplicación específica se debe tomar en cuenta la resolución, velocidad de adquisición de datos y apertura del lente de la misma, debido a que si no cumple con una de estas características puede hacer que se obtenga datos erróneos.

- Como esta aplicación es experimental se recomienda usar sensores mecánicos por facilidad de manejo e instalación.

- Se recomienda probar el sistema diseñado en condiciones reales.

RESUMEN

Se diseñó e implementó un prototipo de semáforo inteligente mediante una tarjeta electrónica de arreglos de compuertas programables en campo (FPGA), para minimizar la congestión vehicular.

Para su diseño se empleó LabView 2012 ya que facilita la unión de la visión artificial con la tarjeta electrónica de arreglos de compuertas programables en campo (FPGA); en su implementación se utilizó madera MDF, pistas eléctricas, sensores de contacto, fuente de computadora, cámara, tarjeta electrónica de arreglos de compuertas programables en campo (FPGA) y Arduino.

El control automático del semáforo, se realizó con la tarjeta Spartan 3-E, la cual se encarga de procesar mediante el algoritmo de coincidencia de colores la información adquirida por la cámara.

Como resultado se obtuvo que en el sistema inteligente el tiempo de espera aproximado para 2 autos fue 18,75 ms, mientras que en el sistema tradicional el tiempo de espera aproximado para la misma cantidad de autos fue 60 ms.

Se concluye que el prototipo de semáforo inteligente diseñado e implementado, actúa en función de la cantidad de autos, disminuyendo los tiempos de espera y minimizando la congestión vehicular.

Se recomienda probar el prototipo diseñado en condiciones reales.

SUMMARY

It was designed and implemented an intelligent prototype of traffic light with an electronic card fix of programmable gate (FPGA) to minimize the traffic.

The LabView 2012 design was used because it facilitates the binding of the machine vision with electronic card fix of programmable gate (FPGA); in its implementation were used MDF wood, electric tracks, touch sensors, computer source, camera, electronic card fix of programmable gate (FPGA) and Arduino.

The automatic control of the traffic lights was performed with the Spartan 3-E card, which is responsible of processing through the matching algorithm color with the information acquired by the camera.

As result the intelligent system in lead time for 2 cars was 18.75 ms, while the traditional system in lead time for the same number of cars was 60 ms.

It concluded that the designed and implementation of the intelligent traffic light prototype, acts according to the number of cars, reducing the traffic and lead times.

It recommended try out the prototype designed in real conditions.

GLOSARIO

ADQUISICIÓN DE DATOS	Es tomar un conjunto de muestras y digitalizarlas de manera que puedan ser procesadas por una computadora.
ALGORITMO	Es un conjunto de reglas definidas, ordenadas y finitas que permiten dar solución a un problema.
ARDUINO	Es una plataforma de hardware y software libre para el desarrollo de prototipos electrónicos.
CÁMARA	Dispositivo electrónico cuya función es capturar imágenes.
COMUNICACIÓN SERIAL	Es el proceso de envío de datos de un bit a la vez.
COMUNICACIÓN USB	Es una interfaz de entrada/salida para transmisión de datos y distribución de energía.
DRIVER	Es un programa que permite la interacción entre un sistema operativo con un periférico.
EFICIENCIA	Se refiere al margen de error que se obtiene al realizar un determinado estudio.

INTELIGENCIA ARTIFICIAL	Rama de la informática que pretende desarrollar programas que permitan exhibir un comportamiento inteligente.
INTERFAZ	Interfaz de usuario es el medio que permite a una persona comunicarse con una máquina.
MONITOREO	Es una evaluación continua de una acción en progreso.
PATRÓN	Es una descripción estructural o cuantitativa de un objeto o alguna entidad de interés.
SEGMENTACIÓN	Es el proceso de dividir una imagen digital en varias partes, con el objetivo de facilitar su análisis.
SENSOR	Es un dispositivo que detecta acciones o estímulos externos y sirve para activar un mecanismo.
VISIÓN ARTIFICIAL	Es un conjunto de técnicas que permiten la adquisición, procesamiento y análisis de información obtenida a través de imágenes.
ZÓCALO	Es el soporte que un componente electrónico tiene para conectar otro sobre él, y lo realiza mediante pins o contactos.

ANEXOS

ANEXO 1

MANUAL DE USO DE LA FPGA SPARTAN-3E MEDIANTE LABVIEW

Introducción

Bienvenido a la serie de programación de una tarjeta Spartan3E Starter Kit para el uso de LabView FPGA.

Requisitos de software:

• LabView 8.5 o por encima.

• LabView 8.5 FPGA módulo.

• Tabla de inicio XUP Spartan3E: descargar de forma gratuita a partir de: https://lumen.ni.com/nicif/us/infolvfpgaxilsprtn/content.xhtml

Requisitos de hardware:

• Kit de Xilinx Spartan3E Starter:

http://www.xilinx.com/products/devkits/HW-SPAR3E-SK-USG.htm

• Manual del usuario:

www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf

Primeros pasos

Cuando se desea utilizar estos laboratorios se tiene que configurar su tarjeta. Estos laboratorios están escritos para la tarjeta SPARTAN 3E Starter Kit Xilinx, para lo cual es muy importante leer el manual del usuario de la tarjeta. Asegúrese de conectar el cable USB, enchufe el cable de alimentación y realice un reset antes de iniciar la práctica de laboratorio.

Paso 1: Iniciar LabView

El primer paso es iniciar el ambiente de LabView de National Instruments 8,5.

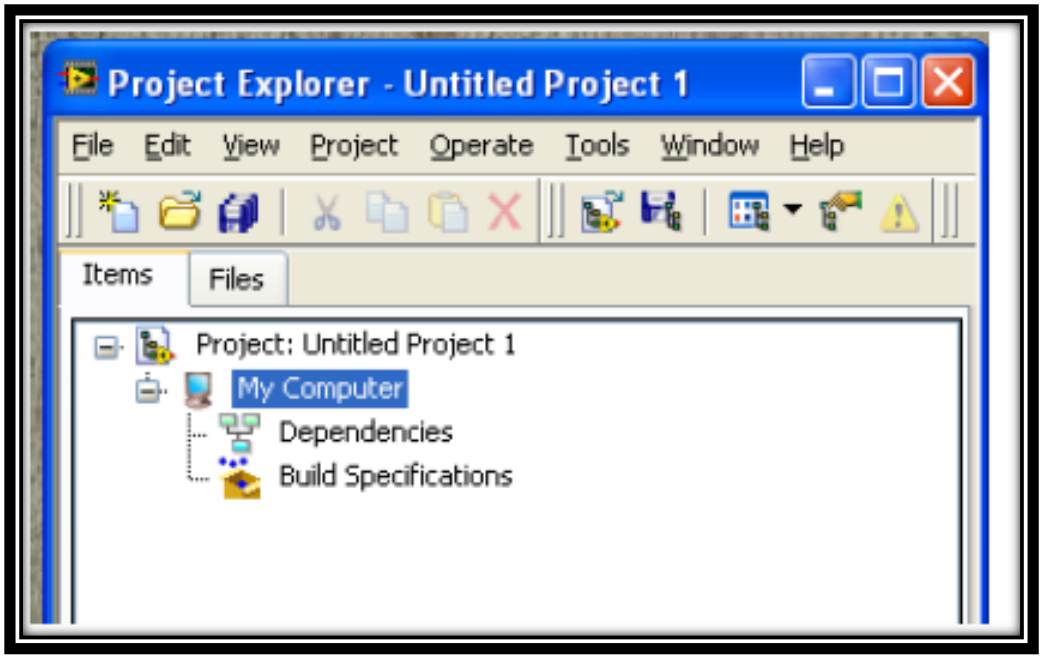
Paso 2: Crear un proyecto de LabView

Después de haber iniciado el entorno de LabView se obtiene la siguiente pantalla:

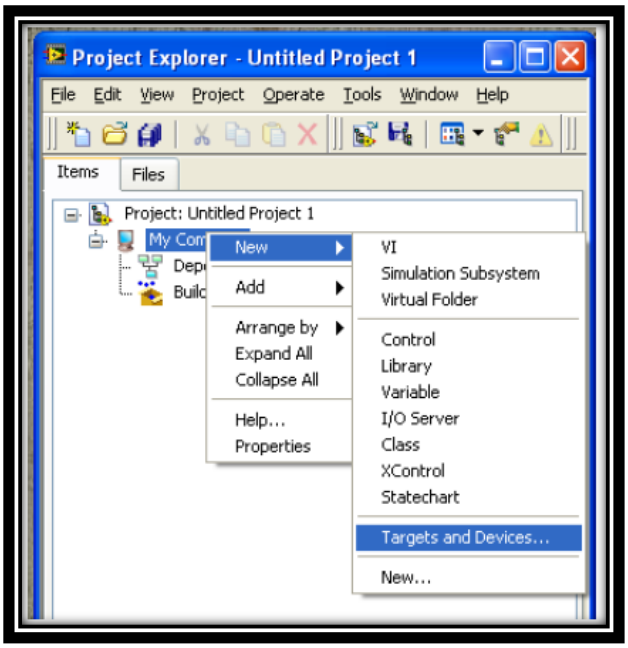
Tenemos que crear un proyecto vacío donde vamos a añadir la tabla de inicio Xilinx Spartan3E como un objeto de hardware. En un laboratorio de futuro voy a explicar cómo podemos crear un vi HOST (esta es una aplicación de LabView que se ejecuta en una PC de escritorio) que se comunica con una vi LabView FPGA.

Paso 3: Añadir la tarjeta Spartan3E como un objeto de hardware

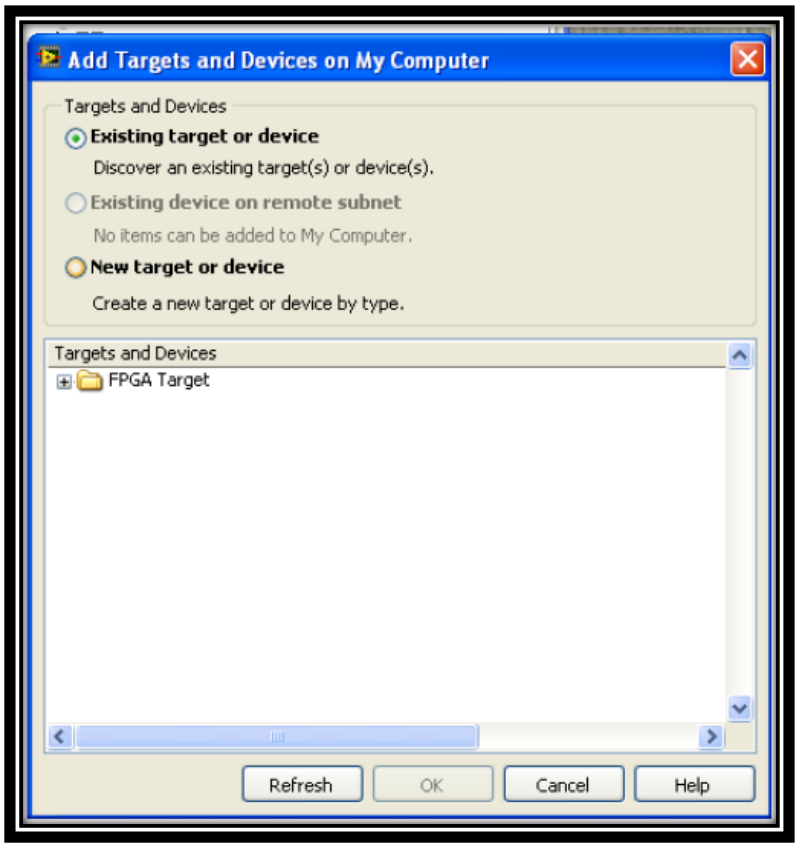
La siguiente pantalla muestra la vista Project Explorer del proyecto vacío que acabamos de crear.



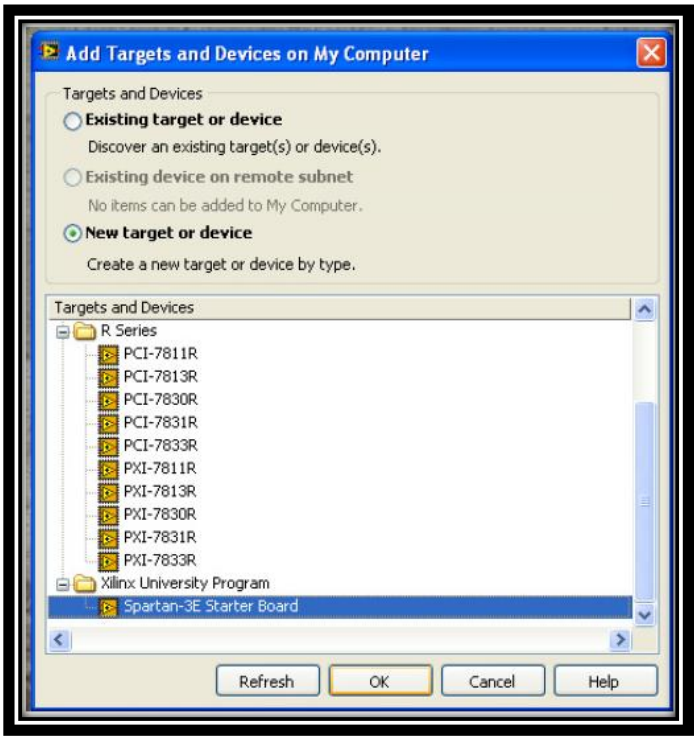
Ahora tenemos que añadir la tabla de inicio Spartan3E como un objeto de hardware. Para ello hacemos un "click derecho" en "Mi PC" en la ventana Project Explorer. Seleccionamos "Nuevo" y luego seleccionamos "Targets and devices".



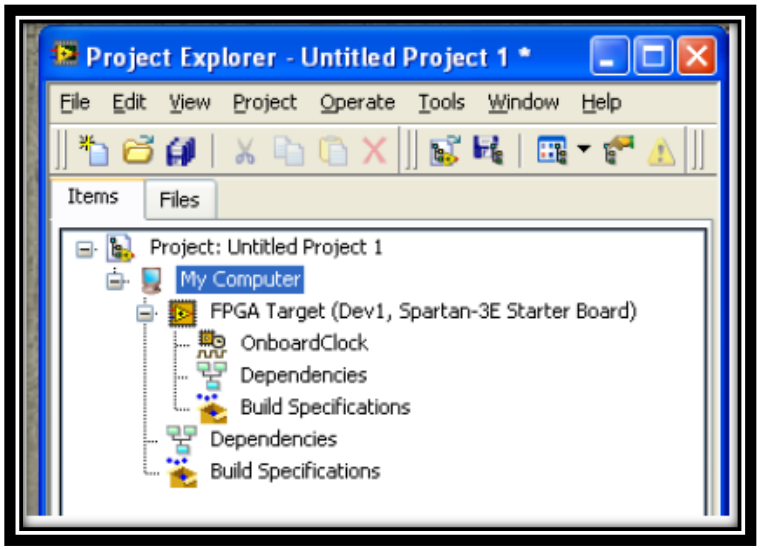
En la pantalla que aparece ahora tenemos que seleccionar la tabla de inicio XUP. Seleccione " New target or device".



Se mostrará una lista de los " Targets and Devices” ". Usted tiene que desplazarse hacia abajo hasta el final y seleccione "tarjeta Spartan-3E Starter" del mapa "Programa de Xilinx universidad".

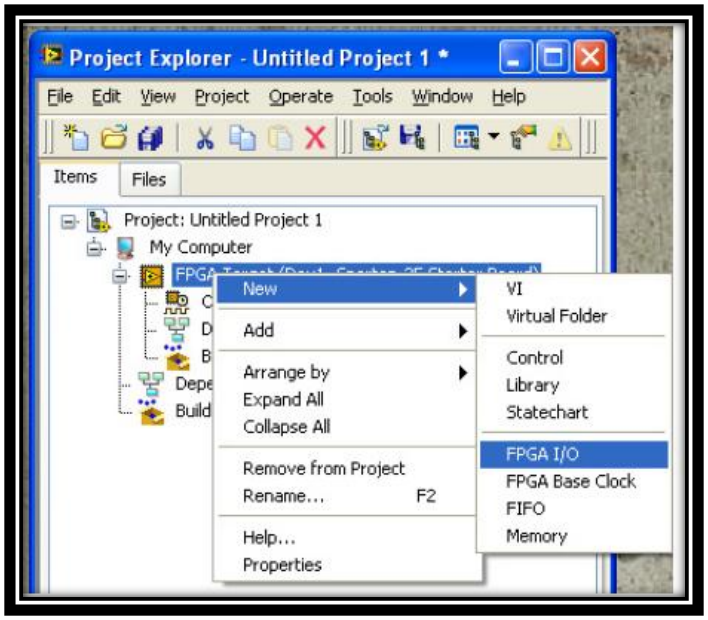


Esto agregará el objeto FPGA a la ventana Project Explorer. Veras que este objeto se coloca en "My Computer".

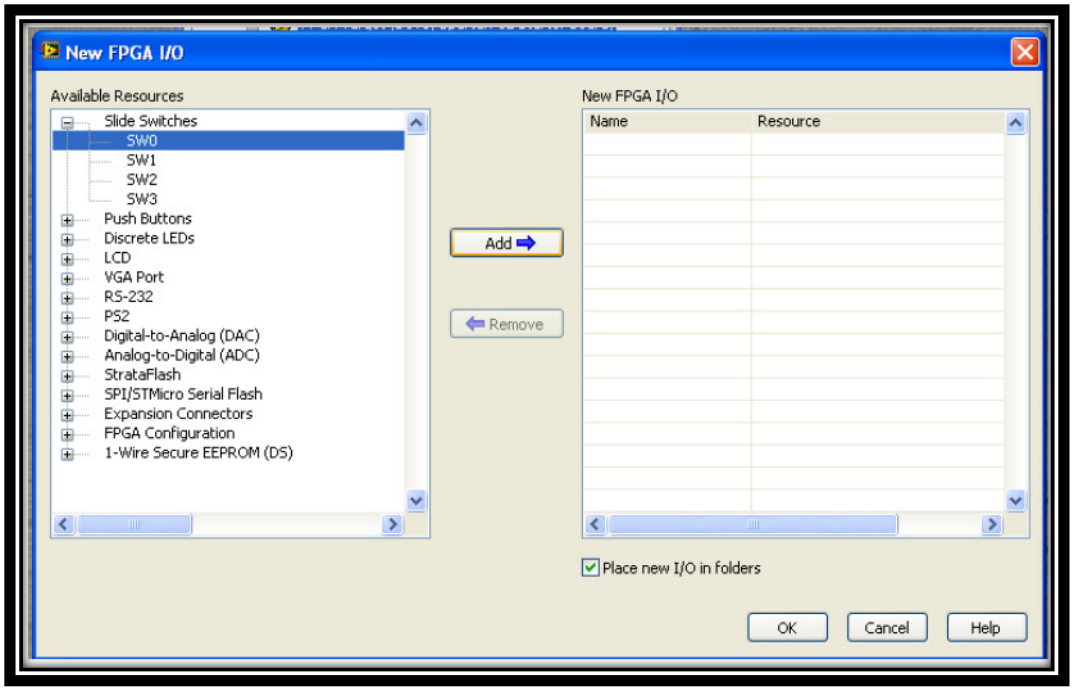


Paso 4: Agregar FPGA I / O.

