SEP **TECNM** **DITD**

INSTITUTO TECNOLÓGICO SUPERIOR DE ATLIXCO

Organismo Público Descentralizado del Gobierno del Estado de Puebla

ALGORITMO DE VISIÓN ARTIFICAL PARA EL CÁLCULO DE LA ORIENTACIÓN DE COMPONENTES ELECTRÓNICOS APLICADOS A PROCESOS DE MANUFACTURA DE ALTA VELOCIDAD

OPCIÓN I.

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

Ingeniero Mecatrónico

P R E S E N T A:
Jesús Ángel Aragón Morales

ASESOR: **Dra. Mariana Natalia Ibarra Bonilla**

ATLIXCO, PUE. ENERO DE 2019

AGRADECIMIENTOS

Agradezco principalmente a Dios por guiarme a lo largo de mi vida y de mi carrera, por darme la fortaleza y la sabiduría necesaria para alcanzar mis objetivos, por darme una vida llena de felicidad, rodeada de personas maravillosas con la cuales he compartido experiencias y lecciones de vida.

Le agradezco a mis padres Pedro y Yuvicela por darme la vida y su apoyo incondicional, especialmente le estoy eternamente agradecido a mi hermosa madre por haberme dado una excelente educación, por su inmenso amor, sus incontables cuidados y desvelos, porque siempre estuvo conmigo cuando más lo necesitaba, además le agradezco por siempre creer en mí, por todos los consejos y las herramientas necesarias que me ha brindado para poder sobresalir y ser una persona de bien. Gracias mamá, por darme tanto, porque todo lo que soy es por ti y prometo que éste es uno de muchos logros que te dedicaré, déjame decirte que te amo con todo mi corazón, y aunque las cosas puedan llegar a salir mal, con tu amor y anhelo soy capaz de lograr hasta lo incapaz.

También le doy las gracias a mis hermanos Aline y Raúl por ser parte importante de mi vida, por todos los buenos momentos que hemos pasado juntos, por su gran cariño y su grata compañía, por apoyarme en cada decisión que he tomado y por sus incontables consejos, también agradezco a cada una de las personas que me han apoyado y acompañado a lo largo de mi vida.

Por último, pero no menos importante le agradezco a todos y cada uno de mis profesores por trasmitirme su conocimiento y prepararme para mi etapa como profesionista. Especialmente quiero agradecerle a mi asesora la Dra. Mariana Natalia Ibarra Bonilla y para el Dr. Fernando Quiñones Novelo quienes fueron los primeros en creer en mí y en este proyecto, me apoyaron de forma personal e institucional, además de alentarme para que concluyera este trabajo. Gracias por compartir sus conocimientos, por su comprensión, por su confianza depositada en mí y sobre todo por su amistad.

ÍNDICE GENERAL

INTRODUCCIÓN ..... **I**

CAPÍTULO 1. PROTOCOLO DE INVESTIGACIÓN **1**

1.1 Planteamiento del problema 1

1.2 Objetivos 2

1.2.1 Objetivo General *2*

1.2.2 Objetivos Específicos *2*

1.3 Justificación 3

1.4 Alcances del proyecto 4

1.5 Limitación del proyecto 4

CAPÍTULO 2. MARCO TEÓRICO **5**

2.1 Antecedentes 5

2.1.1 ProtoPlace S *5*

2.1.2 Patente US7813559B2 ..... *6*

2.2 Estado del arte 6

2.3 Tecnología SMT 7

2.4 Dispositivos de montaje superficial 8

2.5 Placas de Circuito Impreso 9

2.6 Equipo de Montaje Superficial 9

2.7 Sistemas Electrónicos Embebidos 10

2.8 FPGAs 11

CAPÍTULO 3. PROGRAMACIÓN DE LA FPGA EN VHDL **14**

3.1 Software ISE Design Suite 14.7 14

3.1.1 Interfaz de Usuario *14*

3.2 FPGA ASSERTA 16

3.2.1 Características de la tarjeta ASSERTA *17*

3.3 Módulo de la cámara VGA OV7670 18

3.3.1 Configuración de los pines del módulo *20*

3.3.2 Funcionamiento y configuración del módulo *21*

3.4 Programación en VHDL 23

3.4.1 Programación del TOP *24*

3.4.2 Binarización *28*

CAPÍTULO 4. PROGRAMACIÓN EN MATLAB **31**

4.1 EXTRACCIÓN DE CARACTERÍSTICAS DE IMÁGENES BINARIAS 31

4.1.1 Tamaño *33*

4.1.2 Posición *33*

4.1.3 Orientación *34*

4.1.4 Perímetro y circularidad *38*

4.1.5 Envolvente convexa y MBR *39*

4.1.6 Número de agujeros y número de Euler *40*

4.2 EXTRACCIÓN DE CARACTERÍSTICAS CON MATLAB 41

4.2.1 Comando "regionprops" *41*

4.2.2 Lectura y binarización de la imagen *46*

4.2.3 Cálculo del área y los momentos de inercia de primer y segundo orden *47*

4.2.4 Cálculo de la Orientación *49*

CAPÍTULO 5. RESULTADOS **51**

5.1 MONTAJE DEL HARDWARE Y ÁREA DE TRABAJO 51

5.2 PRUEBAS DE DESEMPEÑO CON IMÁGENES DE FIREWORKS 54

5.2 PRUEBAS DE DESEMPEÑO CON IMÁGENES CAPTURADAS POR LA CÁMARA OV7670 61

CONCLUSIONES **66**

REFERENCIAS **68**

ANEXO **71**

ÍNDICE DE FIGURAS **74**

ÍNDICE DE TABLAS **75**

INTRODUCCIÓN

La tecnología de montaje superficial SMT, actualmente es la tendencia en lo que se refiere al desarrollo de productos electrónicos debido a la miniaturización en los productos, esta tecnología cumple con los requerimientos de tiempos, calidad y costos para manufacturarlos.

En México hay diferentes empresas de manufactura electrónica, pero la mayoría son extranjeras, por lo que dependen de los centros de desarrollo de sus países de procedencia, y por lo tanto gran parte de la ingeniería del producto ya está definida y lista para que el producto sea ensamblado en sus líneas de producción.

Actualmente la manufactura de tarjetas electrónicas con la tecnología SMT está altamente automatizada, y a pesar de que el avance tecnológico es muy rápido, en México existe una limitación en cuanto a la manufactura de productos innovadores y funcionales que puedan competir en el mercado internacional, pues siempre se recurre a la maquinaria extranjera. Por ello, el desarrollo profesional de ingenieros mexicanos se debe centrar en la automatización de la manufactura electrónica, desde el diseño, desarrollo y ensamble.

La visión artificial es una disciplina que engloba todos los procesos y elementos que proporcionan “ojos” a máquinas, siendo parte esencial en el proceso de manufactura electrónica para mejorar la calidad de los productos ensamblados con la tecnología de montaje superficial, con ello se pretende mejorar el desarrollo de la tecnología mexicana para comenzar a crear productos de calidad que compitan internacionalmente y éste trabajo busca formar parte de esas mejoras, al ensamblar tarjetas electrónicas y ser implementadas por una empresa mexicana.

Este trabajo consta de 5 capítulos que a continuación se describen brevemente:

En el capítulo 1 se presenta el protocolo de investigación, en donde se menciona la situación problemática, los objetivos, la justificación, los alcances y limitaciones que conforman y han dado surgimiento a este trabajo de tesis.

En el capítulo 2 se muestra el marco teórico, donde se describen algunos conceptos que sirven para comprender el trabajo desarrollado, inicialmente se presentan los antecedentes y el estado del arte del proyecto, para posteriormente presentar brevemente los conceptos teóricos que abarcan desde la descripción de la tecnología de montaje superficial y las máquinas que lo realizan, así como la información del sistema embebido FPGA, su arquitectura y los fabricantes más importantes a nivel internacional.

En el capítulo 3 se presenta una breve descripción del entorno del software de programación en VHDL, así como las generalidades de la tarjeta de desarrollo ASSERTA y el módulo de la cámara OV7670 utilizados en el proyecto. Se explica el desarrollo de la programación del módulo principal en VHDL que se implementó para hacer trabajar la cámara en la FPGA de la tarjeta ASSERTA y del proceso de binarización.

En el capítulo 4 se describen algunas de las propiedades determinadas con la extracción de las características de una imagen binaria y se presenta el desarrollo de la programación en Matlab para determinar el área y los momentos de inercia de los objetos captados por la cámara OV7670, para calcular los grados de desfase que tienen.

En el capítulo 5 que es el último de este trabajo de tesis se presentan los resultados finales donde se muestra el funcionamiento logrado de la tarjeta FPGA y el módulo OV7670. También se muestra el desempeño del algoritmo de Matlab con algunas imágenes de prueba frente a los comandos establecidos en Matlab.

CAPÍTULO 1. PROTOCOLO DE INVESTIGACIÓN

En el presente capítulo se describe el protocolo de la investigación que ha dado surgimiento a este trabajo de tesis.

1.1 Planteamiento del problema

INTESC electrónica & embebidos [1] es una empresa mexicana dedicada al diseño y desarrollo de sistemas embebidos basadas en FPGAs (del inglés, *Field-Programmable Gate Array*) y SoCs (del inglés, *Systems on Chips*), así como a la producción de nueva tecnología. Para la fabricación de circuitos, los ingenieros de INTESC utilizan la tecnología de montaje superficial (SMT, *Surface-Mount Technology*). La División de Ingeniería Mecatrónica del ITSA se acercó a la empresa INTESC para que los estudiantes conozcan el proceso de fabricación de tarjetas de circuito impreso (PCB, *Printed Circuit Board*) usando SMT, con el objetivo de que los estudiantes puedan fabricar sus propias tarjetas de circuito impreso con un acabado profesional, tal como se lleva a cabo en la industria.

La SMT es el proceso de construir circuitos electrónicos en donde los componentes están soldados directamente sobre la superficie de la placa de circuito impreso, PCB. Actualmente, la SMT ha reemplazado a la técnica de la tecnología de agujero pasante (*through hole*), en donde el componente atraviesa la placa, pues con SMT las placas son de menor tamaño y ofrece mejor compatibilidad electromagnética, es decir un menor número de emisiones radiadas, debido al espacio de radiación reducido [2].

Sin embargo, SMT exige más habilidad por parte de los ingenieros, pues el ensamble de los componentes es muy difícil debido a los tamaños reducidos de los componentes. Actualmente, en INTESC, para fabricar las tarjetas una persona está encargada de realizar el montaje de los componentes electrónicos. Por ello, han desarrollado una máquina *Pick & Place* cuya función es tomar los componentes electrónicos (resistencias, capacitores y circuitos integrados) y montarlos en la posición, que le indique un ordenador, sobre la PCB, y así aumentar la producción, disminuir el tiempo de ensamble y sustituir al operador. El problema que presenta actualmente la máquina *Pick & Place*,

desarrollada por INTESC, es el posicionamiento de los circuitos integrados, pues éstos últimos son colocados sobre bandejas alimentadoras que no proporcionan una posición exacta y resulta en un desfase en la posición. Para ello se requiere incorporar a la máquina un sistema de corrección en la posición para disminuir el desfase.

Como solución a este problema, se ha propuesto incorporar un sistema de visión que sirva como retroalimentación al lazo de control y así disminuir el desfase al orden de las micras. Para esto, como etapa preliminar, es necesario desarrollar un algoritmo de procesamiento de imágenes que sea capaz de calcular de forma autónoma el ángulo de rotación de los componentes y que no haga uso de librerías para que sea llevado a una implementación en hardware.

1.2 Objetivos

Se presentan las metas establecidas para el proyecto que se deben cumplir a lo largo del desarrollo e implementación.

1.2.1 Objetivo General

Desarrollar y comprobar el desempeño de un algoritmo de visión, en el software Matlab, que use el método del momento de segundo orden de inercia para calcular el ángulo de rotación de los componentes, que una máquina Pick & Place debe soldar, con respecto a una cámara RGB 0V7670 para una implementación en un FPGA.

1.2.2 Objetivos Específicos

- Crear en VHDL el módulo principal que establezca la interacción de los submódulos necesarios para lograr el funcionamiento de la cámara y los demás procesos.
- Realizar el procesamiento en “tiempo real” del video obtenido por el módulo OV7670, para el proceso de binarización en el FPGA.
- Mostrar en “tiempo real” el vídeo ya procesado en una pantalla TFT ensamblada en la FPGA.

- Desarrollar la comunicación entre el FPGA y la computadora para transferir fotografías capturadas por la cámara, cada vez que sean solicitadas por una aplicación de escritorio.
- Calcular el área y los momentos de inercia de los objetos binarizados captados por la cámara.
- Determinar la orientación de los objetos captados en base a los momentos de segundo orden.

1.3 Justificación

Debido a que el proceso de ensamble SMT requiere trabajar a alta velocidad y precisión, el sistema de visión propuesto debe implementarse en una plataforma computacional capaz de procesar imágenes con alto rendimiento, siendo los procesadores de señales digitales (DSP, Digital Signal Processors) los que han sido ampliamente usados en aplicaciones de alto rendimiento y procesamiento de una gran cantidad de datos [3]. Los FPGA representan una alternativa superior a los DSP, pues los FPGA no tienen una arquitectura en hardware definida, pues puede ser configurada de acuerdo a los requerimientos del usuario usando lenguajes de descripción de hardware, como VHDL, Verilog HDL, etc. En contraste, los DSP tienen una arquitectura fija con memoria, controlador e instrucciones ejecutadas secuencialmente de acuerdo con un software.