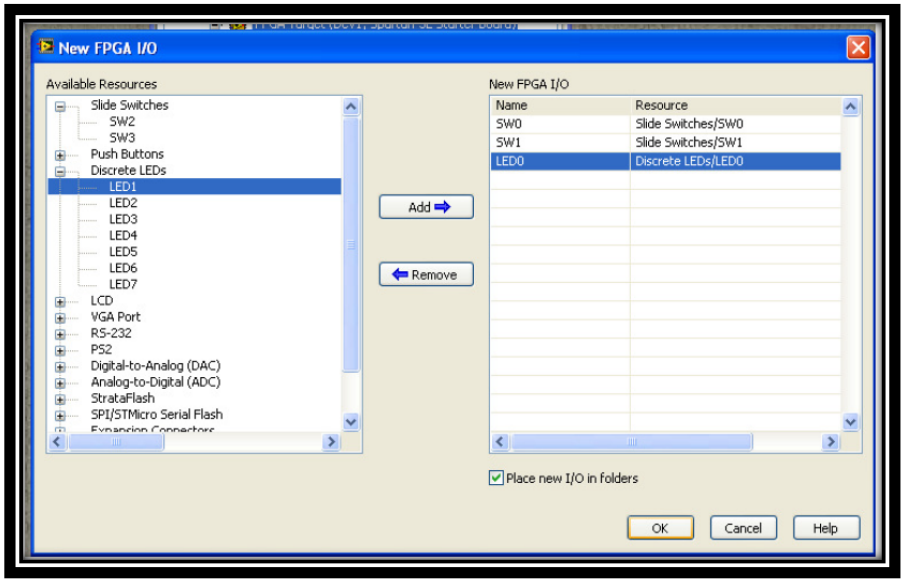
El siguiente paso que hacemos es añadir "FPGA I / O " que vamos a utilizar en este proyecto de la ventana Project Explorer. Para ello, hacemos "click derecho" en el objeto FPGA agregó en el paso 3. Seleccione "New" "FPGA de I / O".



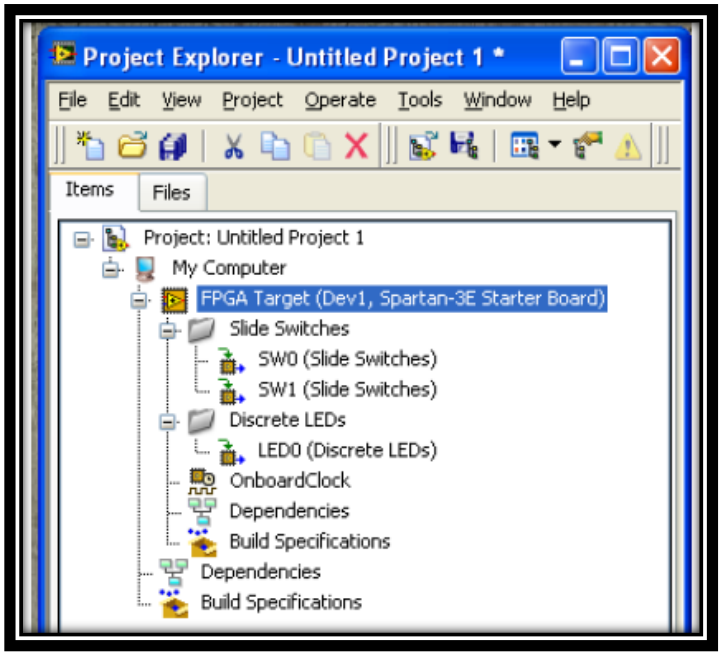
Aparecerá la siguiente pantalla.



En la parte izquierda de esta pantalla se puede ver todos los recursos que están disponibles en la tarjeta Spartan-3E. Para agregar FPGA I/ O al proyecto tenemos que seleccionar “Add” en la ventana izquierda. Ahora la FPGA I / O que desea agregar tiene que aparecer en la ventana en el lado derecho. Para este proyecto hay que añadir SW0, SW1 y LED0 al proyecto.

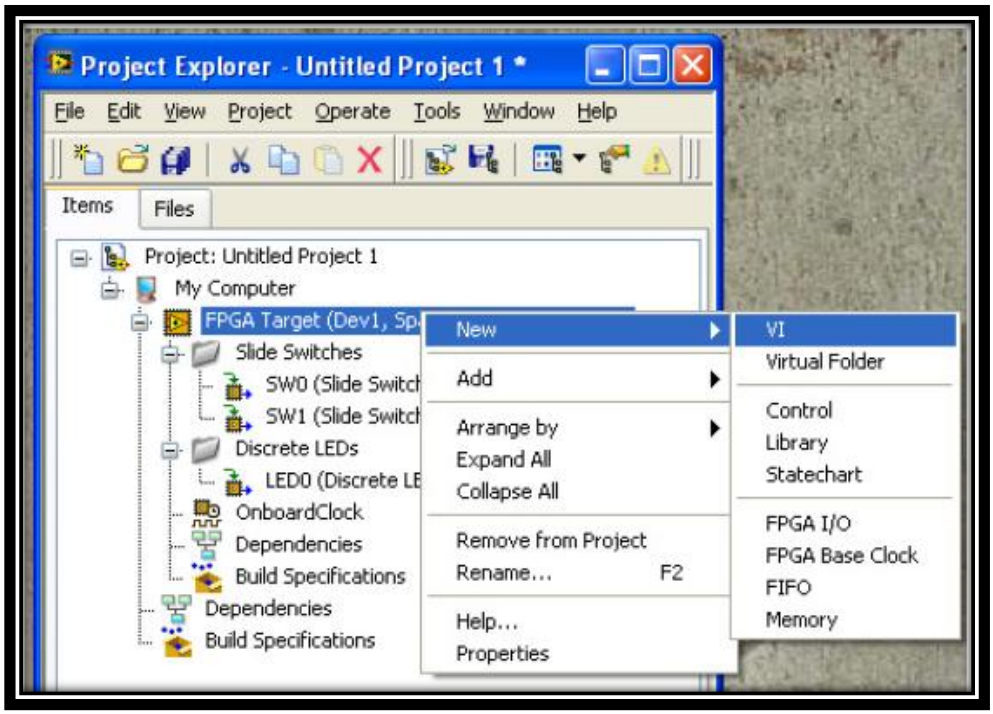


Si regresamos a la ventana " Project Explorer " vemos la FPGA de E / S que hemos añadido. Con este I / O vamos a implementar una función Exclusive-OR.

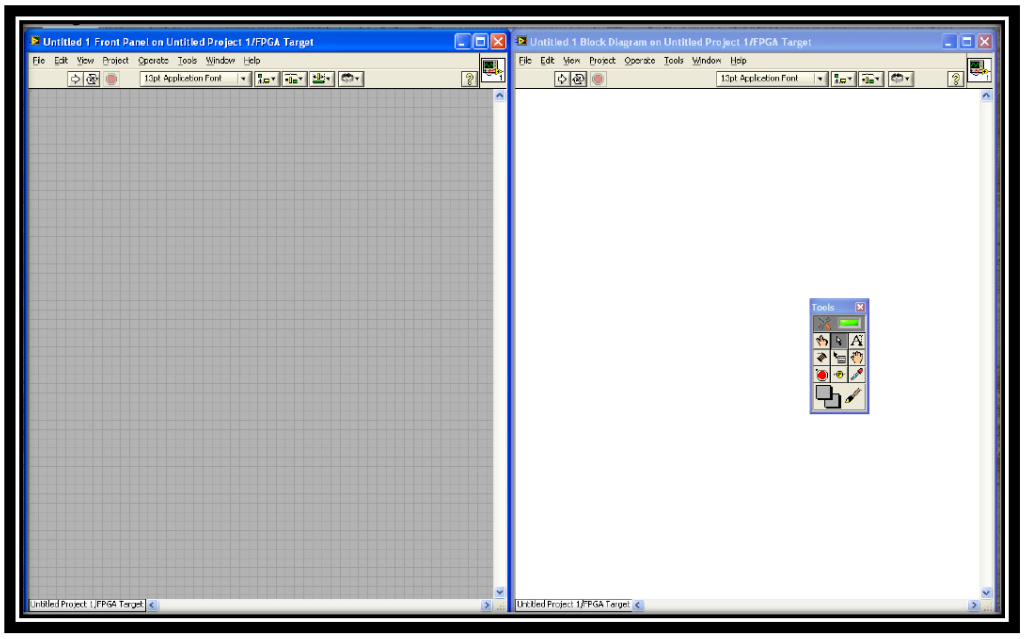


Paso 5: Creación de la FPGA VI

El siguiente paso es la creación de un programa de hardware que se ejecuta en la tarjeta Spartan-3E Xilinx. Para esto hacemos de nuevo un "botón derecho del ratón" en el objeto FPGA en la ventana " Project Explorer". Seleccione VI "Nuevo". Asegúrese de que usted ha hecho clic derecho en el objetivo FPGA y no en "My Computer", porque de lo contrario va a crear un VI que no se ejecuta en hardware (en la FPGA), pero si en el equipo.

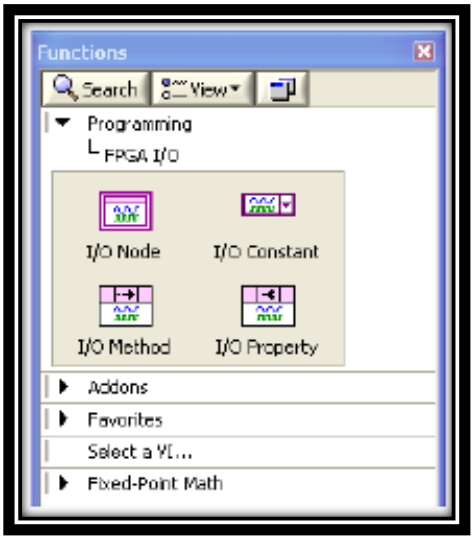


Este paso abrirá el Panel Frontal y el Diagrama de bloques de la FPGA VI que acaba de empezar a crear. El código que estaremos implementando es una función O exclusiva. La ventana que siempre uso en LabView es la "ventana izquierda y derecha". Puede seleccionar esta opción, vaya a " Windows" y luego seleccionar " Tile Left and Right". Cuando usted hizo todos los pasos a la derecha tiene que ver en la esquina izquierda de cualquiera del panel frontal o Diagrama de bloques del nombre del VI que ha creado con la tarjeta FPGA detrás de él. Esto demuestra que la VI que se está creando está dirigido al objeto FPGA.

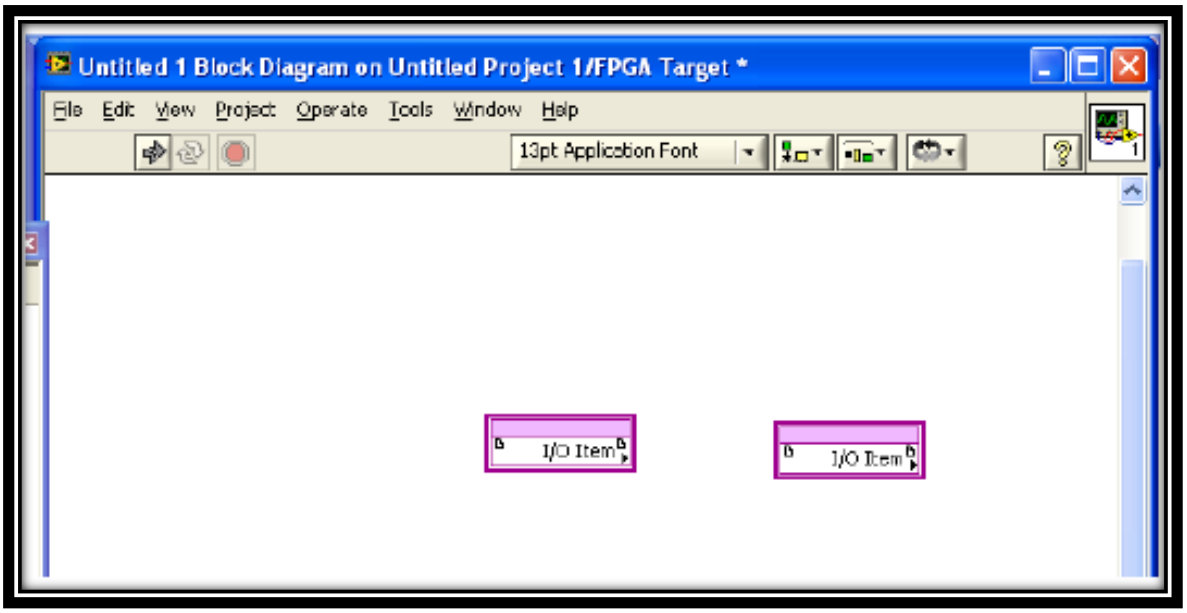


Ahora vamos a echar un vistazo a la " Functions Palette ", que se puede ver al hacer clic en el "Block Diagram" en "View" y luego " Functions Palette ". Usted ve que las funciones disponibles son diferentes. Usted ve "FPGA de I / O", " Memory & Fifo" y "FPGA Math & Analysis". Esos son especialmente para el uso en objetos FPGA.

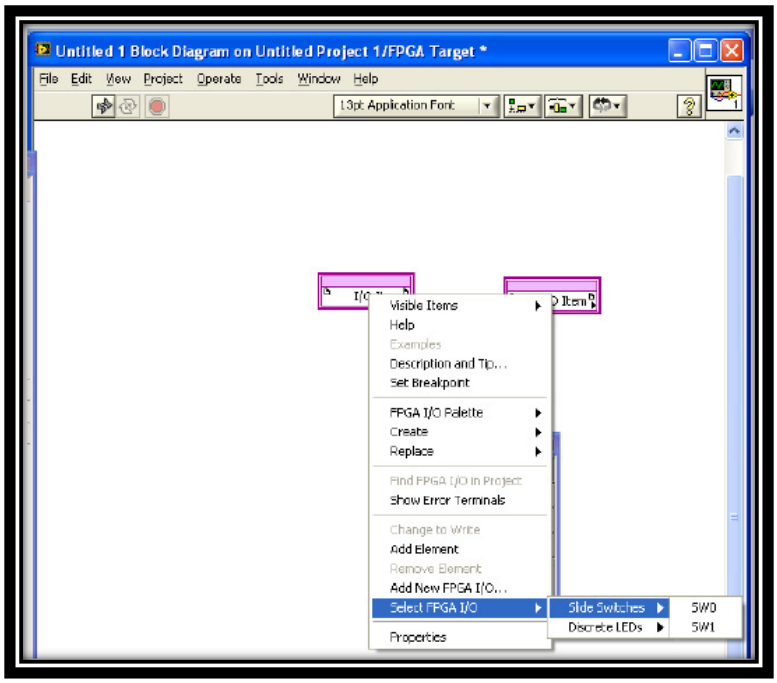
Haga click en Functions Palette en “FPGA I/O”.



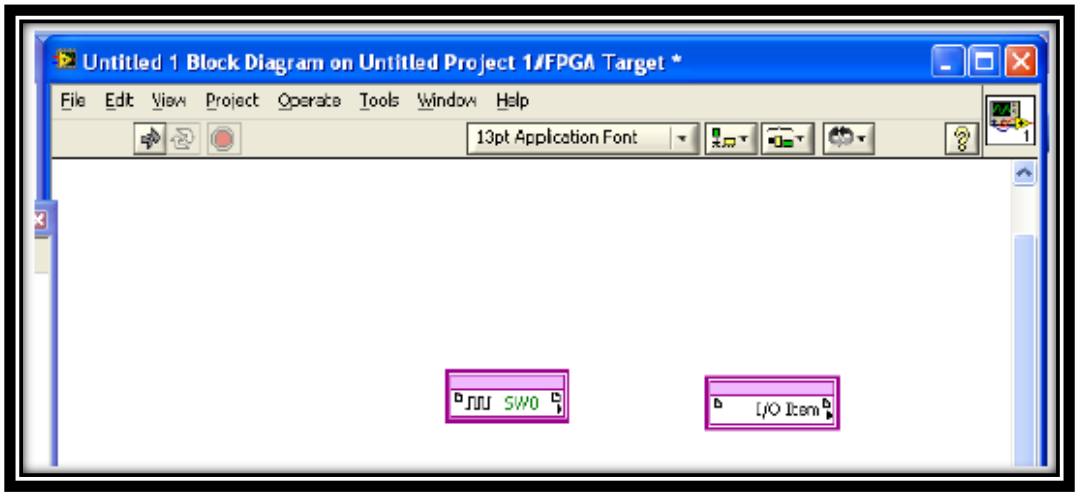
Coloque 2 I/ O de nodos en el diagrama de bloques de la LabView FPGA VI.