Los algoritmos de procesamiento de imágenes deben procesar una gran cantidad de datos (pixeles) a alta velocidad y para este tipo de aplicaciones, los FPGA son más adecuados, debido a su capacidad de procesamiento en paralelo. Lo anterior no es posible con los DSP debido a su arquitectura de procesamiento de datos, la cual está definida como 8-bits, 16-bits, 32-bits, etc., por lo que la capacidad de procesamiento de un FPGA es más eficiente que un DSP.

Para generar en algoritmo de visión en lenguaje VHDL, primero es necesario realizar una búsqueda de los ya existentes, para seleccionar el mejor para esta aplicación, y validarlo en un software, debido a que algunas operaciones matemáticas básicas son complicadas de implementar en un FPGA. El software seleccionado en Matlab, debido a que es ampliamente usado en ingeniería y permite programar en lenguaje C, sin usar comandos

o funciones de las librerías de Matlab. Por ello, el presente proyecto propone el desarrollo y validación del algoritmo de visión en Matlab para su posterior traducción al lenguaje VHDL.

1.4 Alcances del proyecto

- Generar el código en lenguaje C que calcule el error en la posición de los componentes y sirva para su futura implementación en FPGA en la máquina industrial *Pick&Place*.
- Generar el código bajo las condiciones que exige el lenguaje VHDL.

1.5 Limitación del proyecto

- El proyecto se limita a la generación del código en un software, y no llega hasta la implementación en el FPGA.

CAPÍTULO 2. MARCO TEÓRICO

En el presente capítulo se presentan algunos conceptos que sirven para comprender el trabajo desarrollado. Al tratarse de un trabajo de tesis, se presentan inicialmente los antecedentes y estado del arte del proyecto, para posteriormente presentar brevemente los conceptos teóricos del proyecto.

2.1 Antecedentes

Uno de los objetivos de la empresa INTESC es desarrollar tecnología nacional con productos de calidad y accesible a clientes, empresas y/o instituciones. Por ello, aunque comercialmente existen máquinas de *Pick & Place* que incluyen sistemas de visión, el propósito del proyecto no es adquirir una tecnología ya existente, ni desarrollar la máquina, sino desarrollar el sistema de visión en Matlab, que sirva para su implementación en un FPGA, para reducir el error en el posicionamiento de los componentes. Cabe mencionar que la adquisición de este tipo de máquinas puedes tener un precio base desde $4100 USD, más costos de envío e importación [4].

En la búsqueda de la anterioridad se encontraron múltiples tipos de máquinas *Pick & Place* industriales para diferentes aplicaciones, pero para este trabajo, solo son importantes las máquinas *Pick & Place* utilizadas para montaje SMD profesional de PCBs y series pequeñas en laboratorio. Por lo que a continuación se describen aquellas que más se asemejan.

2.1.1 ProtoPlace S

La ProtoPlace S, desarrollada por la empresa alemana LPKF Laser & Electronics [5], es una máquina Pick & Place semiautomática. La diferencia principal de esta máquina con nuestra propuesta es que la cámara se incorpora para enviar la imagen a un monitor, de esta forma le permite al usuario rastrear y controlar la colocación de piezas pequeñas con mucha precisión. En el caso de nuestra propuesta el sistema de video permitirá realizar el ajuste del posicionamiento de manera autónoma, sin necesidad del usuario.

Figura 2.1 Máquina Pick & Place ProtoPlace S [5]**.**

2.1.2 Patente US7813559B2

En el caso de patentes se encontró la US7813559B2, titulada "*Image analysis for pick and place machines with in situ component placement inspection*" [6]. En esta se describe un método para determinar la ubicación de un componente sobre una pieza de trabajo. El método consiste en tomar una serie de fotografías, antes y después de la colocación de la pieza, y mediante la comparación de una imagen estándar, calculan por diferencia el desplazamiento de las piezas.

De acuerdo con lo anterior, aunque se aplica a las máquinas Pick & Place para también compensar el error en la posición de los componentes, no es similar a nuestra propuesta, debido a que esta patente hace uso de potentes ordenadores conectados al sistema de visión para procesar video en tiempo real. En cambio, nuestra propuesta empleará tecnología embebida de FPGA para el procesamiento de video en tiempo real, dotando a la máquina de una autonomía sin necesidad de un ordenador adicional.

2.2 Estado del arte

Se realizó una búsqueda en exploradores científicos, y se encontraron los siguientes artículos en la literatura: la tesis de licenciatura del Instituto Politécnico Nacional titulada "Pick and Place para montaje de componentes de montaje superficial SMD" [7] describe la adaptación de un dispositivo grabador de circuitos impresos en un robot Pick & Place para la colocación de dispositivos de montaje superficial en tarjetas de circuito impreso.

En el trabajo se describe que para alcanzar la máxima precisión monitorean la posición de los motores a pasos mediante una interfaz. Por lo que está propuesta no incorpora video.

El trabajo de Verstegen, Van Gastel y Spronck [8] nos describe una idea que propone el uso de una cámara y un sistema de espejos. Mediante el espejo una imagen de la parte inferior del componente es reflejada, la cámara registra esta imagen y el sistema procesa la posición del componente al momento de ser recogido por el manipulador. Posteriormente el sistema registra una imagen del sitio en donde será colocado el componente, y computa la diferencia en posiciones para lograr la reducción del error en la colocación. Nuestra propuesta no utiliza ningún tipo de espejo o sistema óptico adicional al de la cámara, y la principal diferencia es que se tomará una imagen antes de tomar el componente en la bandeja, para detectar la rotación y desplazamiento, respecto del manipulador, posteriormente se tomará una imagen del sitio para el posicionamiento final del componente.

De acuerdo con la búsqueda de la anterioridad, podemos determinar que nuestra propuesta de sistema de visión, basado en un sistema embebido de FPGA, representa una opción con una alta posibilidad de divulgación y de implementación en el sector industrial. Y aunque es bien sabido que el uso de un FPGA no es una propuesta nueva en el procesamiento de imágenes, su aplicación en la máquina Pick & Place dotará a este tipo de sistemas de una autonomía y desempeño ampliamente buscado en el sector industrial.

2.3 Tecnología SMT

La tecnología de montaje superficial o SMT tiene sus raíces en la tecnología de los paquetes planos y los híbridos de los años 1950 y 1960. El montaje superficial es una tecnología de empaque que monta componentes electrónicos en la superficie de una tarjeta de circuito impreso o PCB. [9] Actualmente es la tecnología que se emplea en las industrias de manufactura de tarjetas electrónicas. Esta tecnología supone muchos beneficios, en comparación con la ya obsoleta y cada vez más en desuso, tecnología de montaje de orificio pasante o THT. Entre los beneficios se puede mencionar que la SMT

proporciona un área reducida de montaje, lo que permite diseñar tarjetas electrónicas de tamaño reducido; Esto supone una mayor densidad de componentes por área de placa, lo que permite una mayor integración de sistemas electrónicos sobre una única placa.

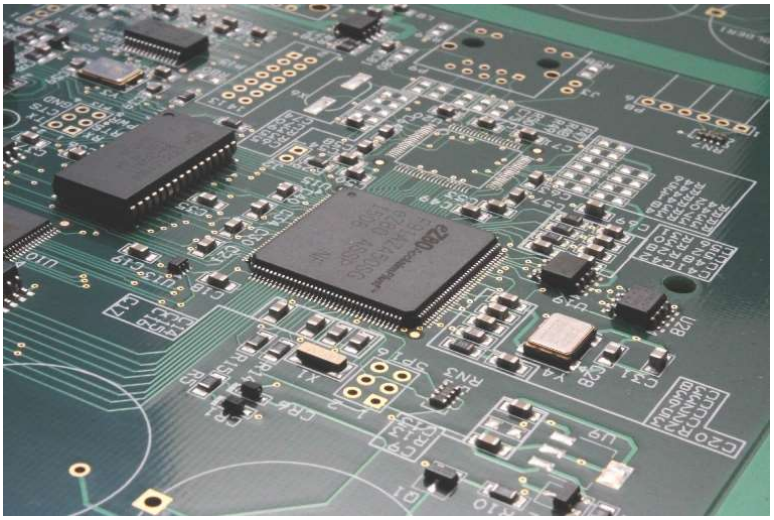


Figura 2.2 Tecnología de Montaje Superficial (SMT) [10].

2.4 Dispositivos de montaje superficial

Los componentes diseñados para SMT reciben el nombre de dispositivos de montaje superficial o SMD. Los SMD son de dimensiones reducidas y pueden ser empaquetados en recipientes especiales para una manipulación más eficiente a la hora de su colocación. Generalmente los recipientes suelen ser carretes de papel en donde se acomodan los componentes, para su distribución. Los SMD se diseñan de forma que permiten ser soldados superficialmente en la placa de circuito impreso. Su pines o patas eléctricas de conexión, se construyen en formas geométricas simples como rectángulos, círculos o una combinación de ambos. [11] Estas formas permiten generar áreas de soldadura planas, lo que a su vez provoca que las conexiones eléctricas sobre las placas se diseñen con estas formas. Los SMD son colocados sobre las placas de manera que ambas áreas se toquen entre si superficialmente y la soladura se adhiera a ambas partes. Esta técnica de soldadura y montaje da el nombre a la tecnología SMT.

2.5 Placas de Circuito Impreso

Las placas de circuito impreso o PCB, son placas especiales que proporcionan el cableado eléctrico para la interconexión de los SMD, además de un soporte físico de los mismos. Las PCB se fabrican con diferentes materiales como por ejemplo baquelita, fibra de vidrio, teflón o una combinación de estos materiales; así mismo emplea metales para la interconexión de los componentes como: el cobre, oro, plata, platino, plomo y estaño. Para proteger la superficie de las placas y evitar la oxidación de los metales, se emplean resinas dieléctricas a prueba de agua que se aplican sobre la superficie de las placas en forma de pintura de diferentes colores. El diseño de una PCB está orientado a generar la menor emisión electromagnética para disminuir la interferencia a otros dispositivos electrónicos. Las interconexiones eléctricas en una PCB se diseñan en capas o niveles, de manera que una capa puede emplearse para interconectar todas las líneas de voltaje; otro nivel para interconectar todas las líneas de la terminal común o GND, y por último niveles adicionales pueden emplearse para interconectar las líneas de comunicación entre los SMD. [12] Generalmente se diseñan placas de doble capa o de dos niveles, debido a los costos de producción de las mismas, ya que a medida que se incrementa los niveles de interconexión, lo mismo ocurre con el precio de fabricación. Las líneas o cables eléctricos que conectan a los SMD siempre son terminados en áreas de interconexión llamados pads. Los pads proporcionan el área metálica sobre la cual se soldán los SMD.

2.6 Equipo de Montaje Superficial

En la industria se emplean equipos de montaje superficial o SME para realizar las tareas de montaje de los SMD sobre las PCB. Estos equipos realizan tareas repetitivas de manera rápida, eficiente e incrementa la producción de forma drástica, en comparación con un operador humano. Un operador humano podría producir una placa completa, de unos 200 componentes en alrededor de 2hrs, mientras que una SME podría producir hasta 100 PCBs de las mismas características en el mismo tiempo [12]. Es por ello que, en la industria de fabricación de circuitos impresos, las máquinas SMD son ampliamente empleadas. Y resulta evidente que los costos de producción utilizando equipos decrecen considerablemente.

Un laboratorio de manufactura de SMT, involucra diferentes tipos de equipos, sin embargo, en esencia solos se requieren de 2 equipos para cubrir la tarea de montaje. Los 2 equipos más importantes son: el equipo llamado Pick and Place Equipment o PPE, que del idioma ingles se traduce como equipo de recoger y colocar; y el segundo es el horno de reflujo. Este último calienta la PCB junto con los SMD en ella, a una temperatura suficientemente alta como para derretir la soldadura de la PCB y soldar los componentes en ella [13].

Por otro lado, el PPE se encarga de tomar los SMD de diferentes carretes alimentadores y los coloca en su posición final correcta en la PCB. Dependiendo del tipo de PCB que se ensamble, los PPE pueden variar en su configuración mecánica y tipo de manipulador mecánico. Existen equipos simples que solo colocan componentes pasivos como resistencias, capacitores, bobinas y que no poseen una retroalimentación con respecto a la posición final de los componentes. Equipos de este tipo cuestan alrededor de unos $2000 USD a $3000 USD. Cuando el PPE incluye un sistema más complejo como la capacidad para colocación de circuitos integrados y retroalimentación por medio de procesamiento de imágenes, el costo puede rondar de entre $4100 USD a $20000 USD o incluso más elevado.

Los sistemas de visión por computador son una parte importante en la colocación de componentes en la tecnología SMT. Existen sistemas de procesamiento de imágenes con la capacidad de reconocimiento de objetos, y formas. Estos se emplean para determinar la posición inicial y final del componente que un PPE coloca en el PCB.

2.7 Sistemas Electrónicos Embebidos

Un sistema electrónico embebido o solamente sistema embebido, es el conjunto de dispositivos o elementos electrónicos que se interconectan para formar una arquitectura de cómputo, diseñado para cumplir con una tarea específica. Los sistemas embebidos generalmente se encuentran contenidos dentro de una sola PCB y no suelen tener el aspecto general de una computadora. Un ejemplo de sistema embebido podría ser la tarjeta electrónica que controla un reproductor de DVD, una tarjeta controladora de una lavadora o una PCB que controla el proceso de ensamble dentro de una fábrica de

automóviles, por mencionar solo algunos. Como se concluye de lo anterior, un sistema embebido contiene en sí mismo una computadora programable. El tipo de computadora que contiene un sistema embebido puede ser una maquina compleja de estados, un microprocesador, un microcontrolador o una combinación de éstos. Los sistemas embebidos pueden ser programados en diferentes lenguajes como, por ejemplo: lenguaje Ensamblador, C, C++, Python, Java y VHDL, solo por mencionar los más utilizados. Generalmente cuando un sistema embebido es programado, se diseña el sistema operativo para que cumpla con las tareas asignadas en tiempo real [14, 15]. Se considera que un sistema embebido procesa una tarea en tiempo real, cuando el sistema puede responder y procesar todas las variables involucradas en el proceso, dentro de un tiempo específico y periódico. Si por el contrario el sistema no es capaz de responder dentro de este tiempo y emplea diferentes tiempos de respuesta para procesar la misma información cada vez, se considera un sistema que procesa información no en tiempo real [15].

2.8 FPGAs

Los FPGA, siglas del inglés *Field Programmable Gate Arrays*, son dispositivos semiconductores que se basan en torno a una matriz de bloques lógicos configurables (CLBs), conectados a través de interconexiones programables. Un FPGA, es una tarjeta que según como se configure, puede realizar cualquier circuito digital. Los componentes lógicos programables pueden ser programados para duplicar la funcionalidad de puertas lógicas básicas tales como AND, OR, XOR, NOT o funciones combinacionales más complejas tales como decodificadores o simples funciones matemáticas.

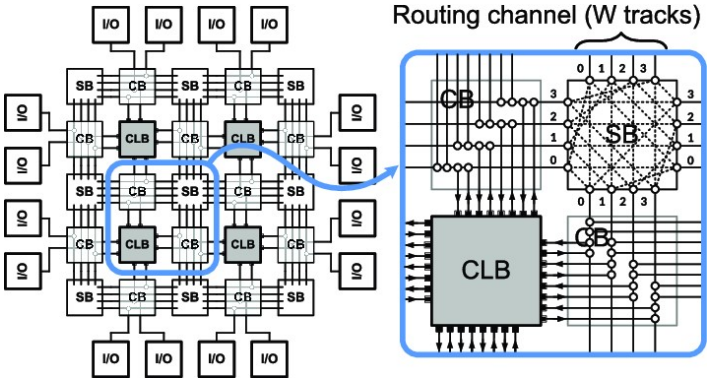


Figura 2.3 Arquitectura global de un FPGA [16].

Son los dispositivos programables por el usuario de aplicación más general. Un FPGA puede ser reprogramado para requisitos de las aplicaciones o funcionalidades deseadas después de la fabricación. Esta característica distingue a los FPGAs de circuitos integrados de aplicaciones específicas (ASIC), que se fabrican a medida para las tareas de diseño específicos. Una jerarquía de interconexiones programables permite a los bloques lógicos de un FPGA, ser interconectados según la necesidad del diseñador del sistema. Estos bloques lógicos pueden ser programados después de los procesos de manufactura por el usuario/diseñador, así que el FPGA puede desempeñar cualquier función lógica necesaria. A diferencia de los procesadores que se encuentran en una computadora personal, al programar un FPGA el chip se vuelve a cablear para implementar su funcionalidad en lugar de ejecutar una aplicación de software [15].

Con respecto a la arquitectura, cada chip de FPGA está hecho de un número limitado de recursos predefinidos con interconexiones programables para implementar un circuito digital reconfigurable y bloques de Entradas y Salidas para permitir que los circuitos tengan un acceso al mundo exterior. Los bloques de lógica configurables, también conocidos como CLBs, son la unidad de lógica básica de un FPGA. Algunas veces referido como segmentos o células de lógica, los CLBs están hechos de dos componentes básicos: Flip-Flops y Tablas de consulta (LUTs) o de sus siglas del inglés “look Up Table”. Es importante tomar esto en cuenta porque distintas familias de FPGAs se diferencian en la manera en que los Flip-Flops y las LUTs están empacados.

A finales del año 2005, el mercado de los FPGAs se ha colocado en un estado donde hay dos productores de FPGAs de propósito general que se encuentran a la cabeza del mismo, y un conjunto de otros competidores quienes se diferencian por ofrecer FPGAs de capacidades únicas. Existen muchos más fabricantes de FPGAs pero aquí solo se han mencionado los más relevantes a nivel internacional [15]:

- Xilinx es uno de los grandes líderes de la producción de FPGAs. Xilinx es una compañía de tecnología americana, primeramente, un distribuidor de dispositivos lógicos programables. Es reconocida mundialmente por inventar los FPGAs y también por ser la primera compañía con modelos de manufactura Fabless*

- Altera es el otro gran líder de la producción de los FPGAs. Altera corporation es uno de los pioneros de la lógica programable. Desarrolla algunas características que están orientadas hacia capacidad de sistemas en chips programables.
- Lattice Semiconductor es un fabricante de FPGAs que lanzo grandes FPGAs con tecnología de 90nm. En adición, Lattice es un proveedor líder en tecnología no volátil, FPGAs basados en tecnología flash, con productos de 90nm y 130nm.
- Actel tiene FPGAs basadas en tecnologías flash reprogramables. También ofrece FPGAs que incluyen mezcladores de señales basados en Flash.
- QuickLogic tiene productos basados en fusibles, es decir, solo son programables una sola vez.
- Atmel es uno de los fabricantes que sus productos no son re-configurables. Ellos se enfocaron en proveer microcontroladores AVR con FPGAs, todo en el mismo encapsulado.

Figura 2.4 Fabricantes de FPGAs.

CAPÍTULO 3. PROGRAMACIÓN DE LA FPGA EN VHDL

En este capítulo se presenta una breve reseña del entorno de programación en VHDL y del software utilizado, así como las generalidades de la tarjeta de desarrollo ASSERTA y el módulo de la cámara OV7670 utilizados en este proyecto. También se explica el desarrollo de la programación en VHDL que se implementó para hacer trabajar el módulo en la FPGA de la tarjeta ASSERTA.

3.1 Software ISE Design Suite 14.7

ISE Design Suite 14.7 es un software desarrollado por Xilinx para sintetizar y analizar diseños de HDL (Hardware Description Language), el cual permite al desarrollador sintetizar ("compilar") sus diseños, realizar análisis de tiempo, examinar diagramas RTL, simular la reacción de un diseño, y configurar diversos dispositivos de destino con el programador. Xilinx ISE (Integrated Synthesis Environment) es un entorno de diseño para productos FPGA y está estrechamente relacionado con la arquitectura de dichos chips [17].

Figura 3.1 Logotipo del software ISE Design Suite de Xilinx [18].

3.1.1 Interfaz de Usuario

La interfaz de usuario principal de ISE es el Project Navegator, que incluye varios paneles que se mencionan a continuación:

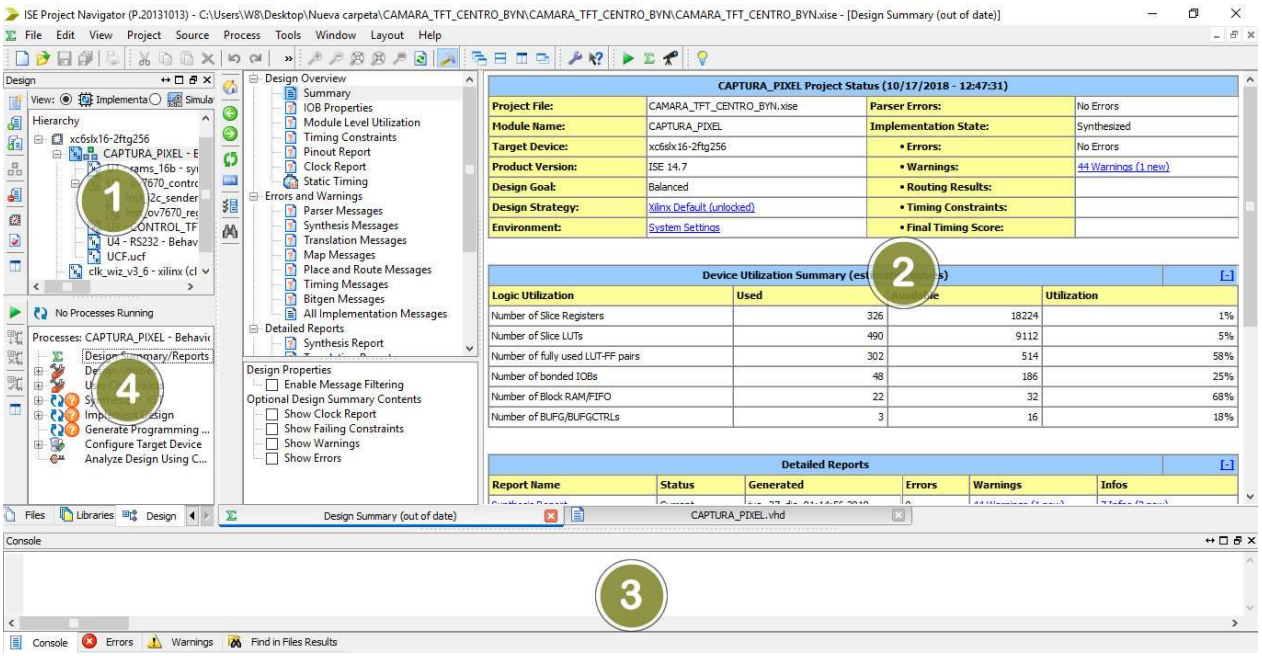


Figura 3.2 Interfaz de Usuario del software ISE Desing Suite.

1. Hierachy: La Jerarquía de diseño muestra los archivos de diseño, es decir los módulos VHDL, de descripción de hardware o de estímulos (test bench), que componen el proyecto, cuyas dependencias son interpretadas y mostradas por el ISE como una estructura de árbol, puede haber un módulo principal con otros módulos incluidos por el módulo principal, dichos módulos incluyen la configuración y asignación de pines, además se detalla el nombre del dispositivo FPGA utilizado y los nombres de todos los módulos.
2. Workplace: Es un editor de código fuente en el cual se puede mostrar desde un módulo VHDL y un reporte de uso del FPGA, hasta resúmenes de los recursos que la FPGA está consumiendo para el diseño implementado.
3. Console: La consola de salida o de transcripción proporciona el estado de las operaciones que están en ejecución además de informar sobre los problemas que se pueden encontrar en el diseño, dichos problemas pueden ser filtrados para mostrar Advertencias, Errores o ambos.
4. Process: En el árbol de procesos se describen las operaciones que realizará el ISE en el módulo actualmente activo. La jerarquía incluye funciones de compilación, sus funciones de dependencia y otras utilidades. También denota problemas o errores que surgen con cada función [17].

3.2 FPGA ASSERTA

Para realizar la programación del módulo OV7670 y hacer que la cámara funcione se empleó la tarjeta de desarrollo ASSERTA. La tarjeta de desarrollo ASSERTA, diseñada y manufacturada por la empresa mexicana INTESC, ha sido diseñada para satisfacer las necesidades de estudiantes, investigadores y profesionistas que trabajan con FPGAs, la tarjeta cuenta con numerosos recursos lo que la hace versátil para desarrollos de aplicaciones embebidas, además la hacen ideal para hacer adquisición de datos.

El núcleo de ASSERTA contiene un FPGA Spartan 6 XC6SLX16 que le permite diseñar sistemas de procesamiento en paralelo, cuenta con 12 convertidores analógico/digital (ADC), 4 convertidores digital/analógico (DAC), un compartimiento para integrar una pantalla TFT de 4.3” con opción de touchscreen capacitivo lo que le permite ser usada como un sistema autónomo de adquisición, 3 fuentes internas para el FPGA, una fuente externa de alimentación y un oscilador de 50 MHz que es la principal fuente de reloj, además de la circuitería. La Figura 3.3 presenta a la tarjeta ASSERTA y la Figura 3.4 muestra el diagrama a bloques de los diferentes dispositivos contenidos en ASSERTA [19].

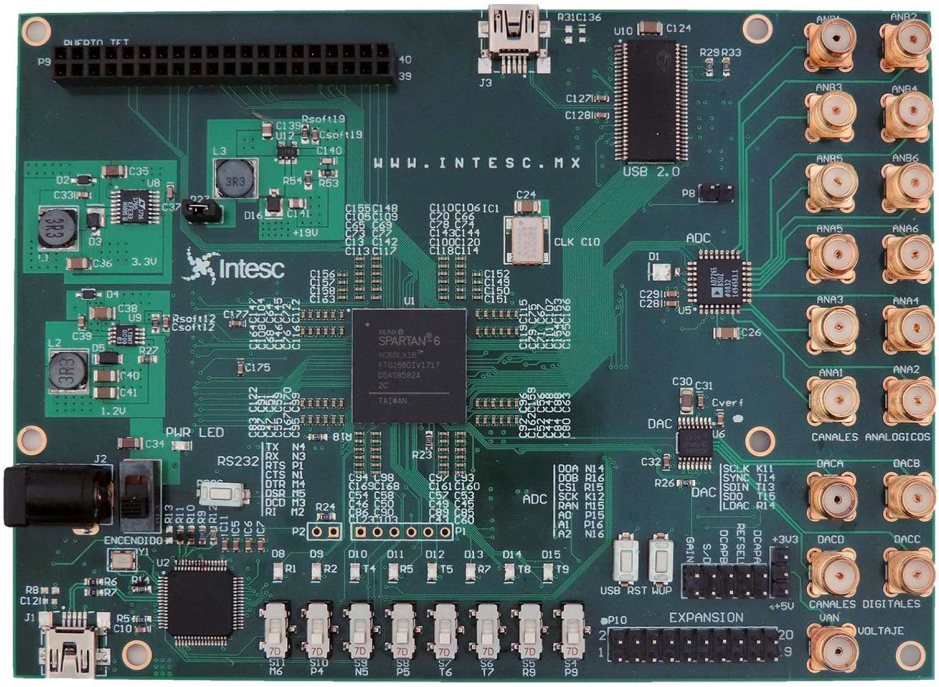


Figura 3.3 Tarjeta de desarrollo FPGA ASSERTA [19].