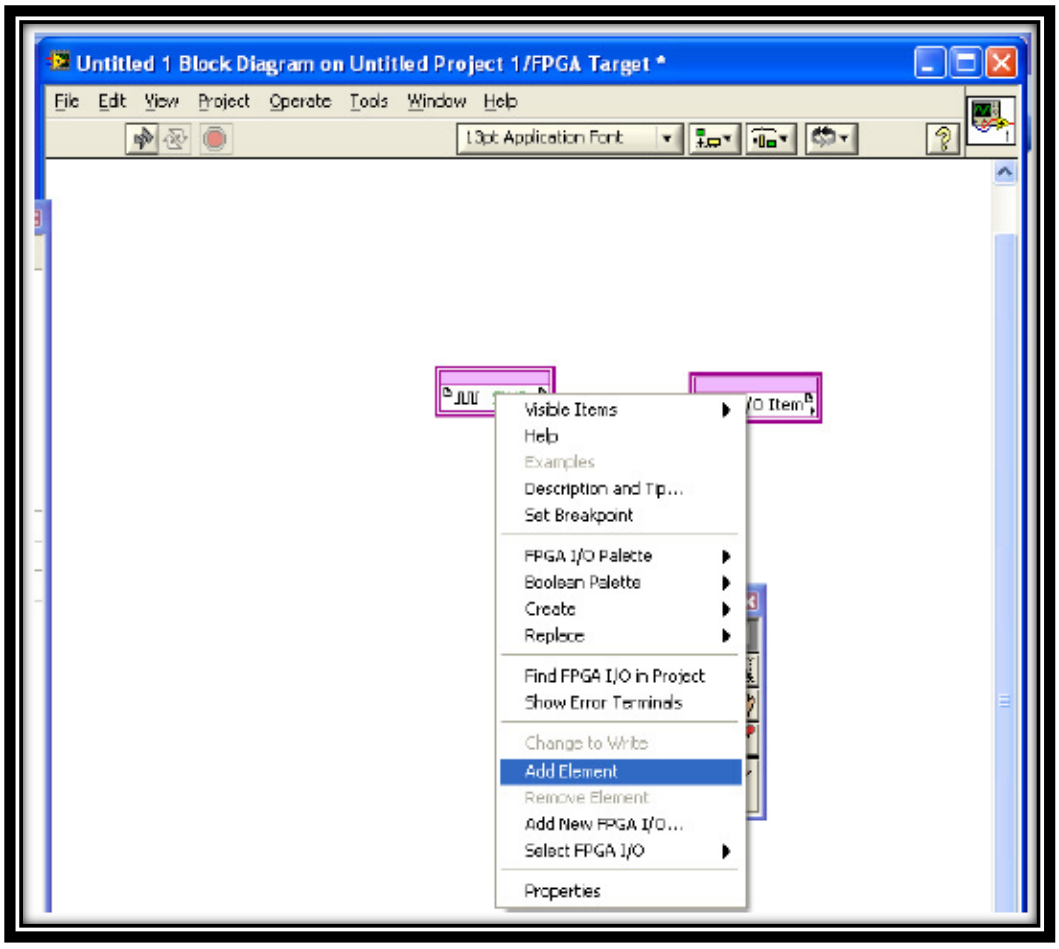
Ahora haga "click derecho del ratón", en uno de los nodos de I / O. Seleccione " Select FPGA I/O " y luego " Slide Switches” " y luego "SW0".



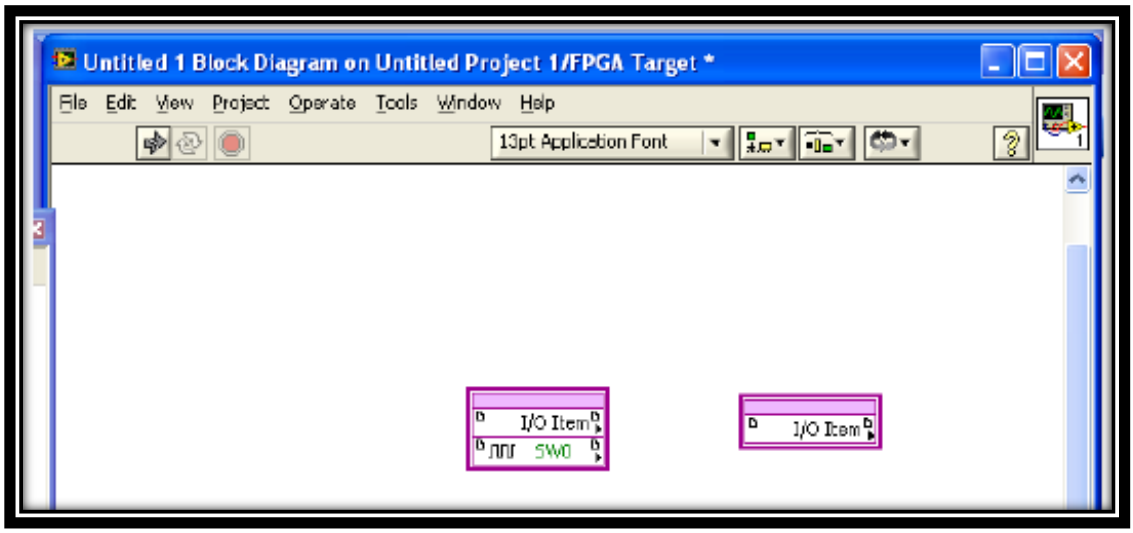
Usted debe ver que el nodo de I / O se llena con una etiqueta verde "SW0". El color verde está de pie para una variable booleana. Esto es correcto desde un interruptor deslizante sólo puede tener el valor de verdadero o falso.



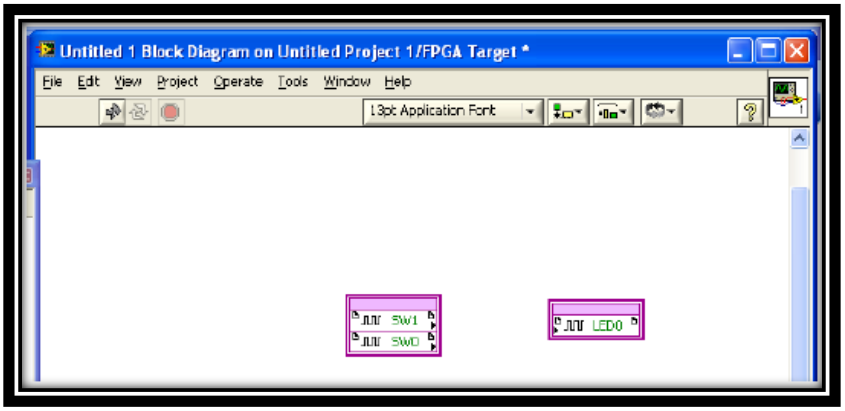
Este proyecto va a utilizar 2 interruptores deslizantes así que tenemos que añadir otro FPGA I / O. Para ello haga un clic derecho en la FPGA de I/ O que acaba de rellenar de SW0. Seleccione " Add Element ".



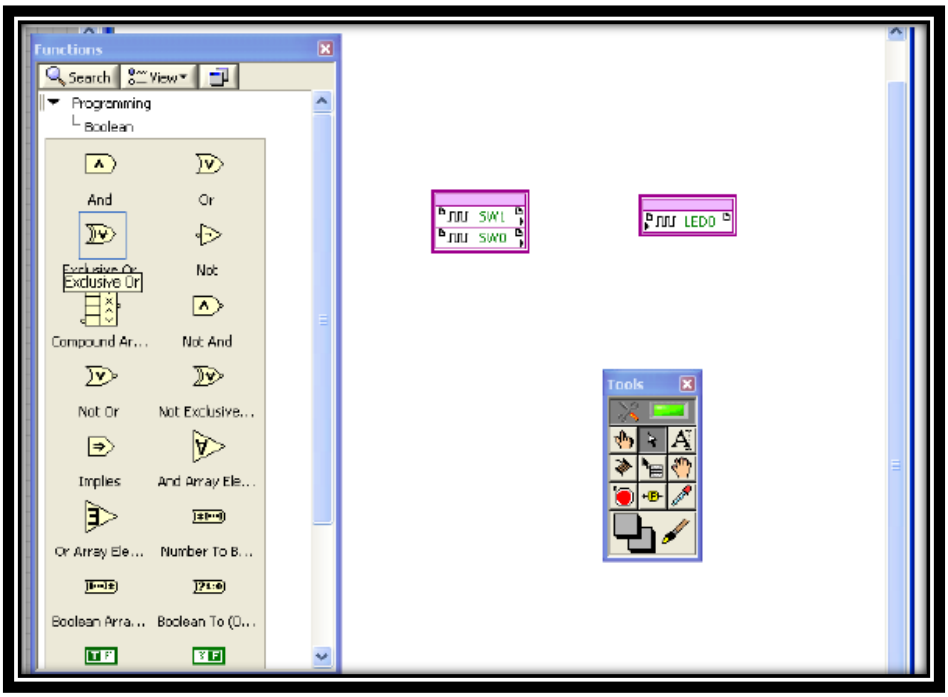
La siguiente es la pantalla que usted debe tener ahora:



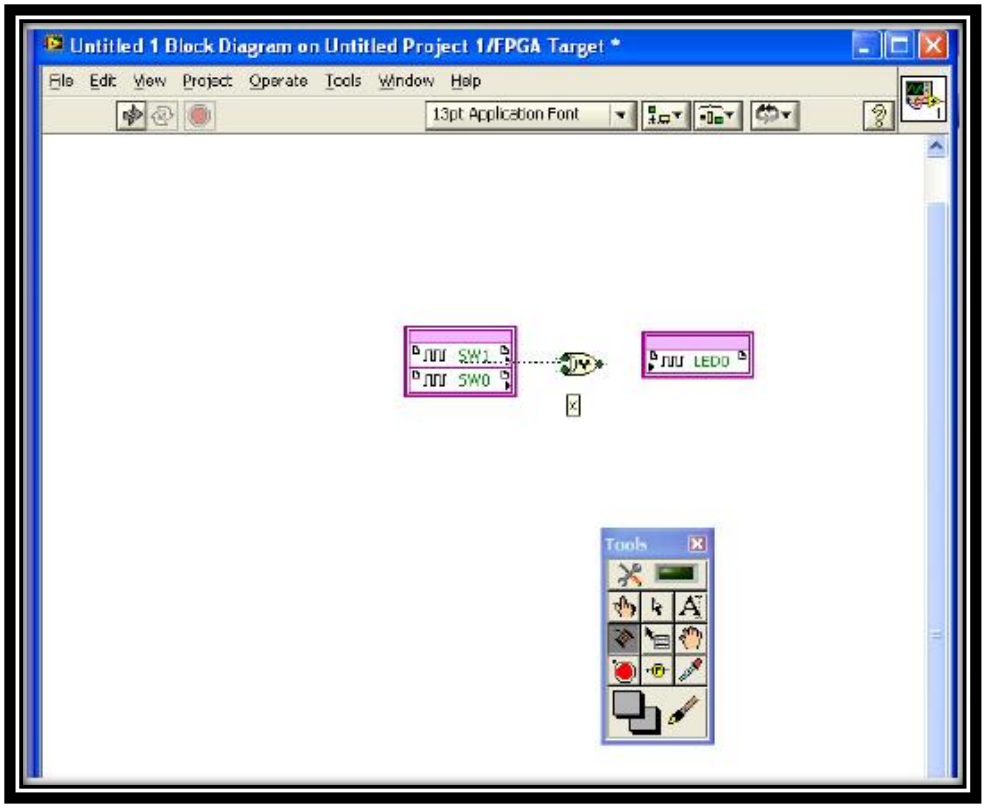
Trate ahora de llenar el ítem I / O usted mismo con SW1. Pruebe también para llenar el nodo de I/ O que está vacía con LED0. Esto se hace con el ratón botón derecho del ratón y luego seleccionando " Select FPGA I/O" ". En la opción de " Select FPGA I/O" " ver sólo los recursos que agregó en la ventana " Project Explorer " para el proyecto. Se creará la siguiente pantalla:



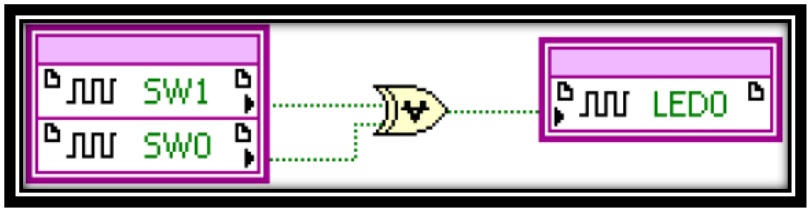
Ahora vamos a poner en práctica como un ejemplo de una función booleana exclusive-OR en la FPGA. Desde la " Functions Palette ", seleccione" Programming " "boolean" y buscar la "función O-exclusiva". Coloque éste en el diagrama de bloques.



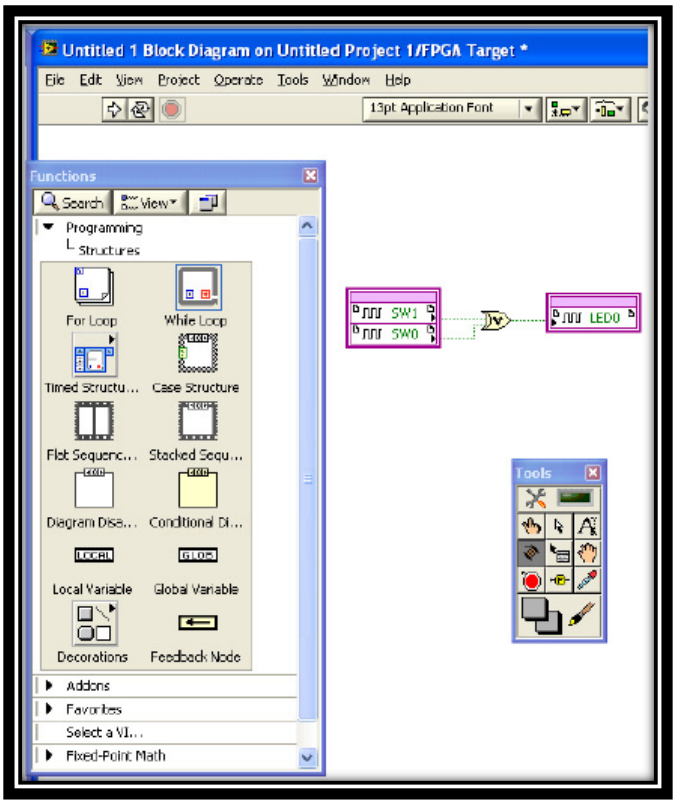
Conecte el "SW1" y "SW0" FPGA de I/ O para las entradas Exclusive-OR. Conecte la salida de la función de "LED0".

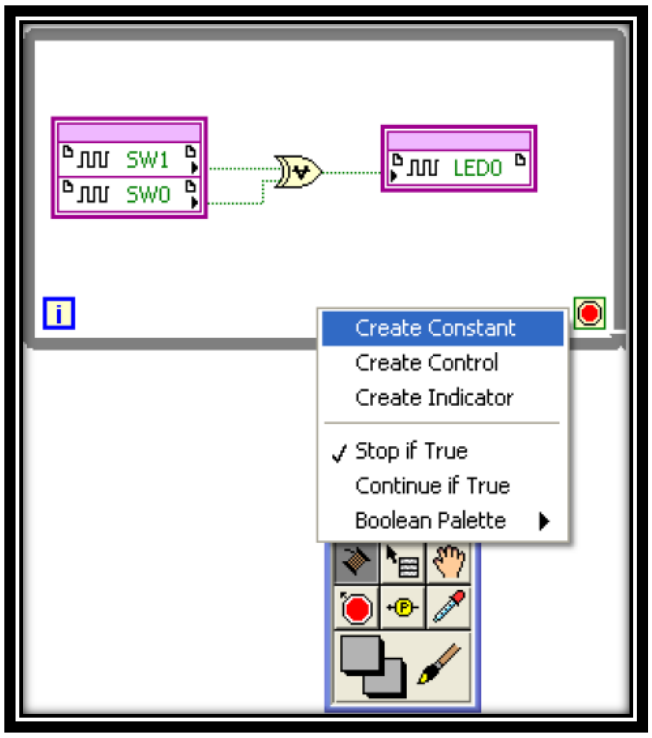


El código debe tener este aspecto:

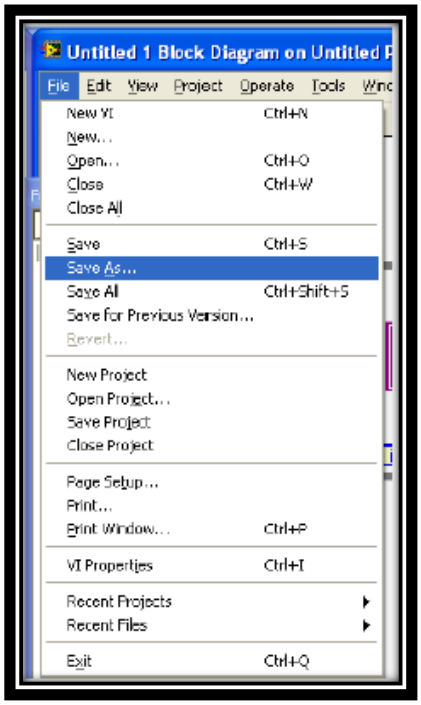


Cuando se implementa la función como se presenta por encima de la función se ejecutará sólo una vez. Nos gustaría poner en práctica lo que funciona continuamente. Para ello se coloca un "While Loop" a su alrededor y que conecta una booleano "Falso constante" a la "condición de parada" de este bucle. Esto normalmente no lo hacen en una PC, porque esto hará que su PC en un bucle sin fin. El "While Loop" te encuentras en la " Functions Palette".

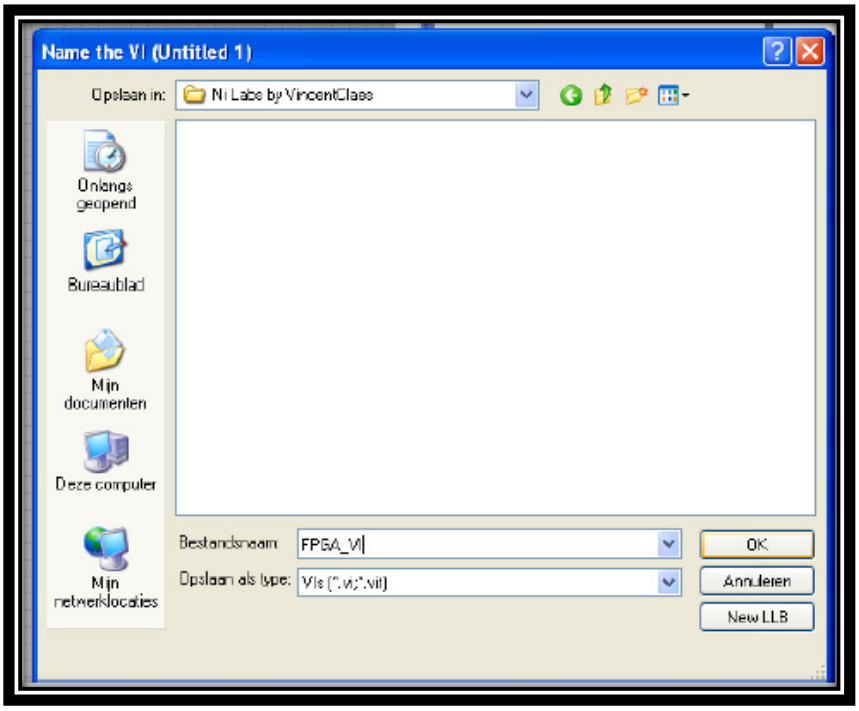

Untitled 1 Block Diagram on Untitled Project 1/FPGA Target *
File Edit View Project Operate Tools Window Help
13pt Application Font
Functions
Search View
Programming
Structures
For Loop
While Loop
Timed Structu...
Case Structure
Flat Sequenc...
Stacked Sequ...
Diagram Disa...
Conditional Di...
Local Variable
Global Variable
Decorations
Feedback Node
Addons
Favorites
Select a VI...
Fixed-Point Math
SW1
SW0
LED0
Tools


SW1
SW0
LED0
Create Constant
Create Control
Create Indicator
Stop if True
Continue if True
Boolean Palette

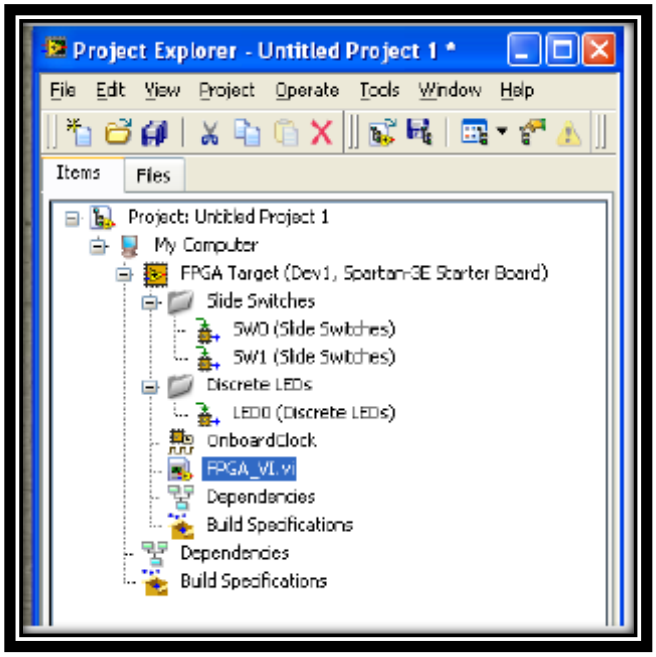
Ahora es el momento de guardar la vi que ha creado para la FPGA. Esto se hace mediante la opción "New" en "Save as", ya sea en el " Front Panel" o el " Front Panel".