3.2.1 Características de la tarjeta ASSERTA

- FPGA Familia Spartan 6 modelo XC6SLX16 Empaquetado 2FTG256
- 576 kb de Block RAM
- Oscilador de 50 MHz
- Convertidor USB-RS232 (FTDI FT2232HL)
- Memoria Flash 64 Mb
- 16 pines de entrada/salida digital (FPGA)
- Puerto para TFT de 4,3”
- ADC modelo AD7265, contiene 2 ADC de 6 canales cada uno
- DAC modelo AD5684, contiene 1 DAC de 4 canales
- 9 LEDs
- 8 Switches
- 3 push button
- Interruptor de encendido o apagado
- Fuente Principal: 5V desde alimentación externa
- Un Transceiver USB 2.0 de Cypress CYC68013A
- 17 Conectores SMA: 6 para ADC, 4 para DAc y 1 para voltaje de salida con selector de 5V o 3.3V
- 1 jumper para habilitar o deshabilitar fuente de 19V
- 1 jumper para programar Transceiver desde memoria EEPROM
- 5 jumpers para configurar ADC y DAC
- 1 puerto de 3 pines para seleccionar voltaje en VAN utilizando un jumper [19]

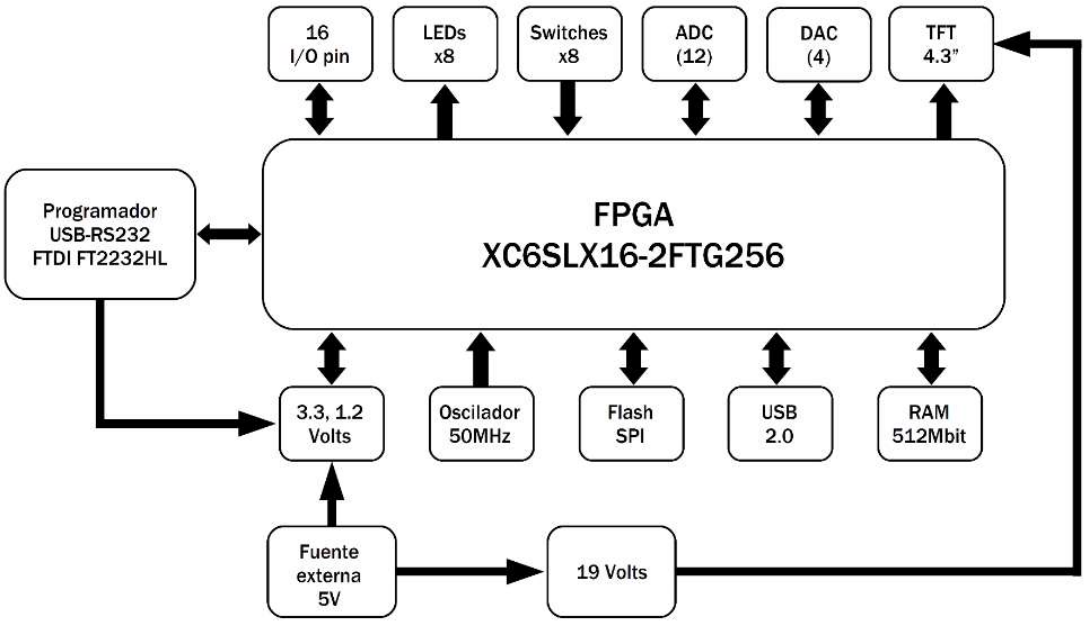


Figura 3.4 Diagrama a bloques de la Tarjeta de desarrollo ASSERTA [19].

3.3 Módulo de la cámara VGA OV7670

El módulo de visión posee un sensor de imagen CMOS VGA OV7670, fabricado por Omnivisión, capaz de trabajar a un máximo de 30 fps (cuadros por segundo) a una resolución de 640x480 pixeles (0.3 MPx). Es un SoC (sistema en chip) por lo que es capaz de realizar procesamiento de imágenes, como: control de exposición gamma, balance de blancos, saturación de color, control de tono. Estos parámetros son configurables mediante la interface SCCB (Bus de Control de Cámara Serial) compatible con comunicación I2C. El sensor incluye filtros propios de eliminación de ruido eléctrico, fixed pattern noise (FPN), smearing, blooming, etc. Lo que produce una imagen más estable y limpia [20].

Figura 3.5 Módulo OV7670 [21].

Cuenta con un buffer FIFO AL422B de 380 KB de memoria, que permite trabajar de manera más sencilla, pues los datos se almacenan en el buffer temporalmente y así disminuyen la carga computacional del controlador, el diagrama de bloques funcional del módulo se puede observar en la Figura 3.6 [20].

Sus características son:

- Voltaje de Operación: 3.3V DC.
- Consumo de energía: 60mW/15fpsVGAYUV.
- Corriente Sleep: <20uA.
- Cámara: OV7670.
- Buffer: AL422B.

- Transmisión de datos en paralelo (8 bits).
- Interface de control estándar SCCB, compatible con I2C.
- Necesita una frecuencia de 8MHz.
- Lente óptico de 1/6”.
- Ángulo de visión (FOV): 25°.
- Resolución: 640x480 VGA.
- Sensibilidad: 1.3V / (Lux-sec).
- Ratio Señal-Ruido (SNR); 46 dB.
- Rango Dinámico: 52 dB.
- Modo de vista: Progresivo.
- Lente de alta calidad F1.8/6 mm.
- Formatos de Salida: Raw RGB (8 digit). RGB (GBR 4:2:2, RGB 565/555/444), YUV (4:2:2) y YCbCr (4:2:2).
- Máximo refresco de cuadro: 30 fps VGA.
- Exposición electrónica: 1 a 510 filas.
- Covertura de pixel: 3.6 um x 3.6 um [20].

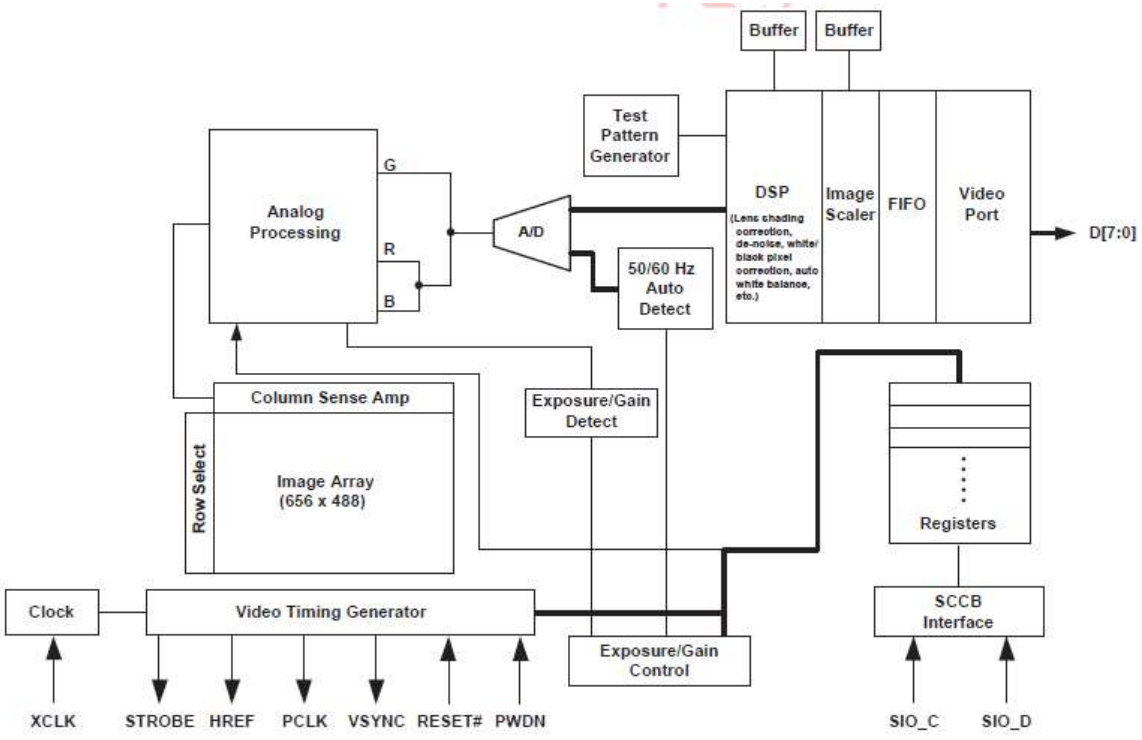


Figura 3.6 Diagrama de bloques funcional [22].

3.3.1 Configuración de los pines del módulo

El módulo cuenta con 22 pines, como se muestra en la Figura 3.7, los cuales incluyen los de la cámara OV7670 y el FIFO, ambos alimentados con una tensión de 3.3V. Los pines más importantes se mencionan a continuación:

- SIO_C: Señal del reloj de la comunicación SCCB.
- SIO_D: Señal de datos de la comunicación SCCB.
- VSYNC: Señal de sincronización vertical (indica el final de salida de información del *"frame"* anterior y el comienzo del siguiente).
- HREF: Señal de sincronización horizontal (indica el final de salida de información de la línea anterior y el comienzo de la siguiente).
- XCLK: Señal de reloj de entrada (indica una base de tiempos al módulo).
- PCLK: Señal de reloj de salida del pixel (indica el comienzo de salida de la información del siguiente pixel).
- D [7-0]: 8 bits de datos (representan la máxima cantidad de información que se extrae en cada instante, 1 Byte).
- Reset: Señal de reset (restablece todos los registros a sus valores por defecto y reinicia el módulo) [21].

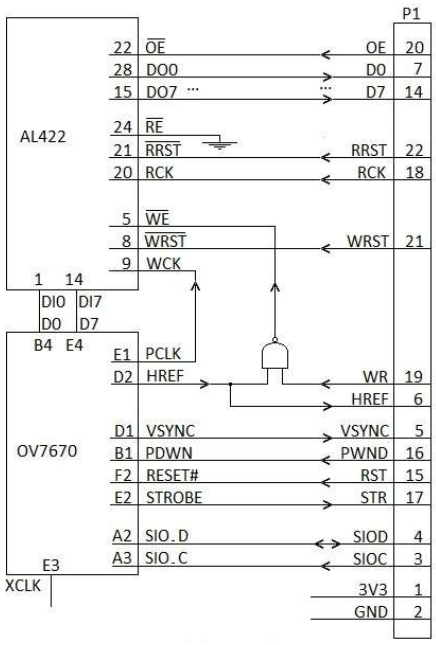


Figura 3.7 Pines del módulo OV7670.

3.3.2 Funcionamiento y configuración del módulo

El diagrama de tiempos que se presenta en la Figura 3.8 presenta el funcionamiento de las salidas digitales más importantes del módulo de la cámara. Se puede observar que la señal de VSYNC indica el comienzo de una imagen (frame) cuando se pone a estado Alto (1), éste dura un corto tiempo para después volver a su estado inicial Bajo (0) en espera de indicar el comienzo de la siguiente imagen. Cuando HREF se pone a 1 inicia la transferencia de información de una fila, esto continua hasta que cambia al estado 0, lo que indica que ya finalizó la transferencia de información de toda la fila. Éste proceso se repite hasta terminar todas las filas, por ejemplo, en la resolución de 640x480 dicho proceso se repetirá 480 veces para a completar 1 frame. Mientras HREF permanezca en 0, toda la información que sale por los pines D[7-0] no es válida.

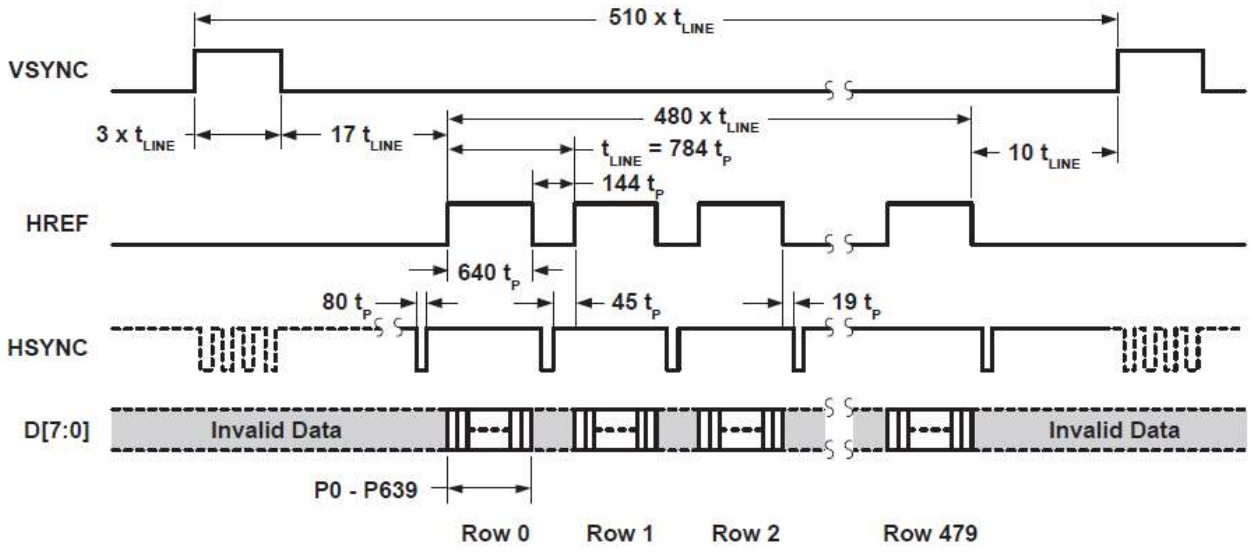


Figura 3.8 Diagrama de temporización de los Frame VGA [22].

Como se observa en la Figura 3.9, los cambios se producen por flanco de bajada de la señal de reloj de salida (PCLK), por cada ciclo de dicho reloj se obtiene 1 Byte de información, por lo tanto, por cada pixel se necesitan 2 Bytes.

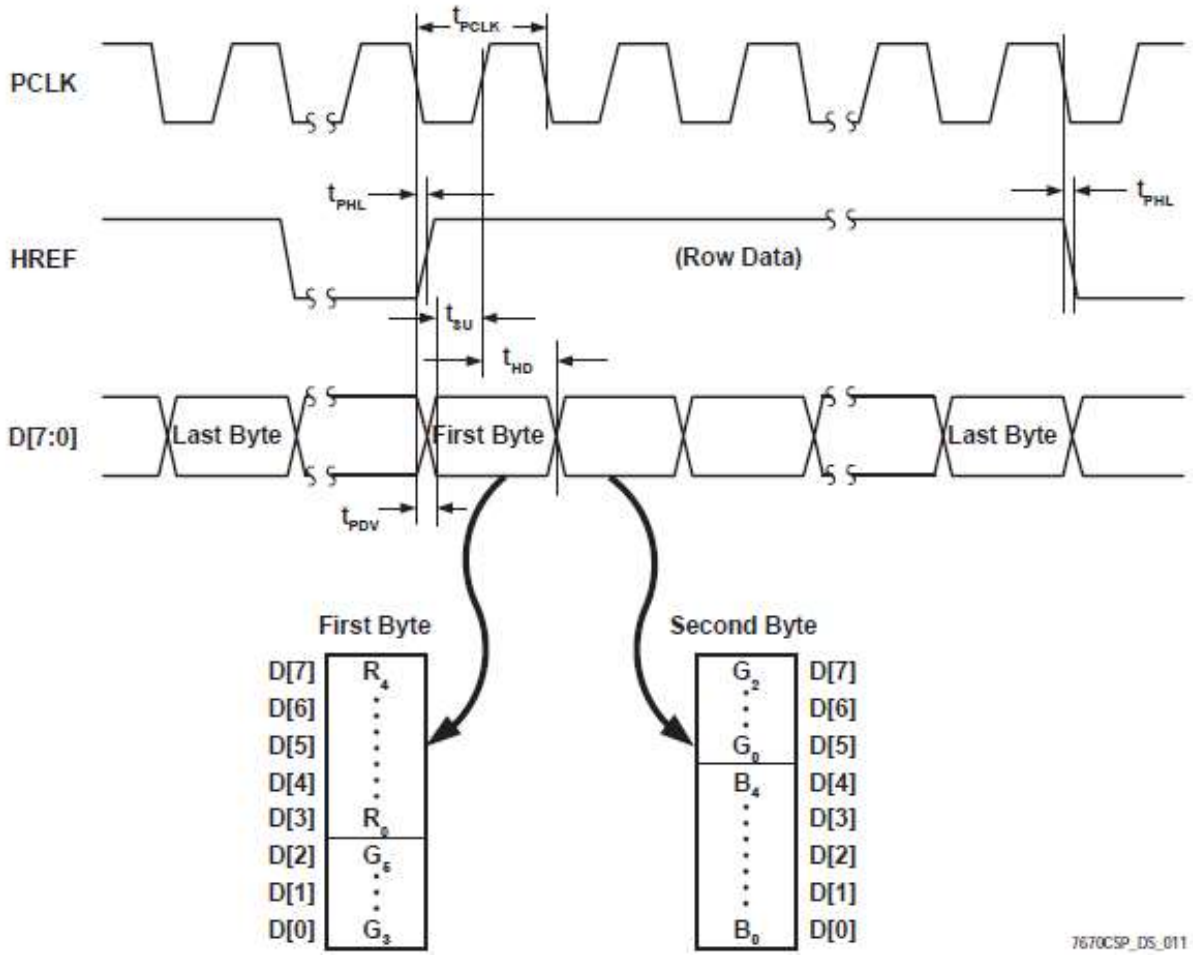


Figura 3.9 Diagrama de temporización del formato de salida RGB 565 [22].

Para configurar los distintos modos de operación de la cámara se utilizan los registros, que son dispositivos digitales donde se almacena información de manera temporal. Los registros están constituidos por flip-flops o biestables, cada uno de los cuales almacena un bit con el valor 0 o 1. Lo más común es que los registros estén constituidos por 8 biestables, es decir, almacenan 8 bits, este es el caso de los registros de la cámara OV7670. Los registros de la cámara nos permiten configurar opciones como la frecuencia de reloj de salida, el formato de colores de los píxeles, la resolución de la imagen y el modo en que actúan las distintas señales [21].

Cabe mencionar que todos los registros tienen un valor por defecto, de forma que la cámara ya viene configurada para funcionar de un determinado modo al encenderla. La configuración del resto de registros se puede ver en el datasheet del fabricante.

3.4 Programación en VHDL

Para el desarrollo de la programación en VHDL para la tarjeta ASSERTA y el módulo OV7670 se hace uso del esquema mostrado en la Figura 3.10, en el cual se puede observar la interacción de los diferentes dispositivos en el proceso.

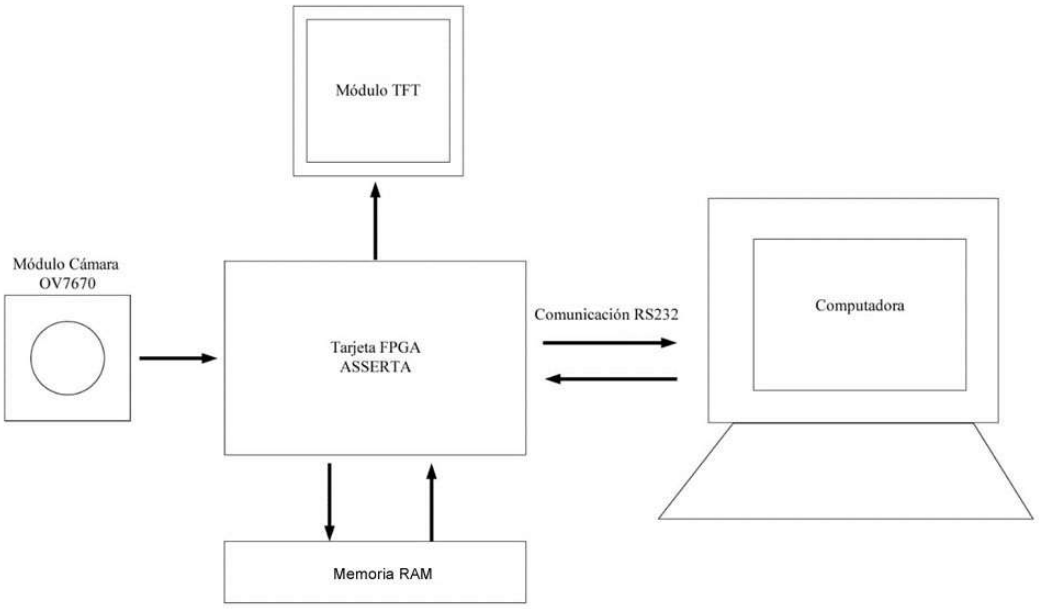


Figura 3.10 Diagrama general del Proyecto.

El diagrama de la Figura 3.10 representa en términos generales que la tarjeta debe procesar los datos que recibe del módulo OV7670, cada frame recibido será guardado en la memoria RAM después de ser binarizado por un corto tiempo para después ser mostrado en la pantalla TFT, también los frames recibidos podrán ser enviados a la computadora por medio de la comunicación RS232 si ésta lo solicita. La programación de la tarjeta se hará por medio de IP Cores, es decir, módulos VHDL previamente creados, los cuales son el controlador de la cámara OV7670, Memoria RAM, comunicación RS232, el controlador de la Pantalla TFT y por último el TOP que es el que une y conecta todos los submódulos antes mencionados. INTESC al ser una empresa de diseño y manufactura de tarjetas de desarrollo cuenta con los Cores necesarios para hacer uso de la memoria RAM, la pantalla TFT, la comunicación RS23 y el del módulo OV7670 por lo que sólo el desarrolla en VHDL es el TOP o módulo principal que cumple con algunos de los objetivos específicos planteados al comienzo del proyecto.

3.4.1 Programación del TOP

El código programado en VHDL se encargará de realizar un bloque o módulo principal conocido como TOP nombrado CAPTURA_PIXEL, tal como se muestra en la Figura 3.11, que englobará el funcionamiento de 4 submódulos o IP cores que son: los controladores de la memoria RAM (rams_16b), la TFT (CONTROL_TFT), la comunicación RS232 (RS232) y el módulo OV7670 (ov7670_controller) mostrados en la Figura 3.12. En el TOP se desarrollan 3 procesos en paralelo que se encargarán de realizar la interconexión de todos los submódulos para obtener y procesar los datos enviados por el módulo OV7670, enviar las imágenes a la TFT y para la comunicación RS232 entre la computadora y la tarjeta.

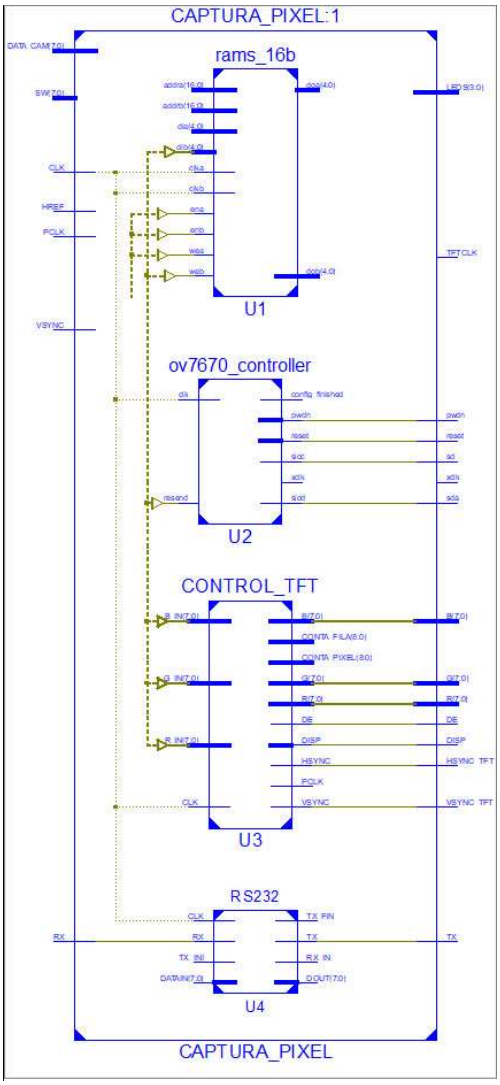


Figura 3.11 Diagrama a bloques del modulo principal (TOP).

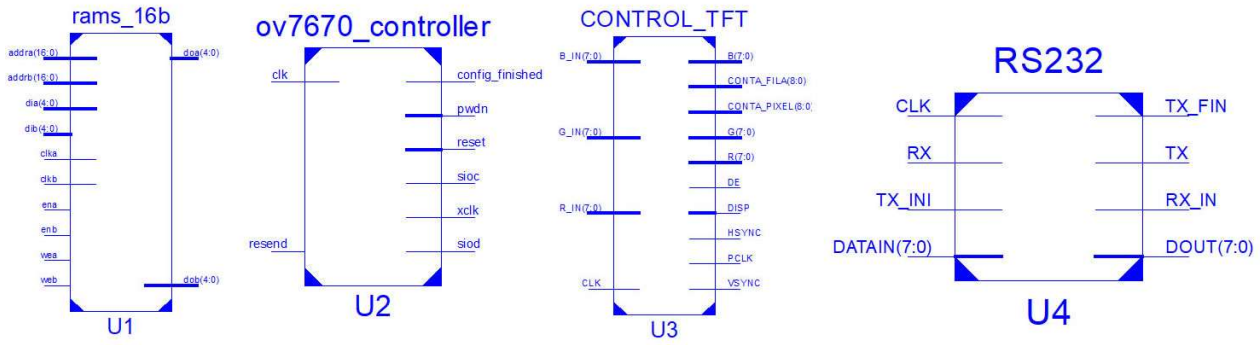


Figura 3.12 Submódulos implementados en el módulo principal (TOP).