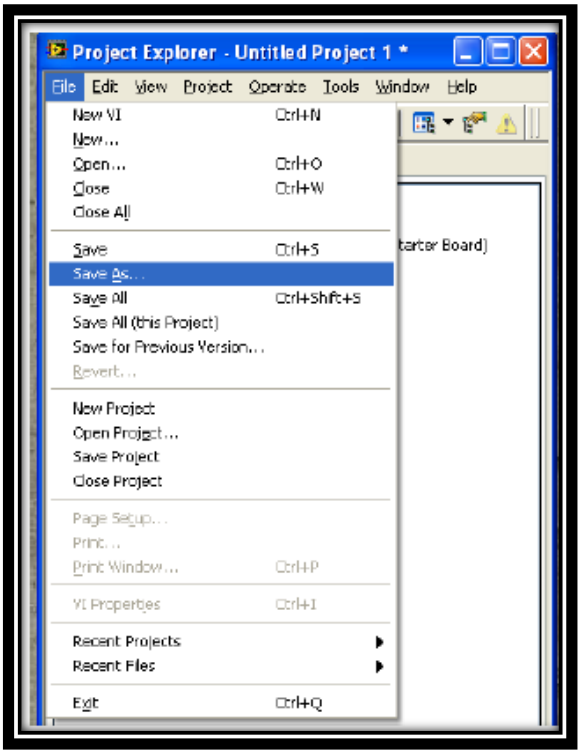
El nombre que suelen utilizar es FPGA_VI para la vi corriendo en la FPGA.



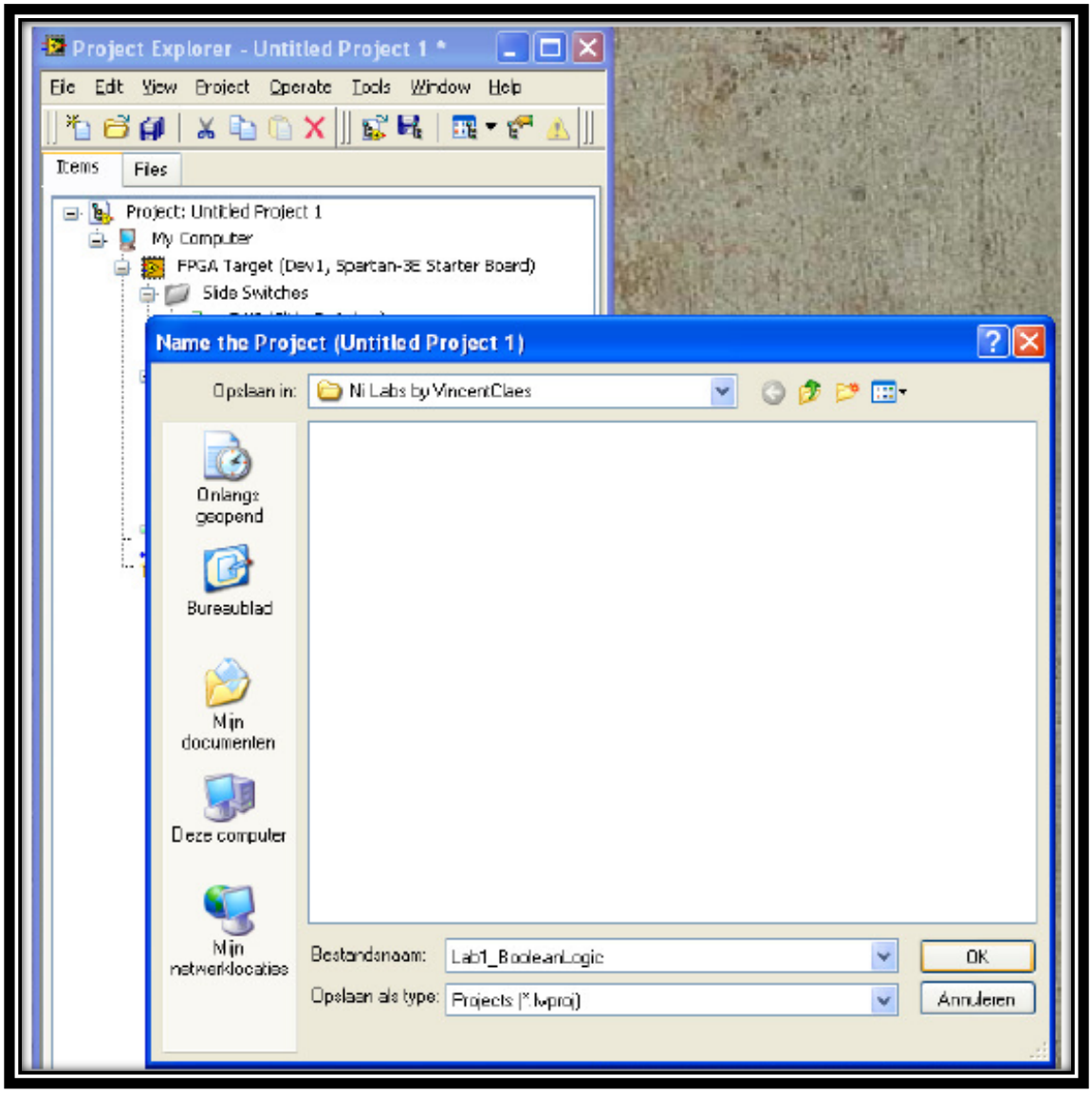
En " Project Explorer” " debería ver el siguiente:



También es una buena idea guardar el archivo de proyecto para esta selección en el “Project Explorer” “New” entonces “Save As”.

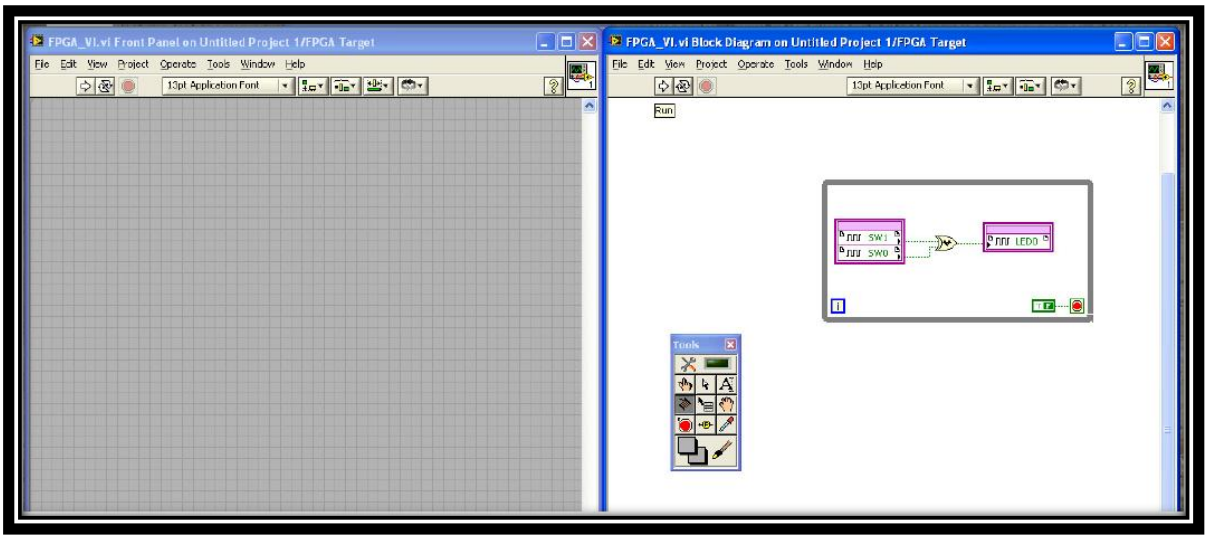


Utilicé el nombre Lab1_BooleanLogic:

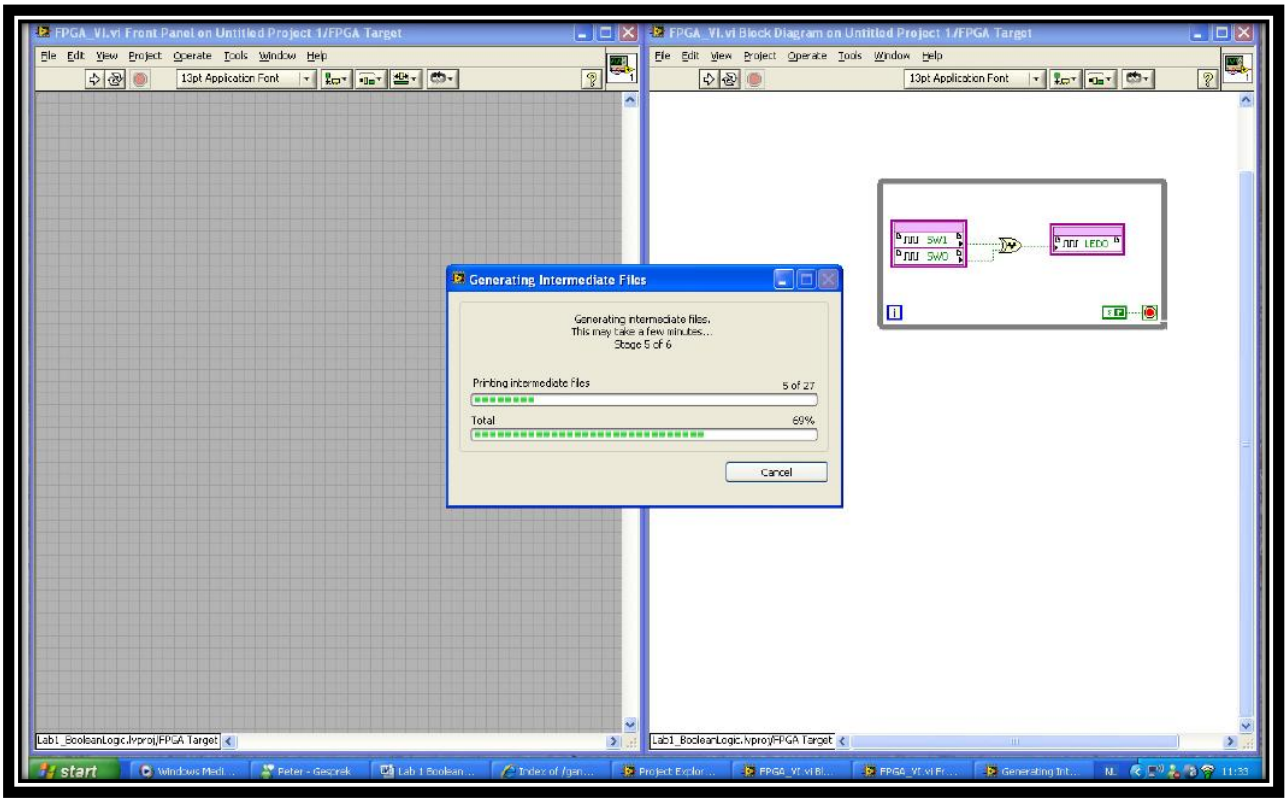


Paso 6: Ejecutar el VI de FPGA

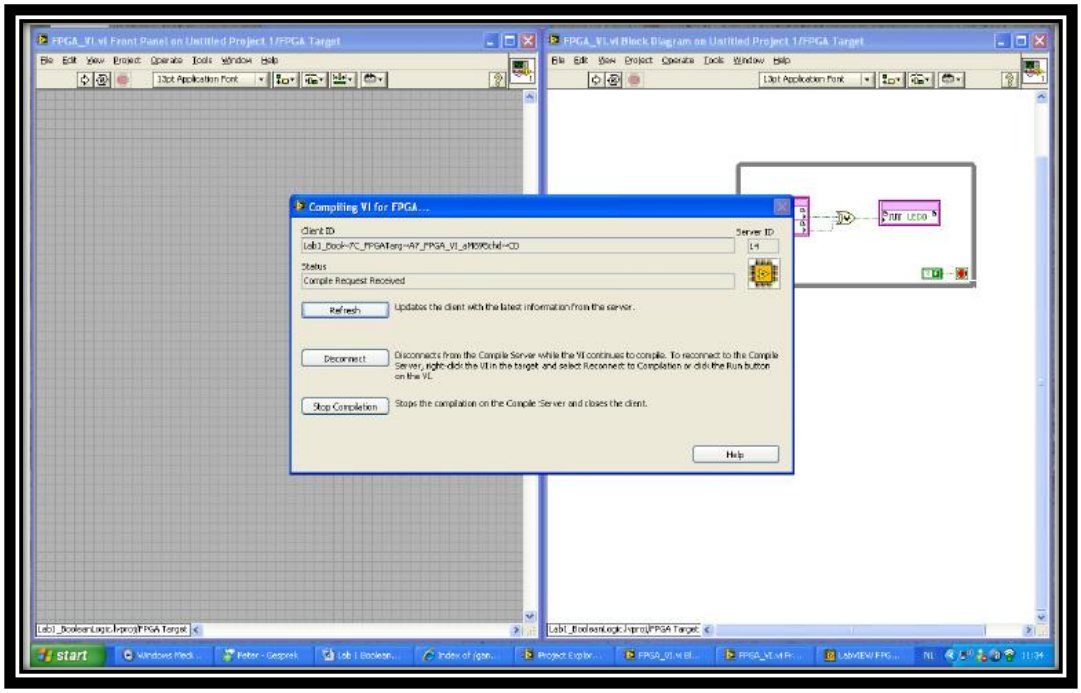
Este paso es donde creamos la VI. Lo hemos diseñado para ejecutarse en un blanco. Para comenzar la ejecución de esta VI que tenemos que pulsar el botón " Run " flecha ya sea en el " Block Diagram” o en la " Front Panel”.

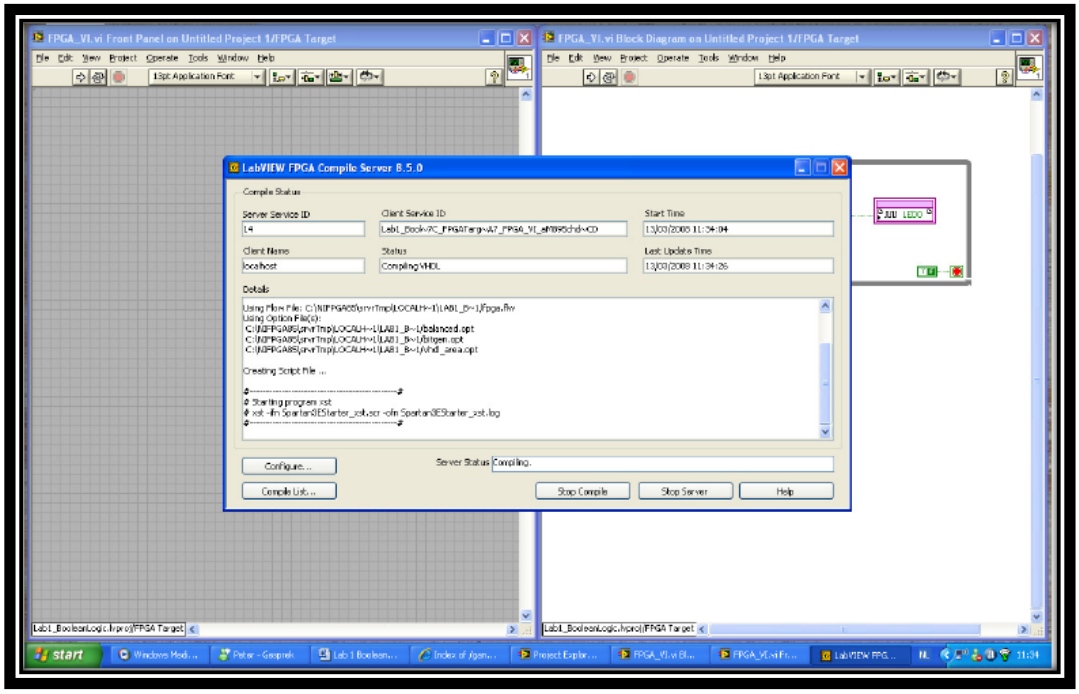


El primer paso LabView hace es "Generación de archivos intermedios". Estos archivos se envían a las herramientas de Xilinx Synthesis. Pero esto no es importante para nosotros como desarrolladores de aplicaciones.

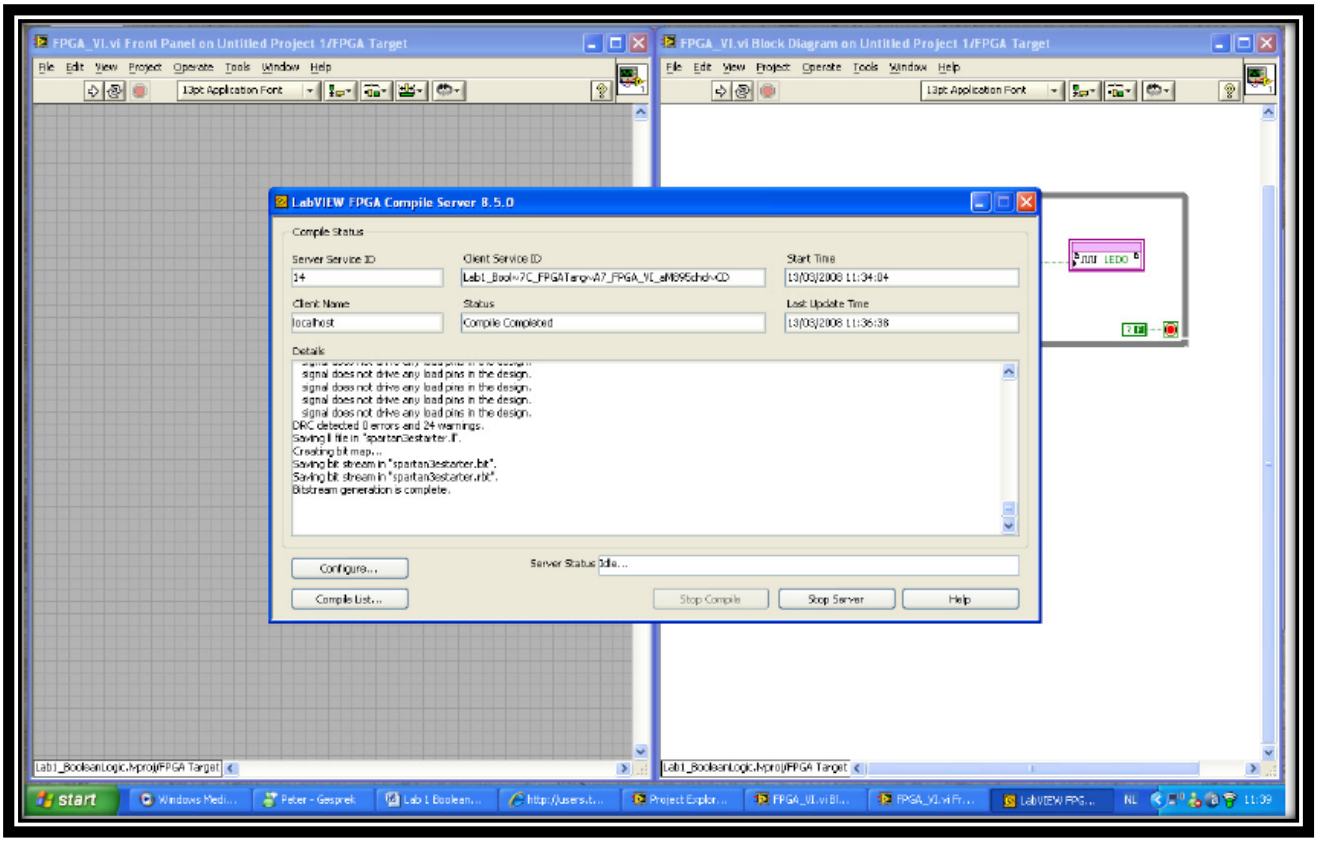


En este paso se ve que LabView está iniciando el "Compilar Server". Este "Compiler Server" también se puede ejecutar en otro "more powerful" de la máquina que está en tu red (para esto, por favor consulte la información en el sitio web de NI.)

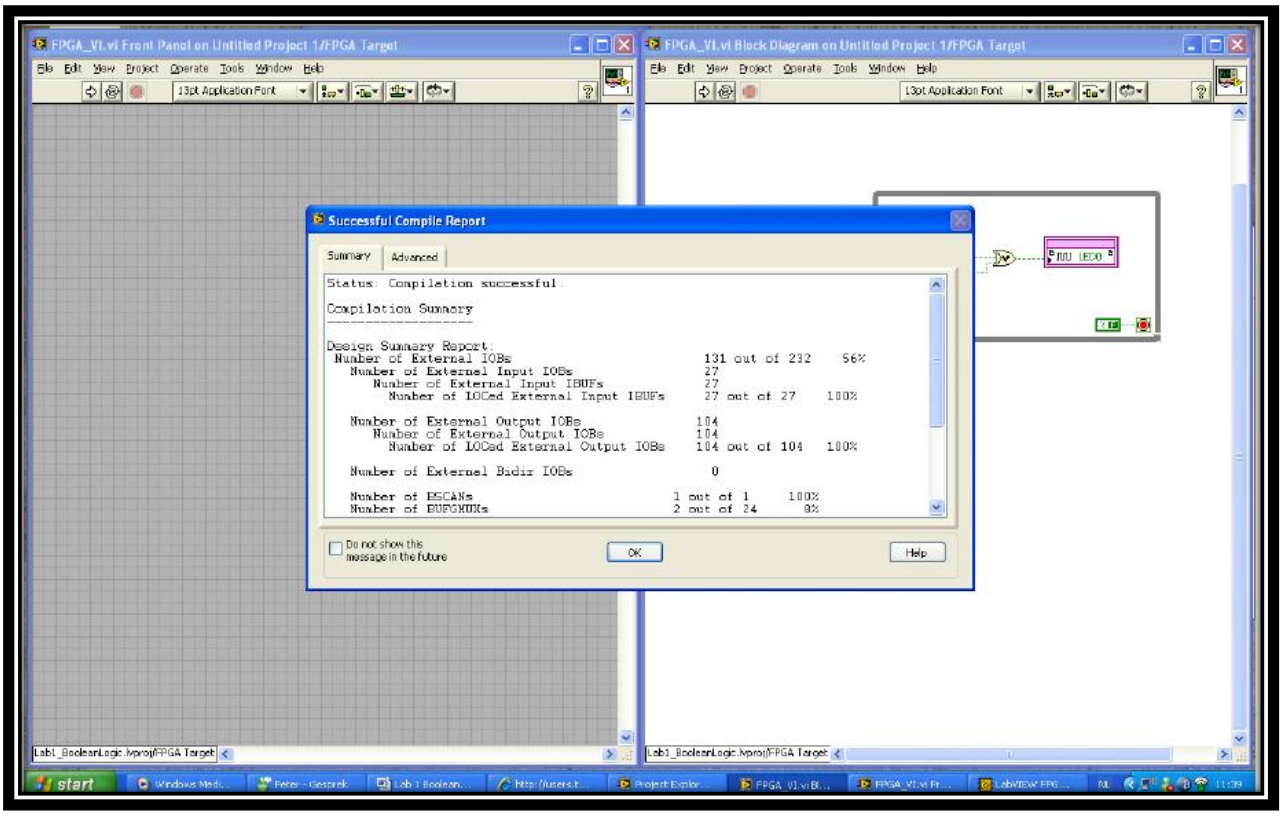




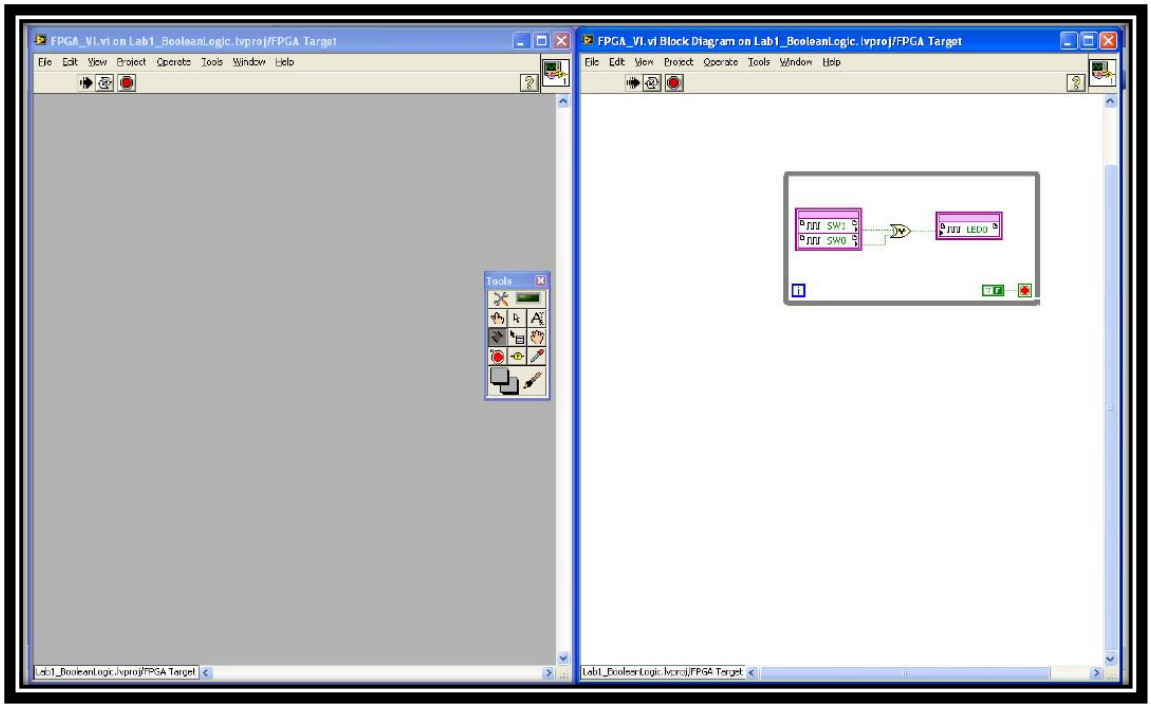
Cuando la "generación de flujo de bits es completa" aparece y el estado del servidor se establece en "Idle" las herramientas de síntesis de Xilinx han hecho su trabajo.



Usted recibe un "Informe exitosa compilación", donde se pueden ver los detalles de implementación de su código. Tienes que pulsar "Ok".



Después de haber pulsado el botón "Ok" de su VI comienza a funcionar en el objetivo FPGA. Está indicado en la pantalla por el negro en "Ejecutar" flecha.

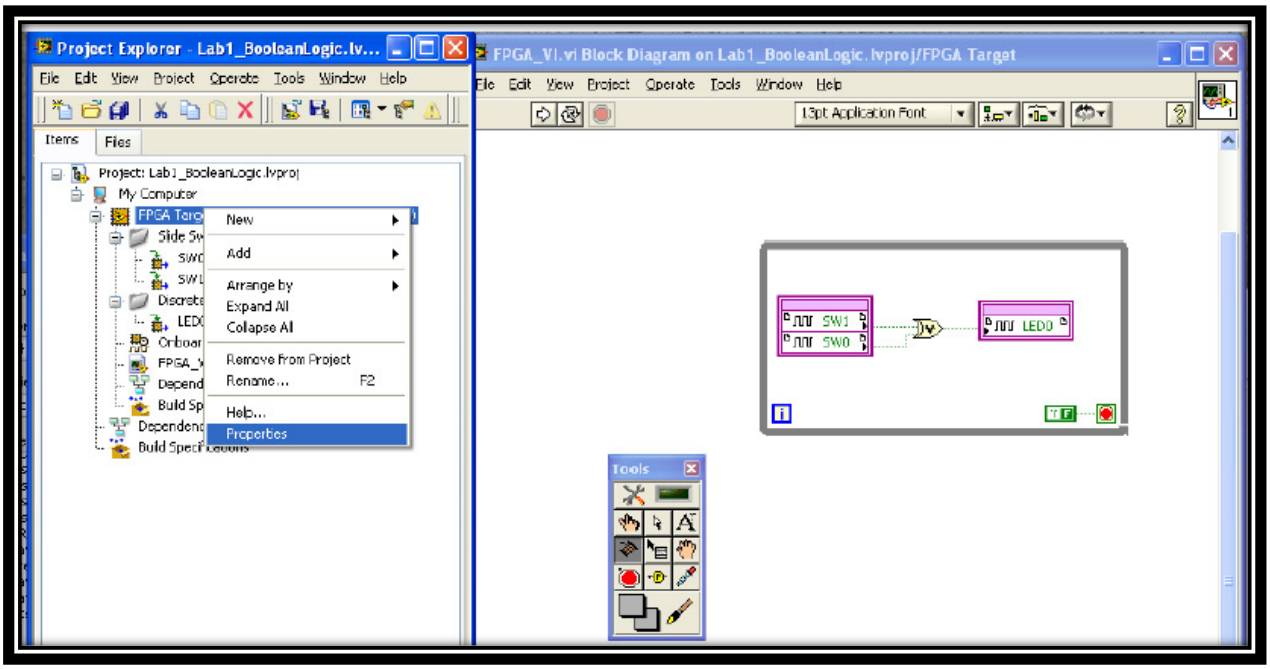


Trate de jugar con los interruptores SW1 SW0 y en el tablón de inicio Spartan3E verás que tienen una función XOR; la LD0 LED se encenderá cuando uno de esos interruptores se enciende.

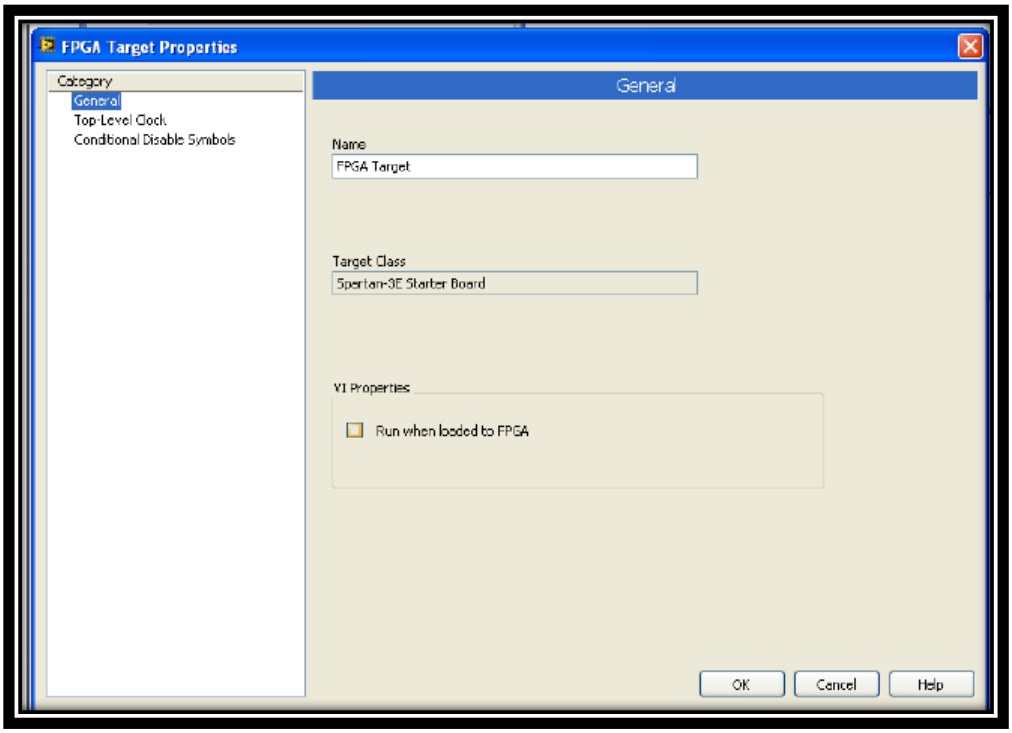
El problema de esta aplicación es que cuando se detiene el VI la función se borra en el tablero de LabView FPGA. Si no desea este efecto puede descargar este VI del FLASH de la tabla de inicio Xilinx Spartan3E (ver paso siguiente).

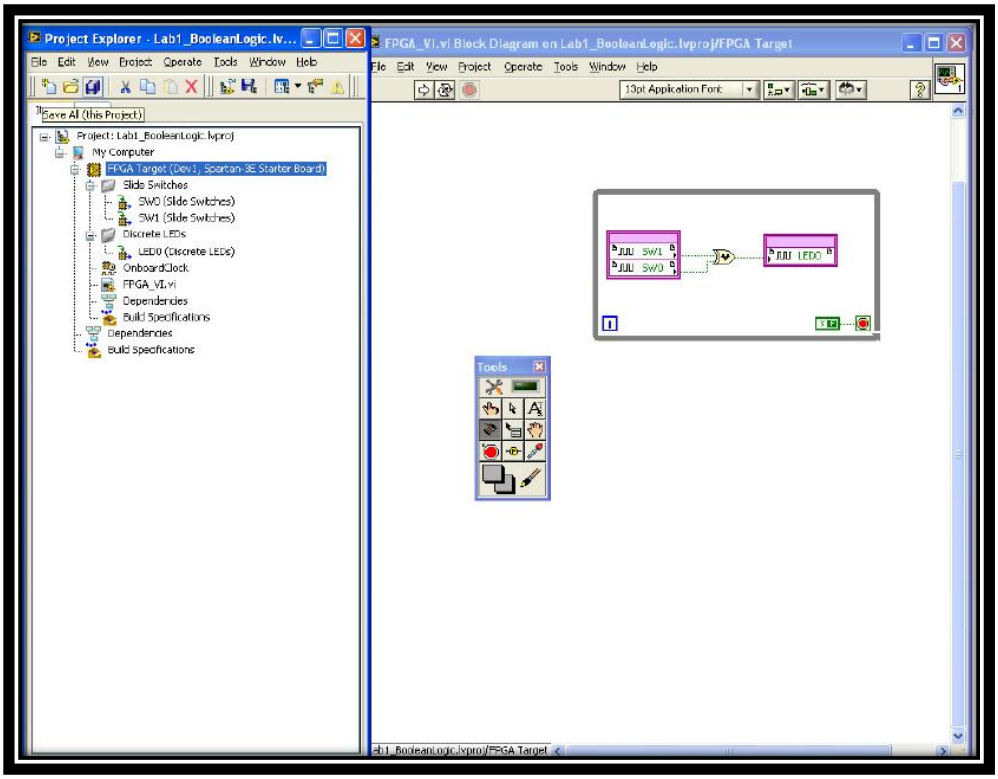
Paso 7: Implementar el LabView VI en Flash

Para la implementación de su VI en el flash en la tarjeta Spartan3E se tiene que hacer algunas cosas. Lo primero es establecer la opción de "Ejecutar cuando se carga a FPGA" en. Para ello tienes que ir a "Explorador de proyectos". A continuación, "derecho del ratón haga clic en" en su objetivo FPGA. Seleccione la opción "Propiedades".