El primer proceso del TOP es para la recepción del vídeo con la cámara y el procesamiento del vídeo, en éste caso es de binarización. Se realiza con una máquina de 6 estados, mostrada en la Figura 3.13. Cada estado realiza lo siguiente:

- Estado E0: Espera la confirmación para saber que la configuración del módulo de la cámara es correcta. Si lo es, el flag 'Config' cambia a 1 y se pasa al siguiente estado, de lo contrario se vuelve verificar la confirmación.
- Estado E1: Indica cuando está disponible un nuevo cuadro (frame) y se hace por medio del flag 'VSYNC' que se pone a 1 si lo hay y se pasa al siguiente estado.
- Estado E2: Captura el primer byte y lo guarda en un vector, siempre y cuando se active el flag 'HREF', es decir este a 1.
- Estado E3: Si el flag 'HREF' sigue en 1 se captura el segundo byte y en un nuevo vector se guardan los 3 bits menos significativos del vector guardado en el estado anterior y los 3 bits más significativos del byte nuevo, es importante aclarar que se eligen estos 6 bits porque son los que contienen mayor información de la imagen al trabajar en el formato de salida RGB 565 de la cámara OV7670. Además, en éste mismo estado los vectores se van guardando en la memoria RAM después de realizar la binarización por medio de la comparación del umbral con el valor de cada pixel. También se lleva un control del número de filas y columnas de los frames, ya que al ir guardando pixel por pixel se debe conocer a que pixel pertenece, en otras palabras, se debe conocer su ubicación en el frame para saber cuándo se pasa a otra fila. Por cada pixel se repiten los estados 2 y 3, hasta llegar al último pixel del frame donde se pasa al siguiente estado.

- Estado E4: Verifica si ha abierto la comunicación con la computadora para transferirle información por medio del flag ‘Mandar’, si no la hay regresa al estado E1, pero si hay comunicación se pasa al último estado.
- Estado E5: Monitorea el flag ‘Fin_envio’, si está a 1 indica que ya finalizo la transferencia de la información con la computadora, y cuando sucede se pasa a E1.

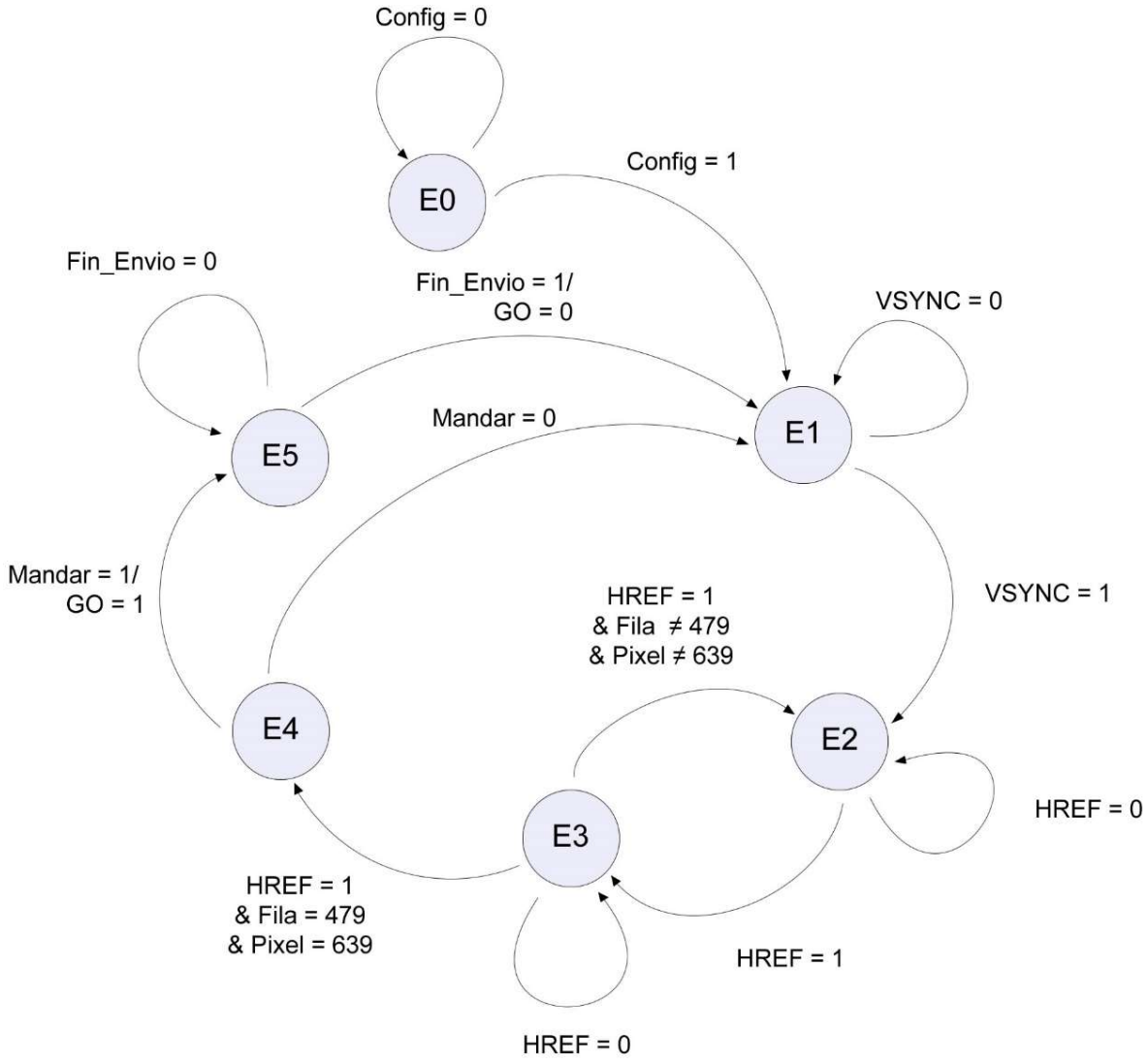


Figura 3.13 Diagrama de transición de estados de la recepción del vídeo.

El segundo proceso transfiere el video a la pantalla TFT, aquí simplemente se envían los frames binarizados contenidos en la memoria RAM a la pantalla, para ello se lee y envía pixel por pixel guardado. El tercer proceso es el de la comunicación RS232, el cual se realiza con otra máquina de estados, ver Figura 3.14. Cada estado realiza lo siguiente:

- Estado E0: Se verifica si el flag 'RX_IN' se ha puesto a 1 durante un ciclo de reloj, lo que indica que se ha recibido 1 byte, una nota importante es que al ser asíncrona la recepción puede ocurrir en cualquier momento por lo que se debe monitorear constantemente dicha señal, si se ha recibido un byte se pasa al último estado directamente y se activa el flag 'Mandar', que se utiliza para conocer si existe comunicación entre la computadora y la tarjeta.
- Estado E1: Se activa el flag 'TX_INI', lo que indica que se enviarán datos por el puerto TX, se preparan en un vector los bits de información, cabe mencionar que el envío se hace de 10 bits (1 bit de inicio, 8 bits de información y 1 bit de paro).
- Estado E2: se verifica si ha finalizado la transmisión de datos por medio del flag 'TX_FIN', es importante tener en cuenta que si éste flag se mantiene en 1 no se permitirá otra transmisión simultánea hasta que 'TX_FIN' sea 0, si 'TX_FIN está a 1 el flag 'TX_INI' cambia a 0 y se pasa al siguiente estado, además se realiza un conteo para conocer si toda la información almacenada en la memoria RAM se ha enviado, si es así el flag 'Fin_Envio' cambia a 1.
- Estado E3: Verifica el conteo de la información enviada, y si no se han enviado toda la información se regresa al estado E1, pero si se enviaron todos los datos almacenados se regresa al estado E0.
- Estado E4: Se da la indicación de realizar el envío de la información por medio del flag GO si está a 1, si es así se pasa al estado E1.

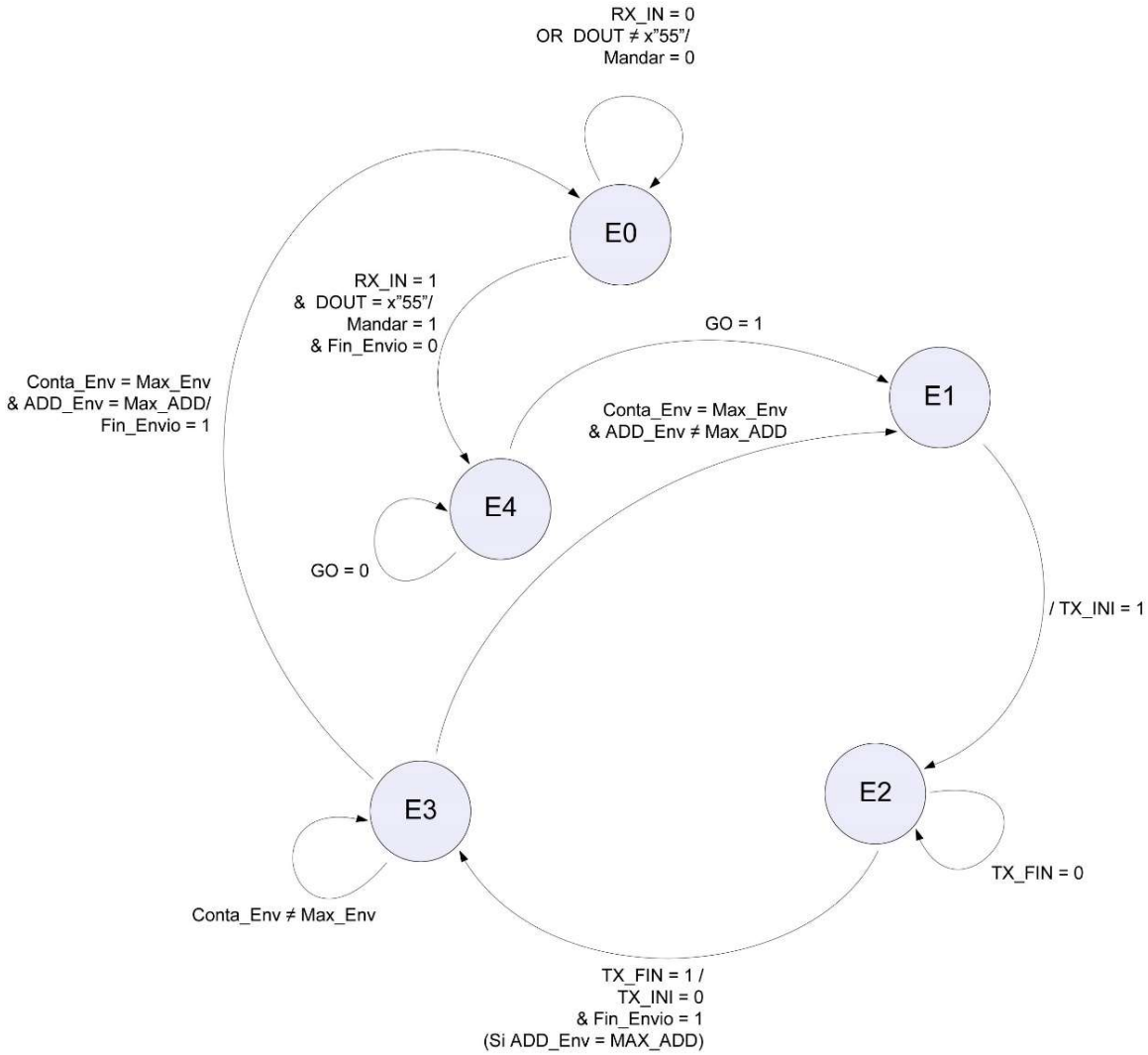


Figura 3.14 Diagrama de transición de estados de la comunicación.

3.4.2 Binarización

Las imágenes digitales están compuestas por un amplio rango de valores de intensidad, algunas denominadas imágenes de nivel de gris, donde los niveles de gris se representan por 256 valores (8 bits por píxel). Cabe mencionar que no todas las aplicaciones precisan de tantos niveles de gris, a veces es necesario utilizar pocos niveles cuando se trabajan con imágenes de muy alto contraste, por ejemplo, en aplicaciones industriales se trabaja únicamente con dos niveles de gris, es decir con imágenes binarias [23].

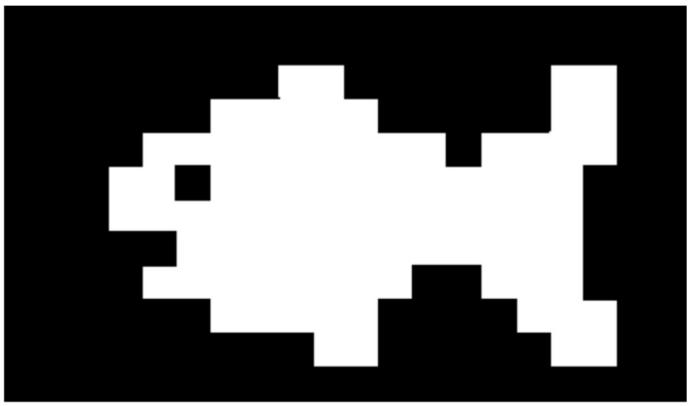


Figura 3.15 Imagen binarizada, donde el pixel en blanco tiene el valor a 1 y el negro valor a 0 [23].

Se ha decidido trabajar con imágenes binarias ya que se reduce al mínimo los datos necesarios para representar las imágenes, aprovechando al máximo los recursos con los que cuenta el FPGA. Además de que las propiedades geométricas y topológicas de los objetos presentes en las imágenes se obtienen de forma más rápida y fácil. Los recursos de la tarjeta ASSERTA son limitados y desde el punto de vista computacional, las imágenes binarias se procesarían mucho más rápidamente, de una forma simple y robusta.

Para obtener las imágenes binarias siempre se debe partir de imágenes de niveles de gris, pero en la actualidad no existen cámaras comerciales que proporcionen imágenes binarias, para ello se necesita un proceso de conversión el cual se denomina binarización. Dicho proceso puede consistir en considerar únicamente el bit más significativo del nivel de gris de cada píxel, es decir fijar un umbral en la escala de gris que servirá para asignar en la imagen binaria un 0 si el nivel de gris del pixel es inferior, o 1 si supera este umbral ver Figura 3.15. Es fundamental que la imagen de nivel de gris tenga un alto contraste, es decir que posean niveles de gris bien diferenciados.

La etapa más importante para la binarización consiste en la elección del valor umbral, pues éste constituye la referencia para separar el fondo claro del objeto oscuro, o viceversa. Por ello, para aplicar un umbral fijo se tendrán que controlar parámetros del entorno para facilitar el procesamiento, como son el fondo y la iluminación, ya que cualquier variación en la iluminación pueden ocasionar cambios en los niveles de gris de la imagen que invaliden el umbral fijado [23].

El proceso para la binarización en VHDL se realiza en E3 de la máquina de estados de la Figura 3.13. El video se adquiere con los vectores del color verde para ahorrar los recursos de la tarjeta, dicho proceso compara cada valor del pixel guardado con un umbral, donde si es mayor al umbral se le da un valor de 0 y si es menor el valor de 1, cabe mencionar que se debe realizar una conversión a vector del umbral que es una señal de tipo entero, ya que dicha comparación se hace entre vectores de 8 bits.

CAPÍTULO 4. PROGRAMACIÓN EN MATLAB

En el presente capítulo se da una breve reseña de la extracción de las características de una imagen binaria y se presenta el desarrollo de la programación en Matlab para determinar el área de los objetos captados por la cámara OV7670, además de los momentos de inercia para calcular los grados de desfase que tienen.

4.1 Extracción de Características de imágenes binarias

En las aplicaciones industriales que utilizan visión artificial siempre se tiene un conocimiento del objeto u objetos que van a aparecer ante la cámara. Saber cómo son permite establecer propiedades que sean especialmente discriminantes entre estos objetos. Entre algunas de las características se tiene el área que ocupa en la imagen, el número de agujeros, momentos de inercia, etc. Cuantificando todas estas propiedades se podrá llegar a diferenciar unos objetos de otros sin más que solo comparar valores numéricos. Esta cuantificación de las propiedades de los objetos que aparecen en las imágenes es lo que se conoce como extracción de características.

La extracción de características consiste en el cálculo de propiedades globales de las regiones que aparecen en la imagen. Por lo que se pasa de trabajar desde propiedades de píxel a un nivel de propiedades de regiones (descriptores de región) que inicialmente no es explícito en la imagen. Esto tiene una gran ventaja ya que, por un lado, las regiones serán en principio mucho menores en número y, por otro lado, tienen muchas más propiedades que un pixel. Un pixel tiene como características su posición y su nivel de gris mientras que una región, aunque sea binaria, tiene un contenido semántico mucho mayor pues se puede establecer su área, perímetro, elongación o circularidad, orientación, número de agujeros, momento de inercia, etc. El análisis posterior sobre regiones es mucho más fiable que si se lleva a nivel de píxel [23].

Para seleccionar las características adecuadas que describan una región es necesario que éstas reúnan una serie de propiedades tales como:

- Capacidad discriminante: los valores numéricos asociados a una característica deben ser significativamente distintos para poder distinguir unos objetos de otros.
- Estabilidad: la característica o características elegidas deben presentar valores numéricos similares para una misma clase de objetos. Si el objeto que debe reconocerse puede aparecer en distintas posiciones en la imagen las características elegidas deberán ser invariantes ante traslación y rotación. Si el objeto tiene distintos tamaños deberán ser invariantes a cambios de escala.
- Fácilmente evaluables: el cálculo de las características no debe requerir de una carga computacional exagerada para que puedan obtenerse dentro del tiempo disponible en la aplicación [23].

Por simplicidad y eficiencia interesa trabajar con el mínimo número de características. Si dos características están relacionadas, alguna de ellas no aportará gran cosa y lo mejor será eliminarla. Cuando el número de objetos diferentes que pueden aparecer en la imagen es reducido, como es el caso de la mayoría de las aplicaciones de inspección industrial (pieza grande o pequeña, envase con tapón o sin él, pieza con o sin agujero, artículo con mancha o sin ella, etc.), no es difícil establecer a priori una o dos características que representen de forma inequívoca cada clase de objeto. Sin embargo, en aplicaciones más complejas con gran número de patrones, por ejemplo, de reconocimiento de caracteres, elegir las características más adecuadas para diseñar el clasificador nos es una tarea tan sencilla [23].

Algunas características más utilizadas para reconocer y localizar objetos en imágenes son:

- Tamaño
- Posición
- Orientación
- Perímetro y circularidad
- Polígonos envolventes
- Número de agujeros y número de Euler.

4.1.1 Tamaño

Suponiendo píxeles cuadrados, que actualmente son estándar, el área de una región vendrá dada por el número de píxeles que la constituyen. Así, el área se obtiene fácilmente como:

$$Área = \sum_{x=1}^{f} \sum_{y=1}^{c} B(x, y) \tag{4.1}$$

			X	1	1	1	X			=	5
		X	1	1	1	1	X			=	6
	X	1	1	1	1	1	1	X		=	8
X	1	1	1	1	1	1	1	1	X	=	10
X	1	1	1	1	1	1	1	1	X	=	10
X	1	1	1	1	1	1	1	1	X	=	10
X	1	1	1	1	1	1	1	1	X	=	10
	X	1	1	1	1	1	1	1	X	=	9
	X	1	1	1	1	1	1	X		=	8
		X	1	1	1	1	X			=	6
			X	1	1	1	X			=	5

Figura 4.1 Determinación del área de una imagen [24].

A este parámetro también se le denomina momento de orden cero. El tamaño de un objeto en la imagen es invariante ante la rotación y traslación, es decir, no depende de la posición en la que aparezca el objeto en la imagen [23].

4.1.2 Posición

La posición de un objeto en la imagen normalmente se calcula determinando la posición del centro de gravedad de los píxeles que integran el objeto, en los que estos se consideran de masa unitaria. El centro de gravedad es poco sensible al ruido y resulta muy útil en la aprehensión de objetos con manipuladores como el engrane mostrado en la Figura 4.1. Para calcularlo se debe conocer los momentos de inercia de primer orden los cuales son [23]:

$$I_x = \sum_{x=1}^{f} \sum_{y=1}^{c} x \cdot B(x,y) \tag{4.2}$$

$$I_y = \sum_{x=1}^{f} \sum_{y=1}^{c} y \cdot B(x,y) \tag{4.2}$$

Insistiendo en que suponemos imágenes con una única región, el centro de gravedad (x_g, y_g) de la región se obtiene a partir de sus coordenadas:

$$x_g = \frac{\sum_{x=1}^{f} \sum_{y=1}^{c} x \cdot B(x,y)}{Área} \tag{4.3}$$

$$y_g = \frac{\sum_{x=1}^{f} \sum_{y=1}^{c} y \cdot B(x,y)}{Área}$$

Figura 4.2 Centro de gravedad de engrane [23].

4.1.3 Orientación

La orientación de un objeto puede determinarse mediante el cálculo del eje respecto al cual el momento de inercia es mínimo. Utilizando la definición que se hace en mecánica, el eje de inercia mínimo de una región será la recta $Ax + By + C = 0$ tal que la suma de las distancias al cuadrado entre los píxeles del objeto y dicha línea es mínima [25]. Se debe tener en cuenta que el vector $(A, B) = (cos\theta, sen\theta)$ es un vector perpendicular a la

recta, siendo θ el ángulo que forma la perpendicular desde el origen a la recta con el eje x como se ve en la Figura 4.2. A continuación, se describe la obtención de la ecuación que gobierna la recta [23].

Dado que el centro de gravedad (x_g, y_g) es un punto que pertenece a los ejes principales de inercia, debe cumplirse:

$$Ax_g + By_g + C = 0 \tag{4.4}$$

Y sustituyendo el valor de C:

$$A(x - x_g) + B(y - y_g) = 0 \tag{4.5}$$

El problema se puede formular de forma sencilla trasladando la región al origen según (x_g, y_g) con lo que el eje principal de inercia en las nuevas coordenadas $x' = (x - x_g)$ e $y' = (y - y_g)$ pasará por el origen y tendrá por ecuación:

$$Ax' + By' = 0 \tag{4.6}$$

Dado que el vector $(A, B) = (Cos\theta . Sen\theta)$ cumple $A^2 + B^2 = 1$, la distancia al cuadrado ${d_i}^2$ de cualquier punto (x_i', y_i') del objeto a la recta vendrá dada por:

$${d_i}^2 = (Ax_i' + By_i')^2 \tag{4.7}$$

Por lo que la suma de todas las distancias al cuadrado $\sum_{i=1}^{N} {d_i}^2$ vendrá dada por:

$$\sum_{i=1}^{N} (Ax_i' + By_i')^2 = A^2 \sum_{i=1}^{N} {x_i'}^2 + B^2 \sum_{i=1}^{N} {y_i'}^2 + 2AB \sum_{i=1}^{N} x_i' y_i' \tag{4.8}$$

donde N es el número de píxeles de la región, es decir el área.

Es fácil identificar en las sumatorias anteriores, deshaciendo la traslación previamente efectuada al origen, los tres momentos de segundo orden de una región $I_{xx}, I_{yy}\ e\ I_{xy}$:

$$I_{xx} = \sum_{x=1}^{f}\sum_{j=1}^{c}(x - x_g)^2 \cdot B(x,y) = \sum_{i=1}^{N}(x_i - x_g)^2$$

$$I_{xy} = \sum_{x=1}^{f}\sum_{j=1}^{c}(x - x_g) \cdot (y - y_g) \cdot B(x,y) = \sum_{i=1}^{N}(x_i - x_g) \cdot (y_i - y_g) \qquad (4.9)$$

$$I_{yy} = \sum_{x=1}^{f}\sum_{j=1}^{c}(y - x_g)^2 \cdot B(x,y) = \sum_{i=1}^{N}(y_i - x_g)^2$$

Por tanto, sustituyendo (A, B) por $(cos\theta, sen\theta)$, se reescribe la ecuación (4.8) como:

$$\sum_{i=1}^{N} {d_i}^2 = cos^2\theta \cdot I_{xx} + sen^2\theta \cdot I_{yy} + 2cos\theta sen\theta \cdot I_{xy} \qquad (4.10)$$

Como se desea encontrar la recta que minimiza la sumatoria de las distancias al cuadrado, se deriva la ecuación (4.10) respecto al parámetro θ y se iguala a 0:

$$\frac{d}{d\theta}\sum_{i=1}^{N} {d_i}^2 = 0 \qquad (4.11)$$

$$-2cos\theta sen\theta \cdot I_{xx} + 2cos\theta sen\theta \cdot I_{yy} + 2(cos^2\theta - sen^2\theta) \cdot I_{xy} = 0$$

Se sabe que por trigonometría:

$$(cos^2\theta - sen^2\theta) = cos(2\theta)$$

$$2cos\theta sen\theta = sen(2\theta)$$

Entonces reformulando la ecuación (4.11):

$$sen(2\theta) \cdot \left(I_{yy} - I_{xx}\right) + 2cos(2\theta) \cdot I_{xy} = 0$$

Despejando se tiene:

$$\tan(2\theta) = \frac{2I_{xy}}{I_{xx} - I_{yy}} \tag{4.12}$$

Obviamente, las soluciones $\theta_1 = \theta$ y $\theta_2 = \theta_1 + \frac{\pi}{2}$ son ambas solución al problema, por lo que no se puede garantizar a cuál de los dos ejes principales de inercia corresponde el ángulo [24].

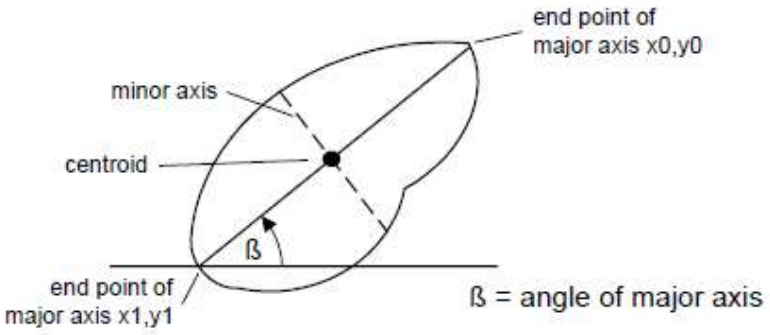


Figura 4.3 Ejes máximo y mínimo de un objeto [24].