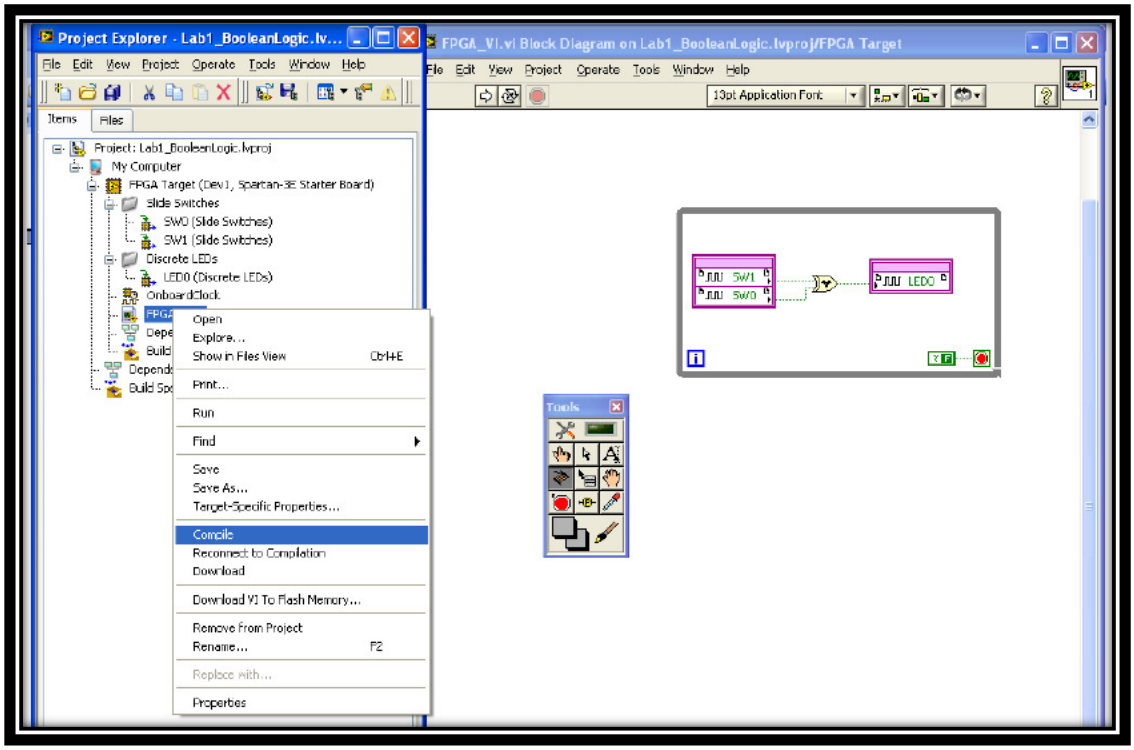


En esta pantalla se puede ver la opción "Ejecutar una vez cargado con FPGA". Asegúrese de seleccionar esta opción. A continuación, pulse el botón "OK".

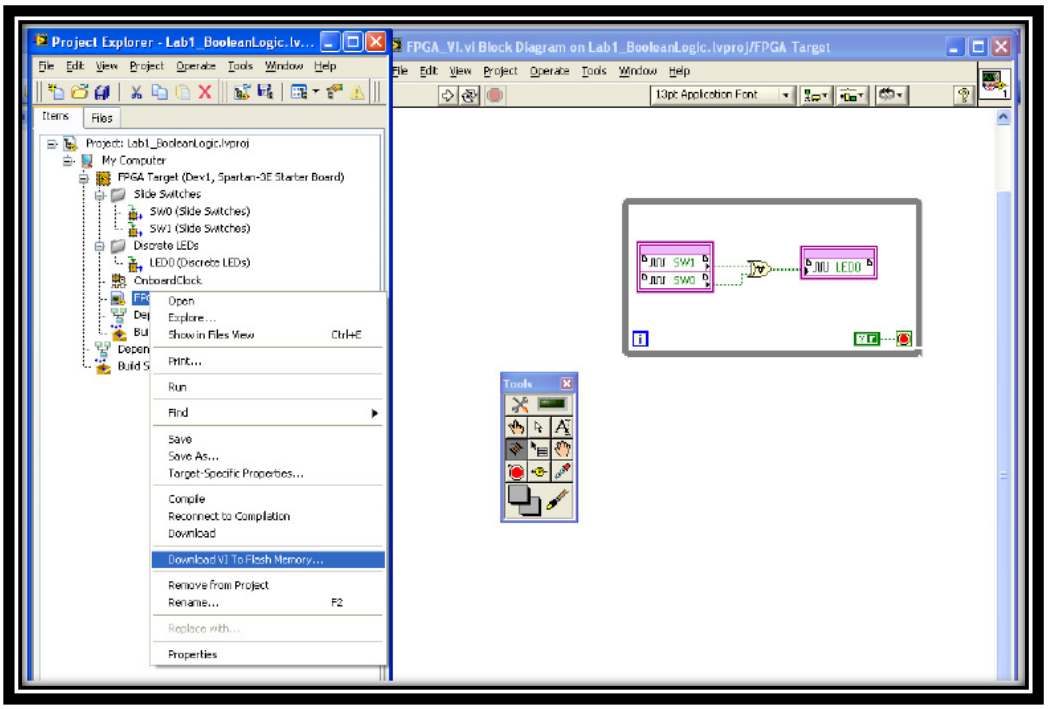




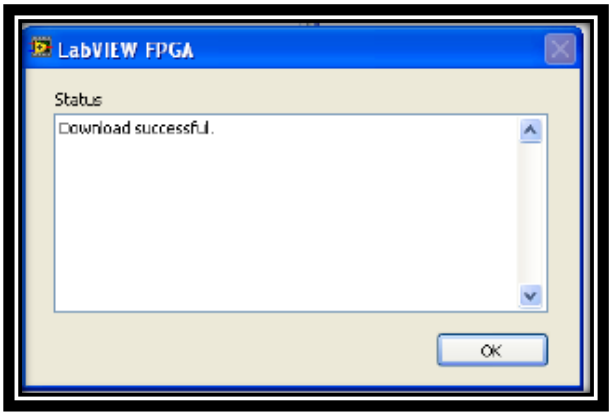
Tenemos que volver a compilar el VI porque hemos hecho un cambio. Para ello, ir a la vista "Explorador de proyectos". Haga clic con el "botón derecho del ratón" en el VI de FPGA que ha creado para este proyecto. Seleccione la opción "Compilar".



Usted verá que hay algunos pasos previos explicados ejecutados. Cuando el "informe de compilación sea exitoso" tiene que pulsar el botón "OK". Ahora vamos a descargarlo en el flash. Para ello tienes que ir a la vista "Explorador de proyectos" y hacer un "botón derecho del ratón" en la FPGA VI creó. A continuación, elegir la opción "Descargar VI de Memoria Flash" opción comenzará a descargar a la Spartan-3E .



Cuando aparezca esta casilla el LabView FPGA VI se descarga en el Flash.ash.



Ahora usted puede sacar el cable USB de la placa Xilinx Spartan3E y pulse el botón PROG en este foro. Verá que la función se implementa en el mismo.

ANEXO 2

MANUAL DE LABVIEW + ARDUINO

6

Labview + Arduino

Utilización de Labview para la Visualización y Control de la Plataforma Open Hardware Arduino

+

Arduino

Ver. 1.0

José Manuel Ruiz Gutiérrez

Serie: Herramientas Gráficas para la programación de Arduino

4. Instalación del Software y el Hardware

A continuación describimos los pasos que se recomiendan para la puesta en marcha de la herramienta **LIFA** (LabVIEW para Arduino):

La configuración de la Interfaz de LabVIEW para Arduino es un proceso de seis pasos que usted sólo tendrá que completar una sola vez. Por favor, siga las siguientes instrucciones para comenzar a crear aplicaciones con la interfaz de LabVIEW para Arduino.

(Para una breve descripción de la interfaz de LabVIEW para Arduino ver post Michaels aquí).

1. Instalar LabVIEW

   Si ha adquirido el paquete de LabVIEW y del Sparkfun.com Arduino puede instalar LabVIEW desde el DVD incluido.

   Si usted no posee una copia de LabVIEW, usted puede descargar e instalar la versión de evaluación de 30 días aquí.

2. Instale los controladores VISA NI-.

   - Windows Download.
   - Linux Download.
   - Mac Download.

3. Instale JKI VI Package Manager (VIPM) Community Edition (gratuito).

   Todos los sistemas operativos. All Operating Systems.

4. Instalación de la Interfaz de LabVIEW para Arduino como se describe en KB 5L38JQYG KB 5L38JQYG

5. Conectar la placa Arduino a su PC como se describe en KB 5INA7UYG KB 5INA7UYG

6. Carga de la interfaz de LabVIEW para firmware Arduino en su Arduino como se describe en KB 5LPAQIYG

7. El firmware se puede encontrar en <LabVIEW> \ vi.lib Interface \ LabVIEW para Arduino \ Firmware \ LVIFA_Base. Utilizar el IDE de Arduino para implementar este firmware de la placa Arduino.)

   Ahora está listo para usar la interfaz de LabVIEW para Arduino.

5. Instalación del Firmware de comunicación entre LabVIEW Interface y Arduino Uno?

Para poder comunicar Labview con Arduino, previamente, debemos instalar en la tarjeta el firmware correspondiente.

Partimos del supuesto de que ya tenemos instalado en nuestro PC el entorno IDE Arduino.

El fichero que debemos cargar en el IDE de Arduino para luego descargar en la tarjeta se encuentra en la carpeta en dinde tengasmos instaldo Labview

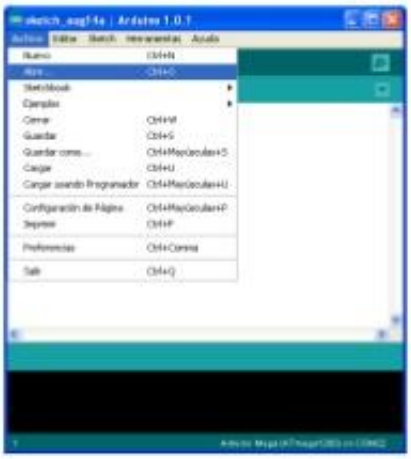
...\National Instruments\LabVIEW 20XX\vi.lib\LabVIEW Interface for Arduino\Firmware\LVIFA_Base

Ejecutamos el IDE Arduino y cargamos el fichero.

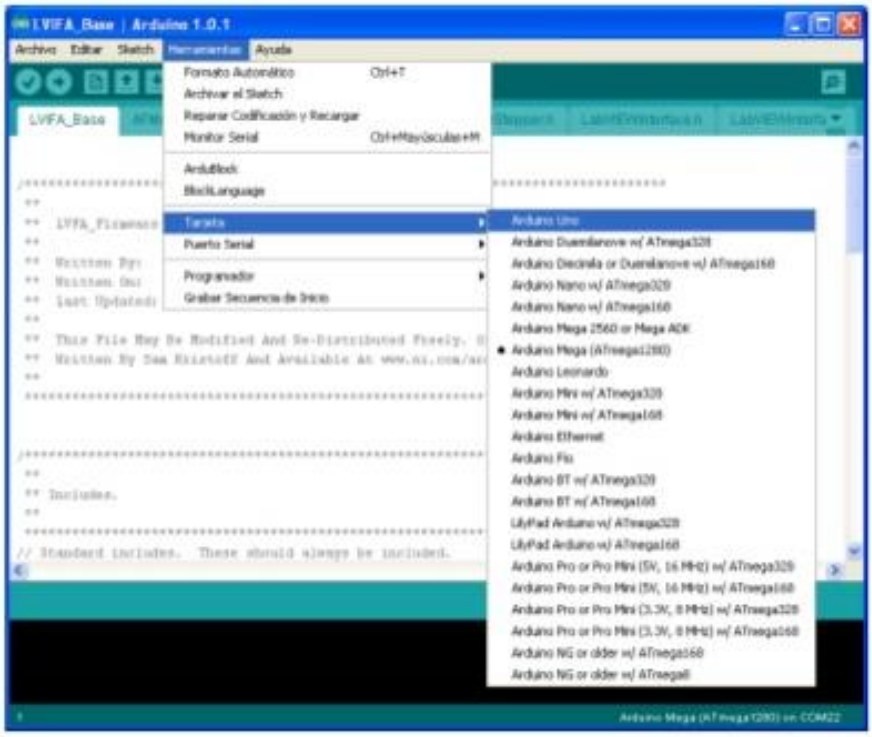
Pasos a seguir:

- Abrir el IDE Arduino . Pulsando sobre arduino.exe

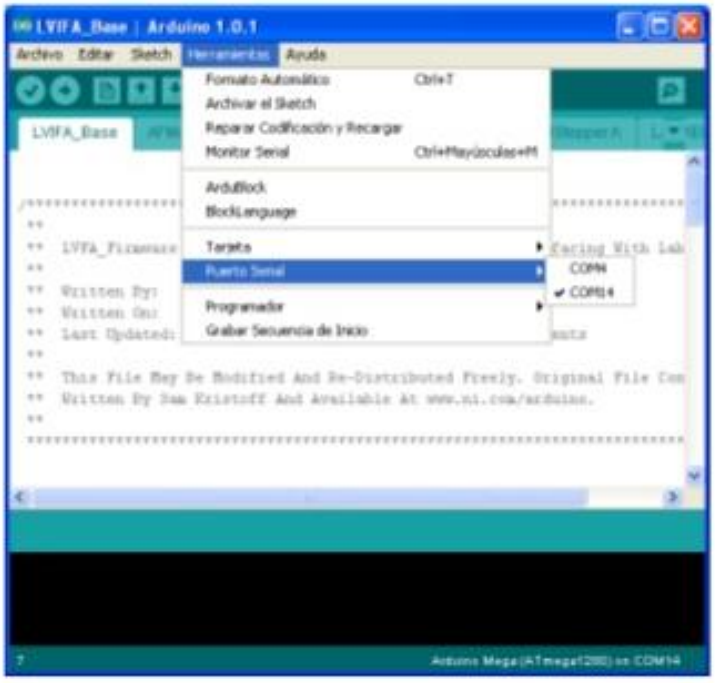
  Con la opción **Fichero->Abrir** Buscamos el fichero LVIFA_Base.pde



- Seguidamente una vez cargado el fichero en el IDE Arduino seleccionamos la tarjeta con la que trabaremos.



- Seguidamente seleccionamos el puerto con el que realizaremos la descarga del firmaware sobre la tarjeta Arduino.



- Una vez realizadas estas operaciones basta con que pulsemos el botón de carga de sketch del IDE para que el fichero se transfiera a la tarjeta y, una vez transferido, ya hemos dejado Arduino listo para comunicarse con LabVIEW

ANEXO 3

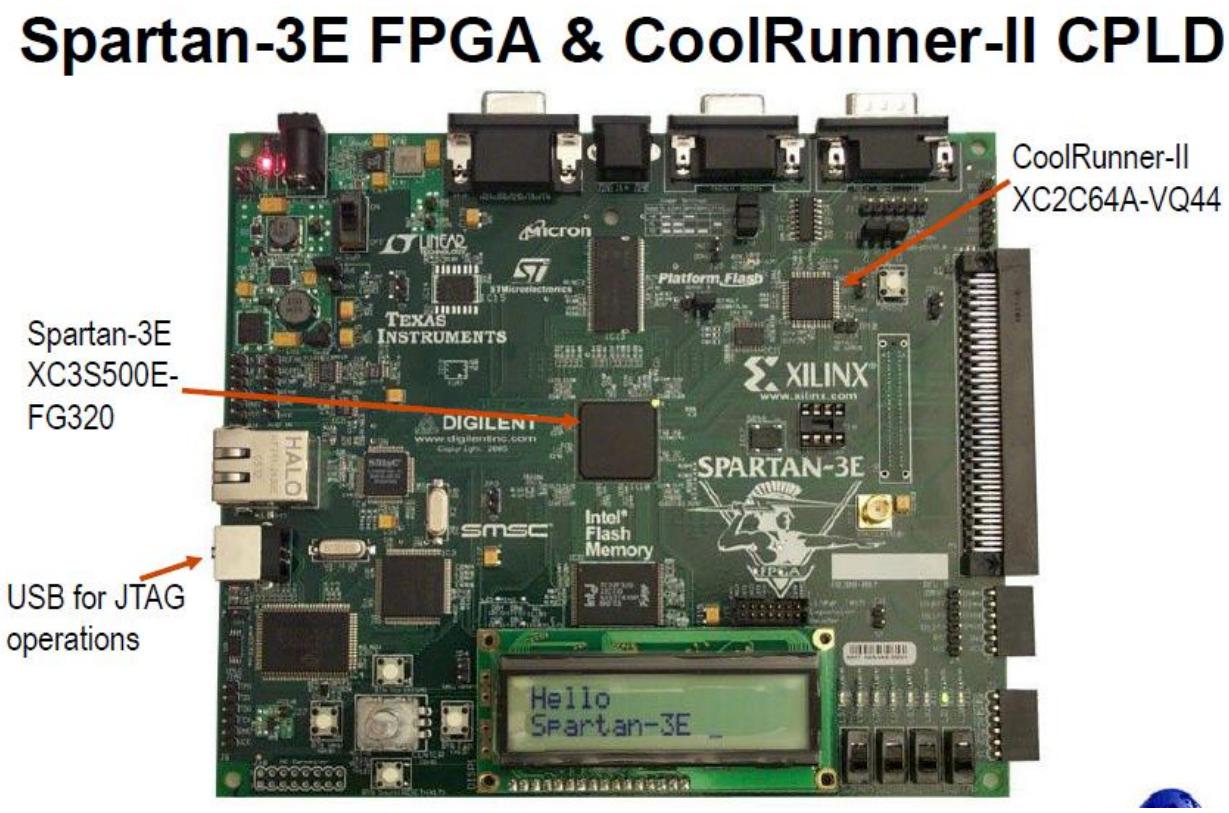
TARJETA XILINX SPARTAN-3E

CARACTERÍSTICAS

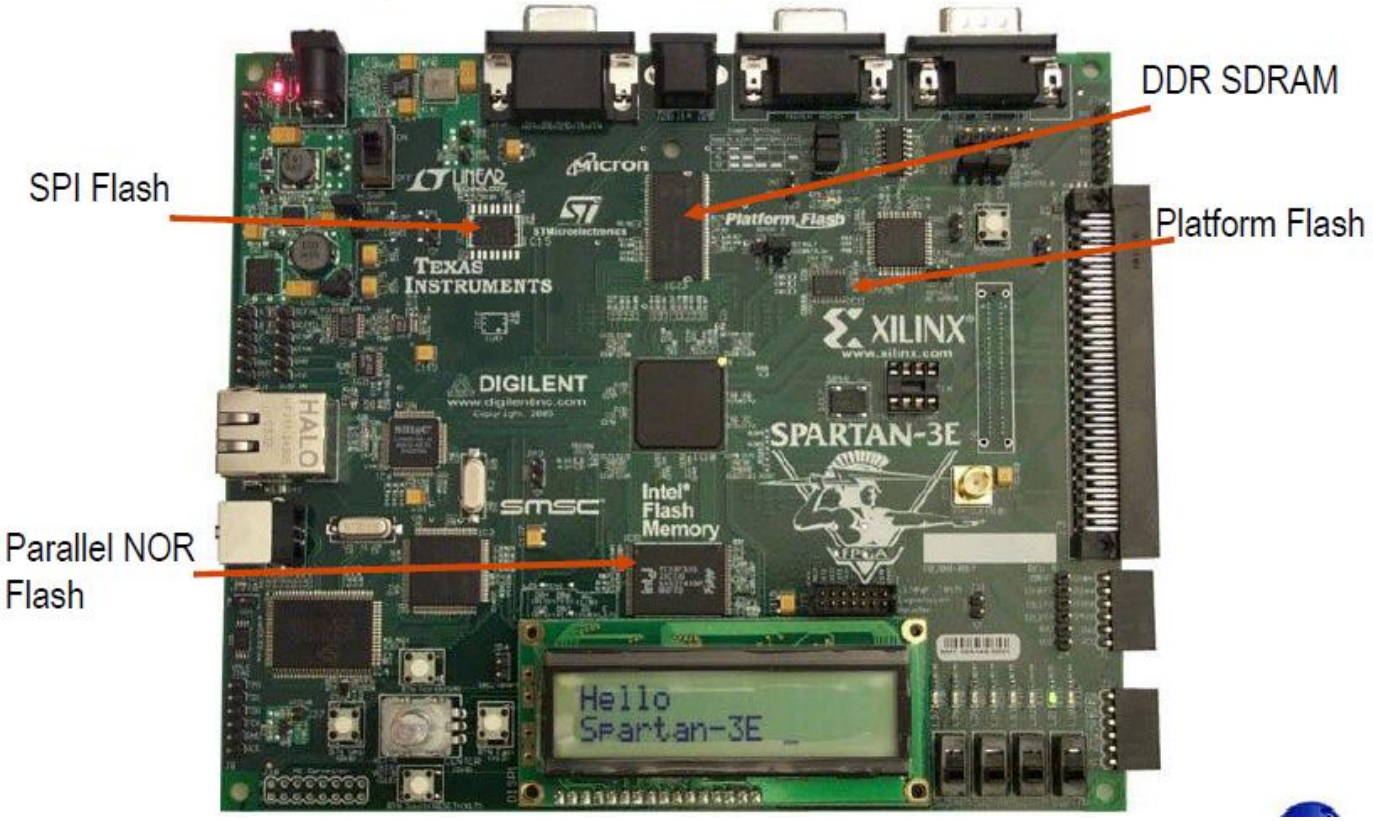
Ic	Xilinx Spartan FPGA 3e (puertas 500k) Dispositivos analógicos adm3232earnz rs-232 driver de línea / receptor
Conectores	Conector FX2 hirose 100 pines Tres conectores PMOD 6 pines VGA DB15HD Ps / 2 para teclado Dos conectores DB9 RS-232 Ethernet RJ-45 Cabecera de 16 pines para módulos LCD opcional Conector SMA para entrada de reloj de alta velocidad
Programación	Programación JTAG a través del puerto USB 2 de bordo; jtag y spi programación flash con cable jtag paralelo o USB; numerosas opciones de configuración adicionales.

XC3S500E FPGA Xilinx. Xilinx xcf04 plataforma Flash para almacenar configuraciones FPGA. ddr sdram micras 64mb StrataFlash Numonyx 16mb Micro electrónica 2mb serie st flash. Fuentes de suministros tecnologías lineales. Texas instruments tps75003 triple suministro IC de gestión de energía.		La tarjeta Spartan 3e proporciona una plataforma de desarrollo autónomo de gran alcance y altamente avanzado para diseños dirigidos a la FPGA de Xilinx Spartan 3e. Cuenta con un puerto de 500k x FPGA Spartan 3e con un procesador RISC de 32 bits y las interfaces de DDR. La placa también cuenta con un flash de la plataforma Xilinx, USB y las interfaces de programación paralela jtag con numerosas opciones de configuración de la FPGA a través de la StrataFlash intel a bordo y la microelectrónica serie st flash. El tablero es totalmente compatible con todas las versiones de las herramientas de Xilinx ise incluyendo la webpack libre. Los buques con una fuente de alimentación y el cable USB para la programación así que los diseños pueden ser implementados de inmediato, sin costes

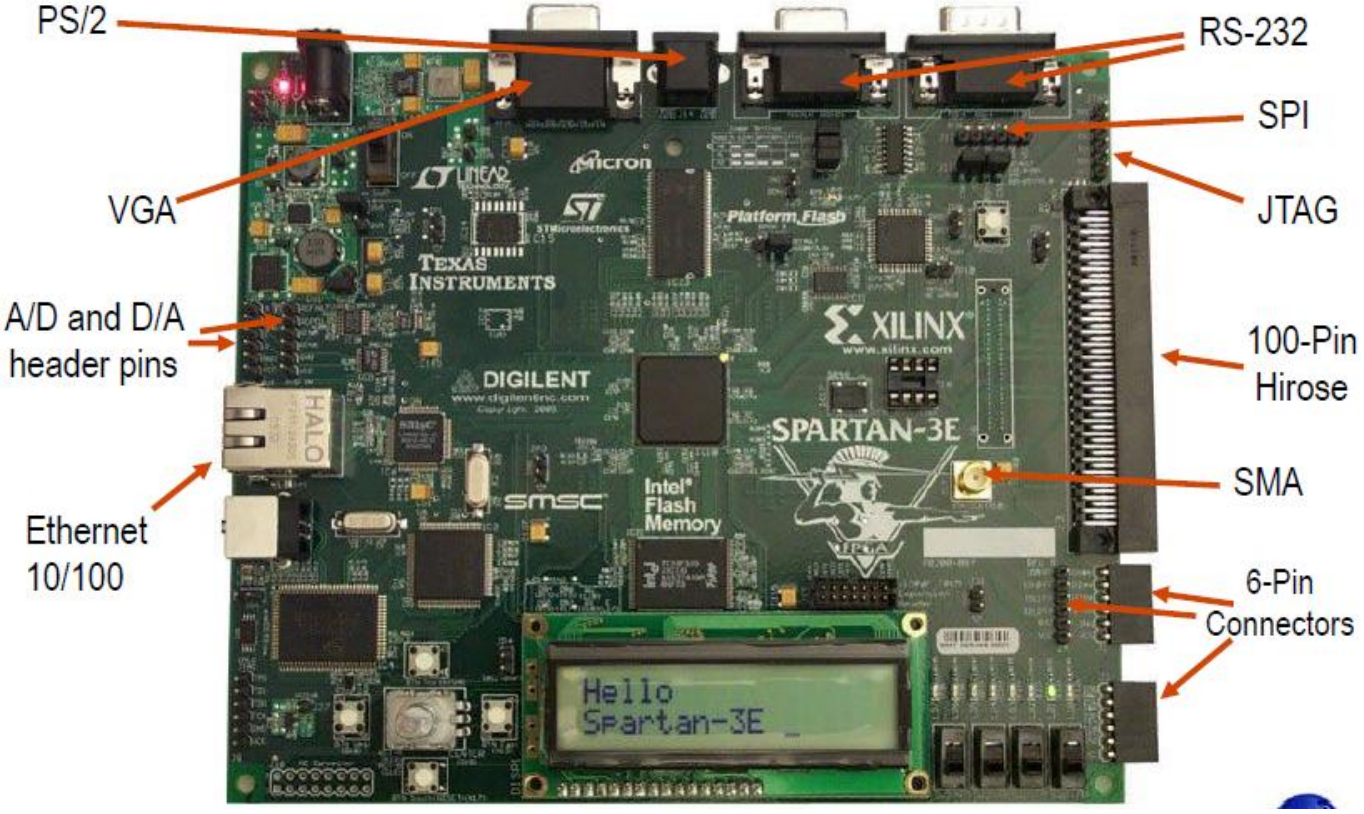
Phy Smsc LAN83C185 ethernet.		ocultos. La tarjeta Spartan 3E también es compatible con el kit de desarrollo MicroBlaze incrustado (EDK) y picoblaze de Xilinx. Por los diseños de referencia y documentación, por favor visite la página de soporte Spartan 3e de Xilinx. Nota: con el fin de programar y comunicarse con la tarjeta Spartan 3E, debe descargar e instalar el software Xilinx ISE webpack libre.



Easy Memory Interfaces



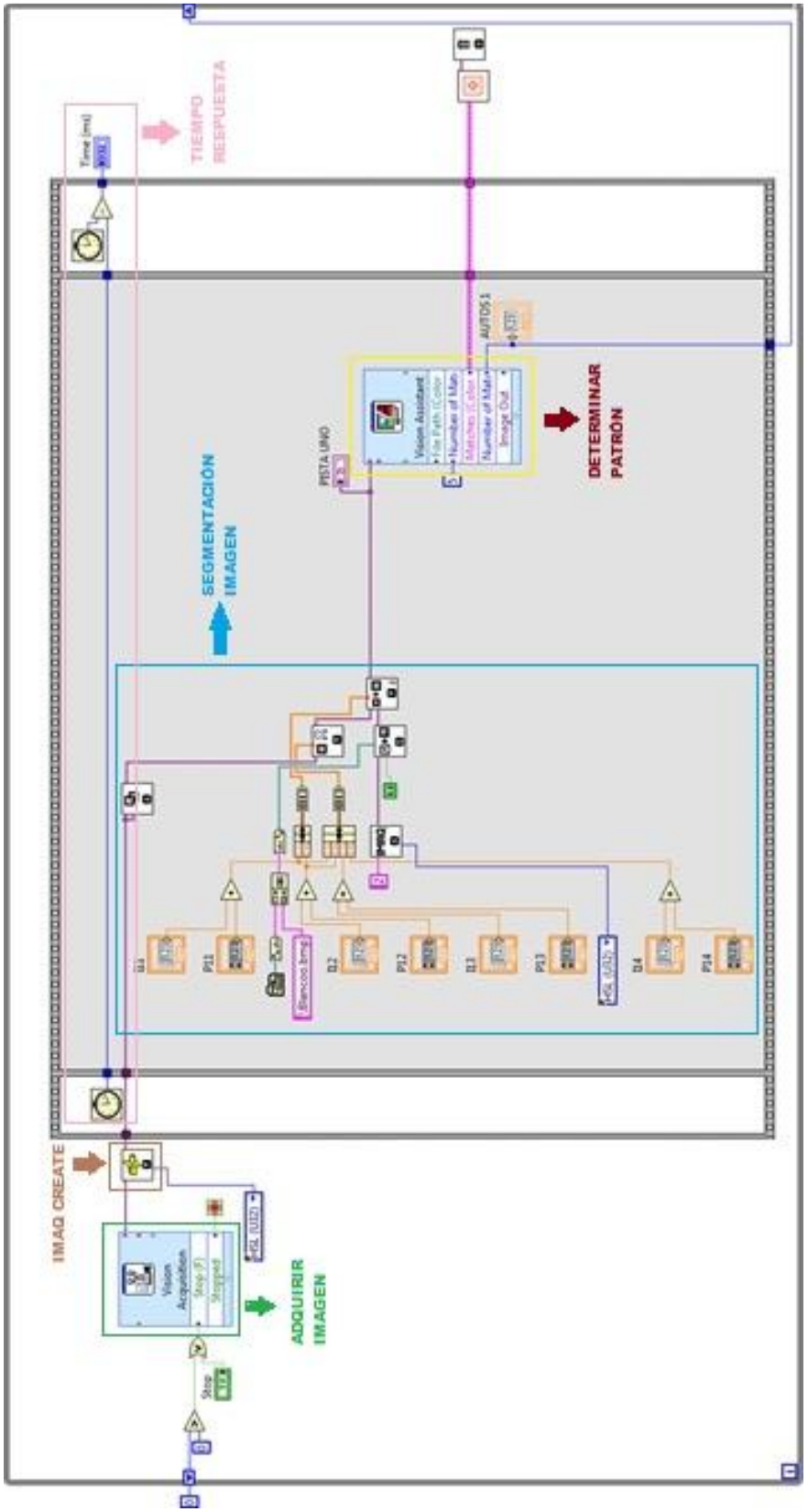
Spartan-3E Excels at Bridging



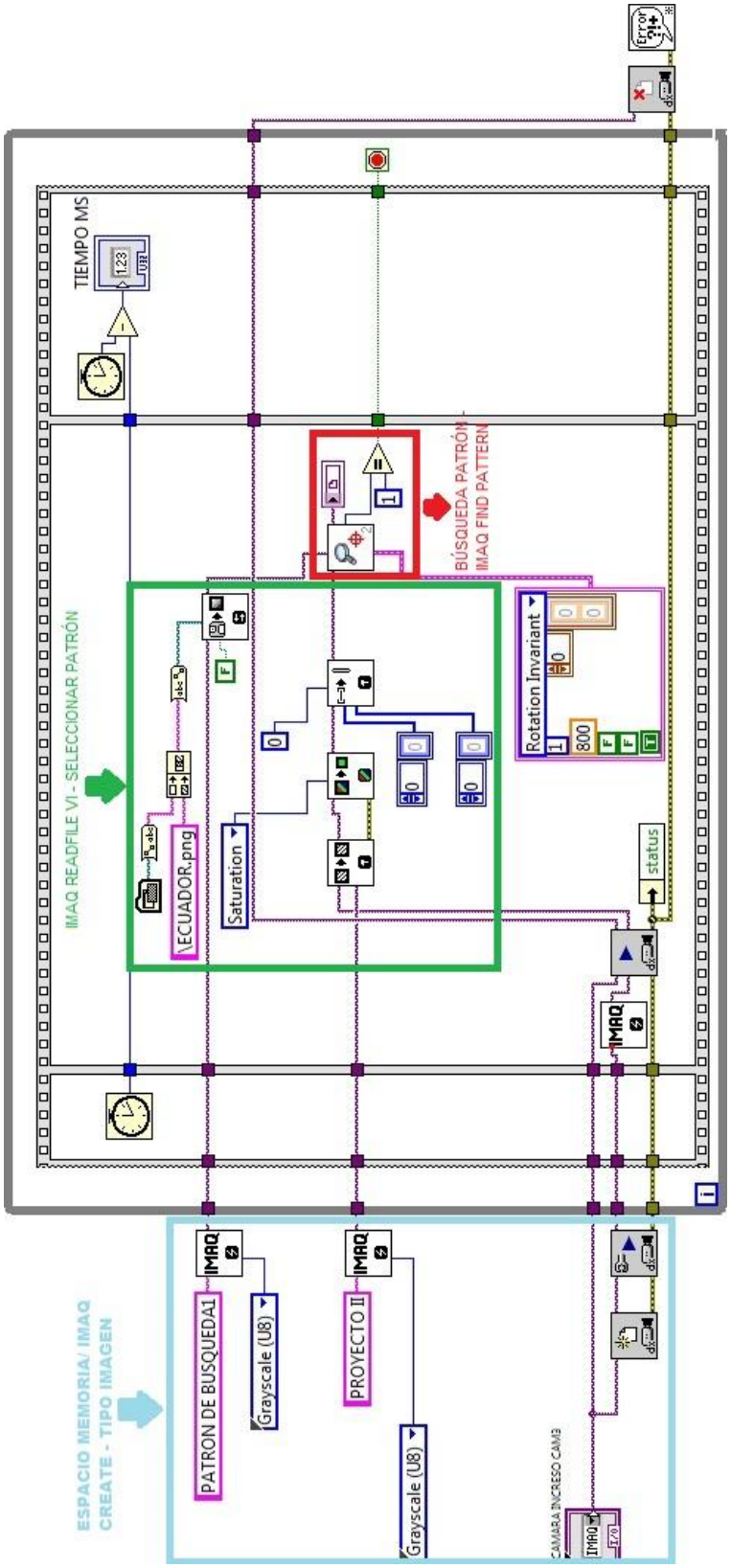
ANEXO 4

PROGRAMACIÓN EN LABVIEW

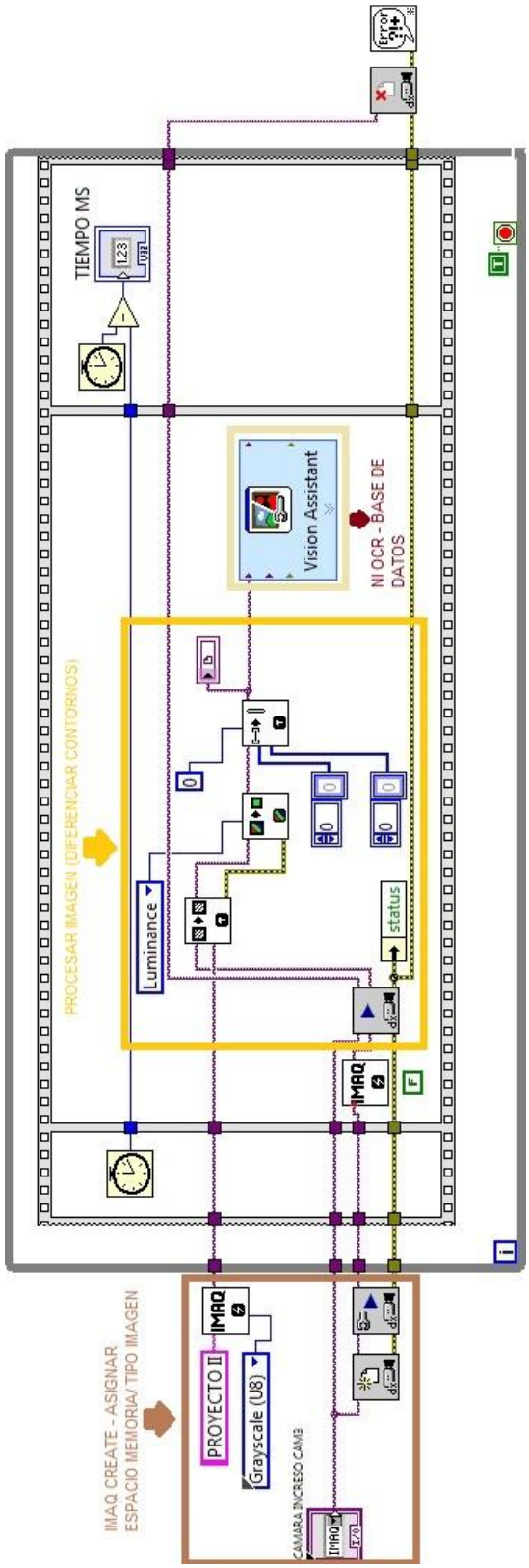
♣ SOFTWARE PARA EL ALGORITMO DE COINCIDENCIA DE COLORES



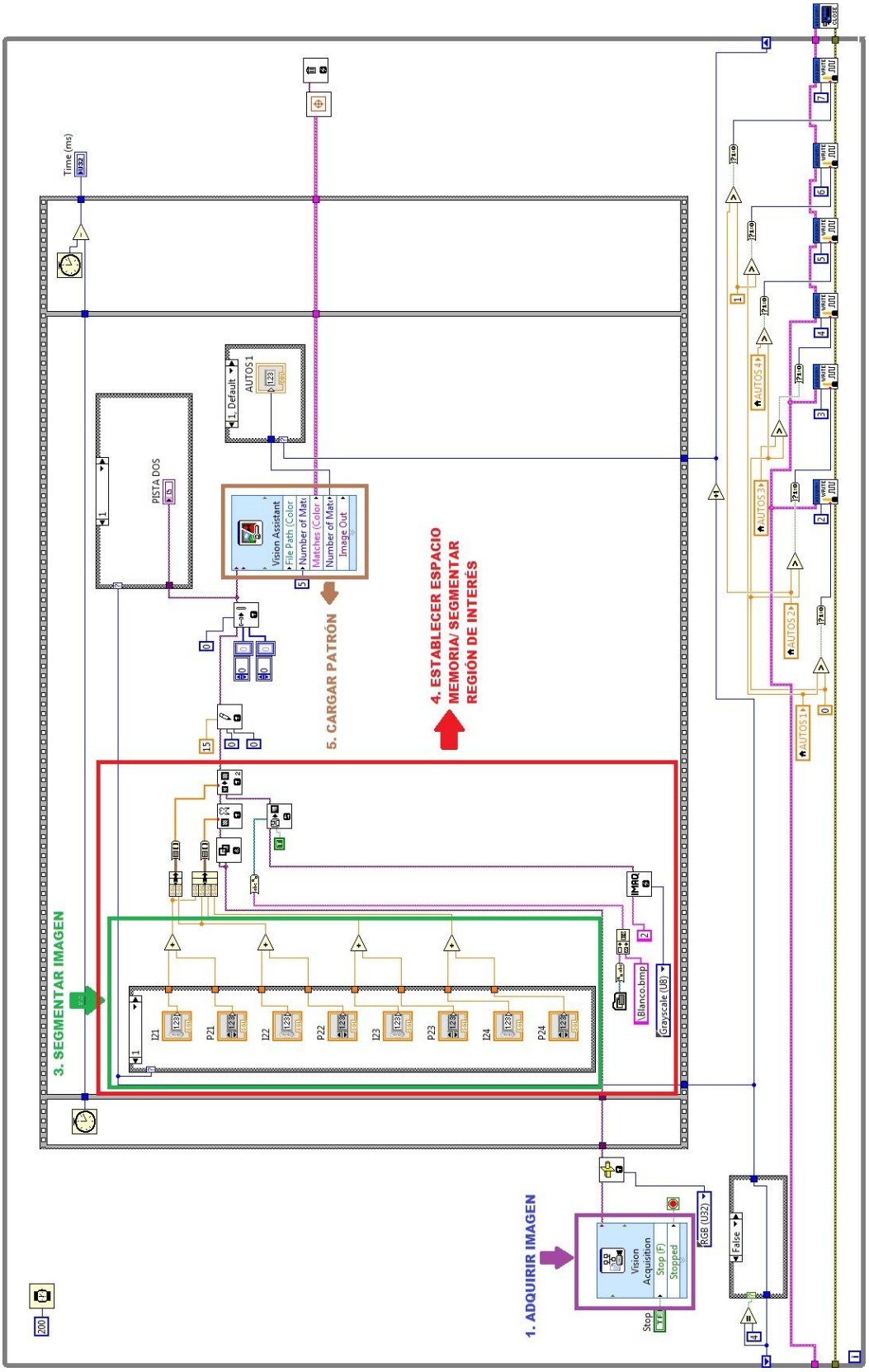
♣ SOFTWARE PARA EL ALGORITMO DE CORRELACIÓN CRUZADA



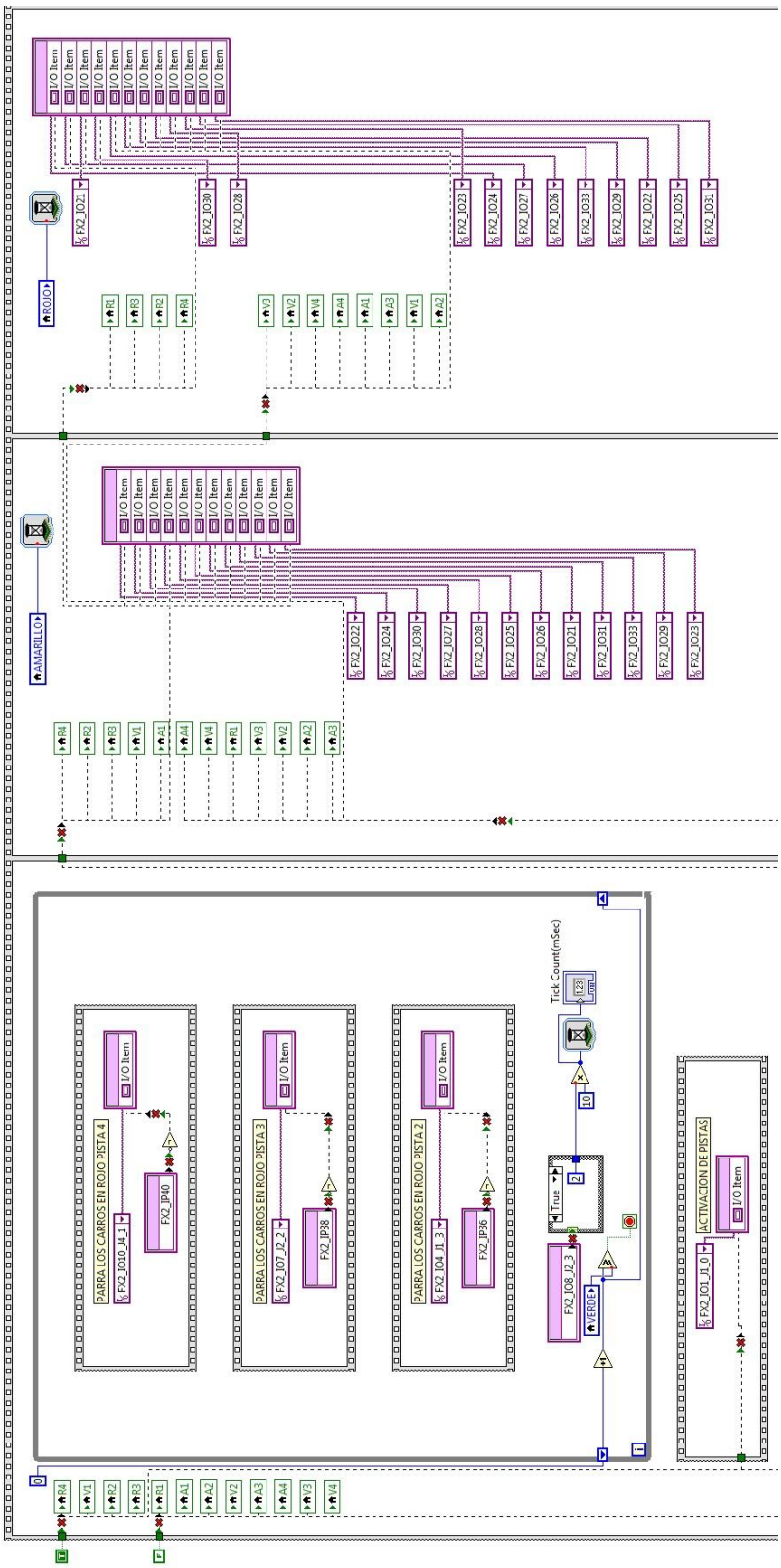
- **SOFTWARE PARA EL ALGORITMO DE RECONOCIMIENTO ÓPTICO DE CARACTERES (OCR)**



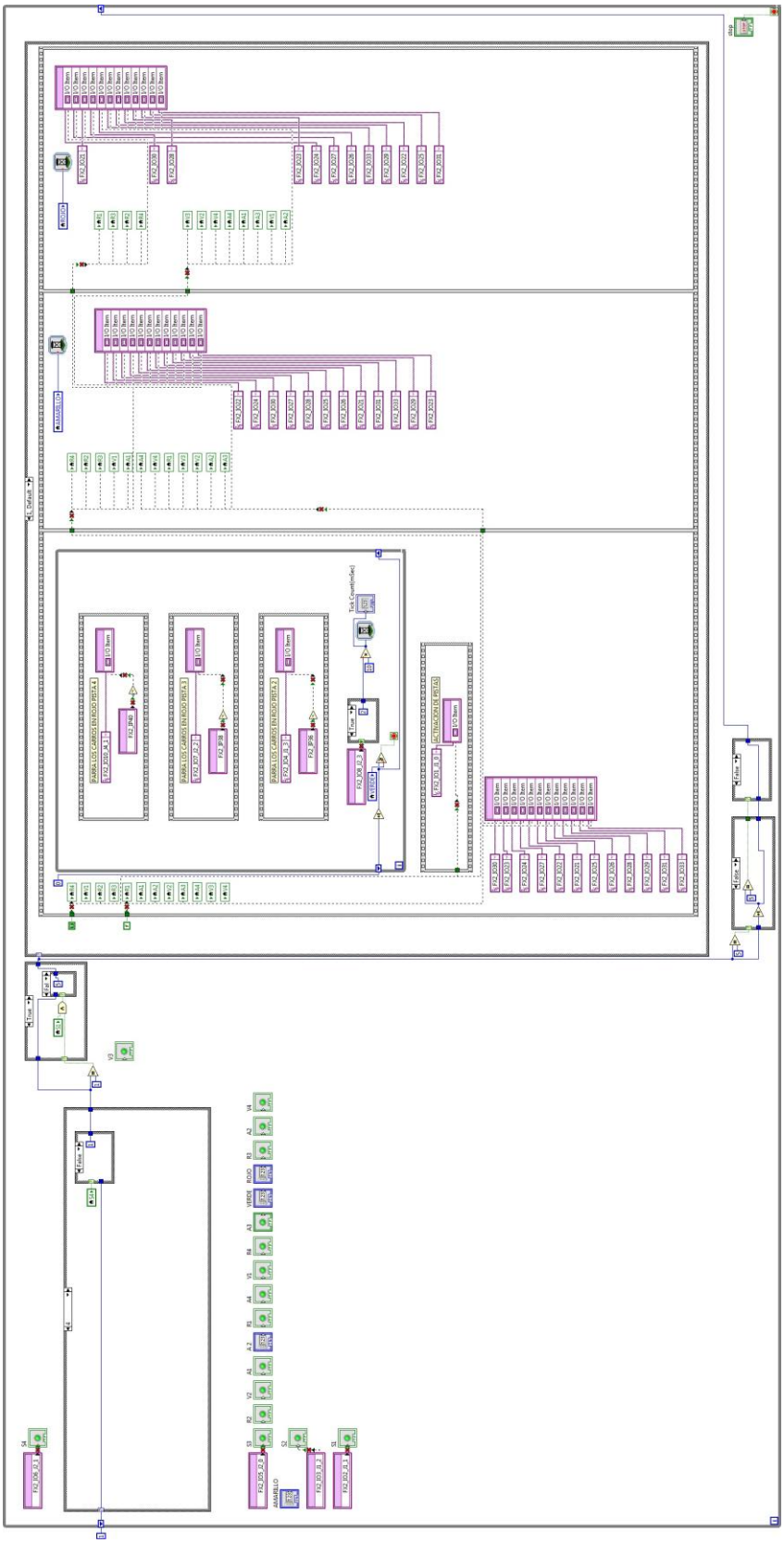
♣ SOFTWARE DE CONTROL – VISIÓN ARTIFICIAL



♣ SOFTWARE PARA LA MÁQUINA DE ESTADOS



♣ SOFTWARE PARA LA FPGA



REFERENCIAS

[1] Baccaglioni, F. (2013). *La historia del semáforo*. Recuperado de http://www.redbull.com/cs/Satellite/es_AR/Article/La-historia-del-sem%C3%A1foro-021243312125605

[2] García, I. (2008). *Visión artificial y Procesamiento Digital de Imágenes usando Matlab.* Recuperado de http://vision-artificial-matlab.blogspot.com/

[3] *Recuperación de Información Multimodal*. (2011). Recuperado de http://recuperaciondeinformacionmultimodal.blogspot.com/2011/05/descriptores-de-imagenes.html

[4] Otiniano, C. (2013). *LabView VISION - Fundamentos del Procesamiento Digital de Imágenes.* Recuperado de https://decibel.ni.com/content/docs/DOC-31879

[5] *Tipos de sensores*. (s.f.). Recuperado de http://www.profesormolina.com.ar/tecnologia/sens_transduct/tipos.htm

[6] *Fundamentos del Entorno de NI LabView.* (s.f.). Recuperado de http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

[7] Reyes, J. (2008). *Reconocimiento de Patrones.* Recuperado de http://www.maginvent.org/articles/pidht/pidtoot/Reconocimiento_Patrones.html

[8] Gordillo, L. & Yánez, J. (2009). *Aplicación de visión con LabVIEW para la detección de frascos con turbiedades.* (Tesis Ing. Electrónica y Telecomunicaciones). Escuela Superior Plitécnica del Litoral. Guayaquil, Ecuador.

[9] Pérez, J. (2008). *Correlación y Orientaciones. (*Apuntes de Fotogrametría). Universidad de Extremadura. España.

[10] *Sistemas de Análisis por Imágenes.* (s.f.). Recuperado de http://www.analiticasa.com.ar/index_optica_sistemas_lavision.html

[11] Noelia, R. (2007). *FPGA*. Recuperado de http://reinaldo-noelia-fpga.blogspot.com/2007/12/fpga_4670.html

[12] *Aplicaciones generales de una FPGA.* (s.f.). Recuperado de http://www.generatecnologias.es/aplicaciones_fpga.html

[13] *Arduino*. (s.f.). Recuperado de http://www.arduino.cc/es/

[14] Reyes, L. (2011). *Diseño de Experimentos al Completo Azar*. Recuperado de http://reyesestadistica.blogspot.com/2011/07/diseno-de-experimentos-al-completo-azar.html

BIBLIOGRAFÍA

- García, A. (2012). *Inteligencia Artificial. Fundamentos, práctica y aplicaciones*. 1ª.ed. Madrid, España: RC Libros. pp. 171-233.

- Nilsson, N. (2001). *Inteligencia Artificial. Una nueva síntesis*. 1ª.ed. Madrid, España: McGraw-Hill Interamericana de España. pp. 33-62, 75-99.

- National, I. (2006). *LabVIEW Básico: Manual de curso*. *Washington DC*, Estados Unidos: National Instruments. pp. 45-62.

- Giraldo, C. (2007). *Dimensionamiento de piezas usando el sistema de visión de la celda de manufactura flexible en la Facultad de Ingeniería Mecánica*. (Tesis Ing. Mecánico). Universidad Tecnológica De Pereira. Pereira, Colombia.

- Sobrado, E. (2003). *Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot*. (Tesis Magister en Ingeniería de Control Y Automatización). Pontificia Universidad Católica del Perú. Lima, Perú.

- *Cámaras.* (s.f.). Recuperado de http://www.infaimon.com/es/camaras-industria

- Palazzesi, A. (2006). *CCD vs. CMOS*. Recuperado de http://www.neoteo.com/ccd-vs-cmos/

- *Sensores CCD y CMOS*. (s.f.). Recuperado de http://www.videovigilancia.eu.com/blog/videovigilancia/sensores-ccd-y-cmos

- *Micro Switch Basic Switches Line Guide.* (2011). Recuperado de http://sensing.honeywell.com/honeywell-sensing-micro-switch-basic-switches-line-guide-004649-7-en.pdf

- Roncancio, H. & Cifuentes, H. (2001). Universidad Distrital Francisco José de Caldas. *Tutorial de Labview.* Recuperado de http://perso.wanadoo.es/jovilve/tutoriales/016tutorlabview.pdf

- *Tutorial de Labview.* (s.f.). Recuperado de http://webs.uvigo.es/euiti_ie1/ie2/Tutorial%20de%20Labview.pdf

- Descripción de MATLAB. (s.f.). Recuperado de http://nereida.deioc.ull.es/~pcgull/ihiu01/cdrom/matlab/contenido/node2.html

- Universidad de Oviedo. (2007). *Práctica 1: Introducción a matlab.* Recuperado de http://orion.ciencias.uniovi.es/~riera/modelado/practicas/pract_01.pdf

- *Historia de opencv.* (2012). Recuperado de http://ocw.unia.es/ciencias-tecnologicas/tecnologia-del-ocio/materiales-basicos-folder/html/B2_U3/historia_de_opencv.html/skinless_view

- *Reconocimiento de Patrones.* (s.f.). Recuperado de http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/gutierrez_g_l/capitulo2.pdf

- Carrasco, J. (s.f.). *Reconocimiento de Patrones.* Recuperado de http://ccc.inaoep.mx/~ariel/recpat.pdf

- Begueri, M. (2005). *Estudio experimental del flujo en una cavidad por técnicas de seguimiento de imágenes de partículas.* (Tesis Ing. Mecánica). Universidad de Buenos Aires. Buenos Aires, Argentina.

- Tasiguano, C. (2011*). Desarrollo de algoritmos de reconocimiento de placas de vehículos.* (Tesis Ing. Electrónica y Control). Escuela Politécnica Nacional. Quito, Ecuador.

- Sánchez, C. & Sandonís, V. (s.f.). *Reconocimiento Óptico de Caracteres (OCR).* Recuperado de http://www.it.uc3m.es/jvillena/irc/practicas/08-09/09.pdf

- *Reconocedores ópticos de caracteres*. (s.f.). Recuperado de http://grupo.us.es/gtocoma/pid/pid10/OCR.htm

- Cazorla, A. & Alados, L. (2005). *Estimación de la cubierta nubosa en imágenes de cielo mediante el algoritmo de clasificación KNN.* Recuperado de http://www.aet.org.es/congresos/xi/ten76.pdf

- D'Amato, J. & Bauza, C. (s.f.). *Procesamiento de imágenes para la clasificación masiva de frutos basado en el color*. Recuperado de http://www.plademan.net/~cgarcia/publications/JIDIS-2007.pdf

- Andrade, G., López, J. & Chávez, P. (s.f.). *Sistema de control vehicular utilizando reconocimiento óptico de caracteres.* Recuperado de www.dspace.espol.edu.ec/bitstream/123456789/1458/1/2973.pdf