Para deshacer la indeterminación simplemente basta con sustituir θ_k, $k = 1\ o\ 2$ en la ecuación (4.9) y elegir el ángulo optimo θ_o que nos dé el menor valor:

$$\theta_o = argmin_{k=1,2}(cos^2\theta_k I_{xx} + sen^2\theta_k I_{yy} + 2cos\theta_k sen\theta_k I_{xy}) \tag{4.13}$$

En definitiva, el eje de inercia buscado es:

$$cos\theta_o x + sen\theta_o y - \left(cos\theta_o x_g + sen\theta_o y_g\right) = 0 \tag{4.14}$$

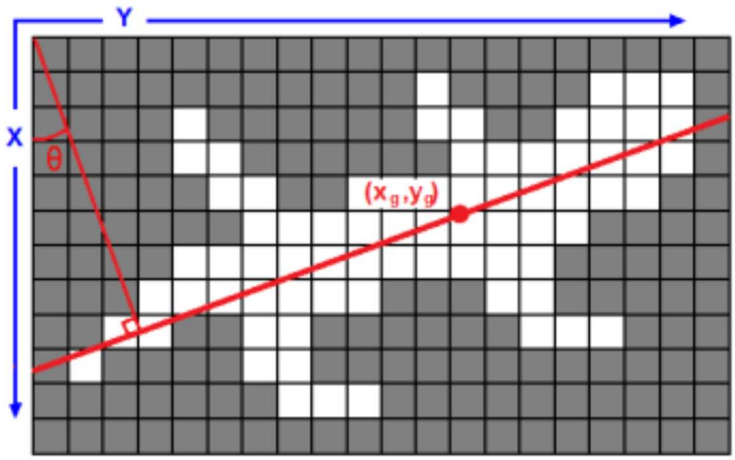


Figura 4.4 Orientación de una imagen [23].

4.1.4 Perímetro y circularidad

La longitud del perímetro se obtiene simplemente sumando 1 cada vez que la secuencia de píxeles del contorno avanza en vertical u horizontal y sumando √2 cuando la secuencia avanza en diagonal [23].

A partir del contorno no sólo se dispone del perímetro del objeto, sino que también podemos cuantificar la forma que presenta el objeto de la imagen combinando el valor del perímetro con el del área. Esta nueva característica nos da una idea de su compacidad o circularidad (ver Figura 4.3). Supongamos que nuestro objeto tiene un área A y un perímetro P. El radio r correspondiente a una circunferencia con ese perímetro $P = 2\pi r$ será, por tanto, $r = P/2\pi$. Entonces, el área que presenta dicha circunferencia de perímetro P vendrá dado por:

$$A = \pi r^2 = \pi(\frac{P}{2\pi})^2 = \frac{P^2}{4\pi} \qquad (4.15)$$

Para cuantificar la circularidad de un objeto compararemos el área de la circunferencia A_c con el área de dicho objeto:

$$circularidad = \frac{A}{A_c} = \frac{4\pi A}{P^2} \qquad (4.16)$$

El valor de la circularidad oscila entre 0 y 1. Sobre objetos redondeados toma valores próximos a 1 y para objetos irregulares o alargados la circularidad tiende a cero [23].

Obviamente, si los objetos presentan agujeros, el área a considerar en la fórmula anterior no será el del objeto real sino el del objeto sumando el área de sus agujeros.

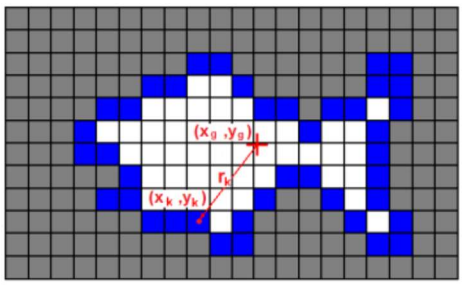


Figura 4.5 Circularidad de una imagen [23].

4.1.5 Envolvente convexa y MBR

La envolvente convexa (convex hull) de un conjunto de píxeles C es la figura poligonal convexa más pequeña que contiene al conjunto. Intuitivamente podemos imaginar la envolvente convexa de la siguiente forma, suponiendo que sobre cada píxel del conjunto se clava una punta, la envolvente convexa será la forma que adopta una goma elástica que encierra al conjunto de puntas y que corresponde a un polígono convexo cuyos vértices son algunos de los píxeles del contorno de la región [23].

El cálculo de la envolvente convexa ha sido muy estudiado en gráficos por ordenador y existen variados algoritmos para su cálculo.

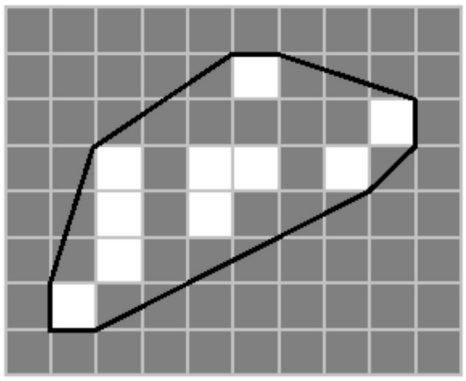


Figura 4.6 Envolvente convexa [23].

Otra característica es el mínimo rectángulo contenedor MBR (mínimum bouding rectangle), cuyos ejes son paralelos a los de la imagen. El MBR tendrá como coordenadas de su esquina superior izquierda a $(\min(x), \min(y))$ y de su esquina inferior derecha a $(\max(x), \max(y))$, con $(x, y) \in C$. (ver Figura 4.5)

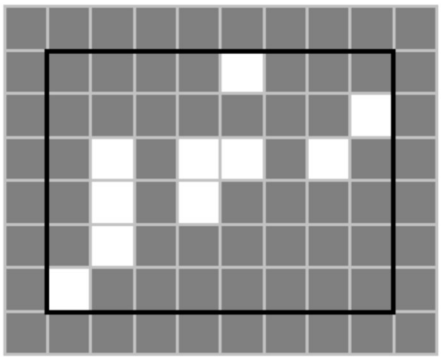


Figura 4.7 Rectángulo envolvente de tamaño mínimo [23].

4.1.6 Número de agujeros y número de Euler

Si los objetos que pueden aparecer en la imagen presentan distinto número de agujeros, resulta muy eficaz utilizar este parámetro para discriminar entre unos y otros.

Para contabilizar el número de agujeros lo más sencillo es invertir la imagen y etiquetar a continuación esa imagen inversa. En la inversión, los agujeros que estaban en negro se transformarán a blanco junto con el fondo de la imagen. Luego, en el etiquetado, se asignarán etiquetas a todos los objetos que aparecen la imagen inversa. Los nuevos objetos serán ahora los agujeros. Además, se etiquetará también el fondo de la escena como un objeto al estar en blanco. Por tanto, se tiene que el número de agujeros del objeto será inferior en una unidad al número de etiquetas obtenidas [23]:

$$numAgujeros = numEtiquetasImgInversa - 1 \qquad (4.17)$$

Si se tienen imágenes con varios objetos es importante indicar que para contabilizar el número de agujeros de un objeto la imagen a invertir debe contener solo ese objeto. Si no, se contabilizarán el número total de agujeros del total de objetos que hay en la imagen, no solo los del objeto de interés.

En determinadas aplicaciones un mismo objeto puede estar formado por dos o más regiones no conexas. Por ejemplo, el reconocimiento de caracteres la ñ, la j, la i,… son objetos formados por dos componentes no conexas. En estas aplicaciones, resulta útil calcular el valor que toma el número de Euler. El número de Euler se define como el número de componentes menos el número de agujeros [23].

$$numEuler = numComponentes - numAgujeros \qquad (4.18)$$

El número de agujeros y el número de Euler es una característica invariante a la traslación, rotación y escala.

4.2 Extracción de características con Matlab

En esta etapa del proyecto, se desarrolla todo el algoritmo empleando el entorno de Matlab que se encarga de calcular los ángulos de rotación del componente captado por la cámara utilizando algunas técnicas para el tratamiento de imágenes mencionadas anteriormente. Como es bien conocido, Matlab cuenta con comandos especiales para el tratamiento de imágenes y para la extracción de características, teniendo en cuenta que el código debe ser traducido a lenguaje VHDL para el uso en FPGAs, se omite el uso de dichos comandos, por lo que la programación se realiza a un nivel más simple.

4.2.1 Comando "regionprops"

En Matlab éste comando se usa para medir las propiedades de las regiones de imágenes y su sintaxis es:

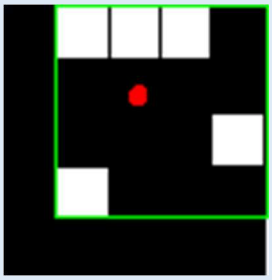
```
stats = regionprops(BW,properties)
```

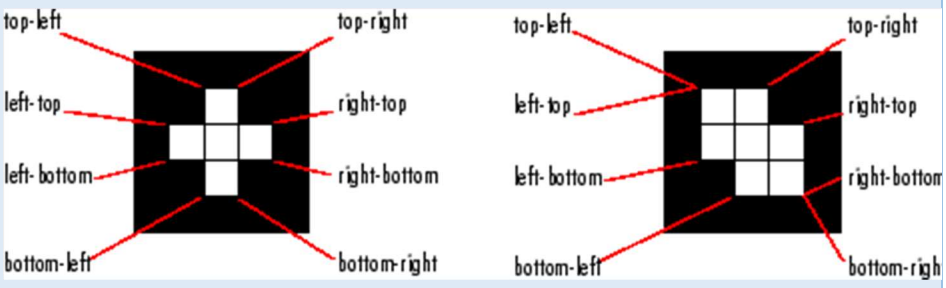
Dicho comando devuelve mediciones para el conjunto de propiedades especificadas por properties para cada componente conectado en la imagen binaria BW.

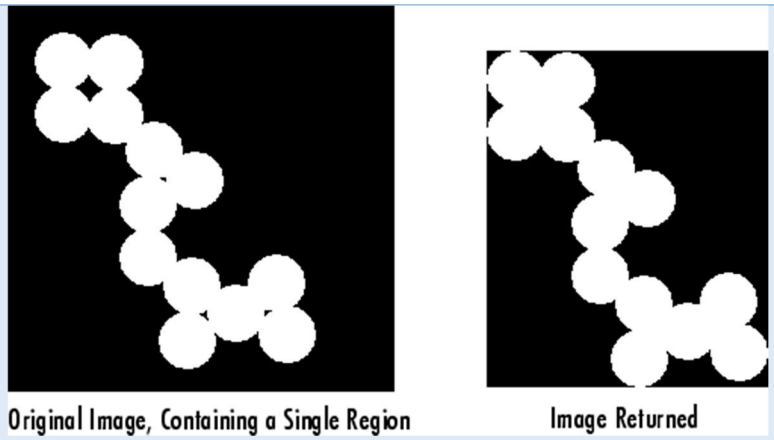
En la tabla se muestran todas las propiedades que el comando regionprops puede medir [26].

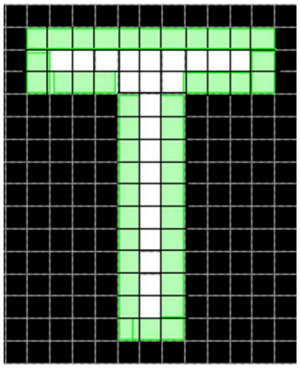
Tabla 4.1 Propiedades medidas con el comando "regionprops"

Nombre de la propiedad	Descripción
'Area'	Devuelve un escalar que especifica el número real de píxeles en la región.
'BoundingBox'	Devuelve el rectángulo más pequeño que contiene la región.
'Centroid'	Devuelve un vector de 1 por Q que especifica el centro de masa de la región. El primer elemento de Centroid es la coordenada horizontal (o coordenada x) del centro de masa, y el segundo elemento es la coordenada vertical (o coordenada y). Todos los

	demás elementos de Centroid están en orden de dimensión. Esta figura ilustra el centroide y el cuadro delimitador para una región no contigua. La región consiste en los píxeles blancos; el cuadro verde es el cuadro delimitador y el punto rojo es el centroide. 
'ConvexArea'	Devuelve un escalar que especifica el número de píxeles en 'ConvexImage'.
'ConvexHull'	Devuelve una matriz p-por-2 que especifica el polígono convexo más pequeño que puede contener la región. Cada fila de la matriz contiene las coordenadas x e y de un vértice del polígono.
'ConvexImage'	Devuelve una imagen binaria (lógica) que especifica el casco convexo, con todos los píxeles dentro del casco rellenado (configurado en activado). La imagen es el tamaño del cuadro delimitador de la región. (Para los píxeles por los que pasa el límite del casco, regionprops usa la misma lógica que roipoly para determinar si el píxel está dentro o fuera del casco).
'Eccentricity'	Devuelve un escalar que especifica la excentricidad de la elipse que tiene los mismos segundos momentos que la región. La excentricidad es la relación de la distancia entre los focos de la elipse y la longitud de su eje mayor. El valor está entre 0 y 1. (0 y 1 son casos degenerados. Una elipse cuya excentricidad es 0 es en realidad un círculo, mientras que una elipse cuya excentricidad es 1 es un segmento de línea).
'EquivDiameter'	Devuelve un escalar que especifica el diámetro de un círculo con la misma área que la región. Calculado como sqrt (4 * Area / pi).

'EulerNumber'	NúDevuelve un escalar que especifica el número de objetos en la región menos el número de agujeros en esos objetos.
'Extent'	Devuelve un escalar que especifica la proporción de píxeles en la región a píxeles en el cuadro delimitador total. Calculado como el área dividida por el área del cuadro delimitador.
'Extrema'	Devuelve una matriz de 8 por 2 que especifica los puntos extremos en la región. Cada fila de la matriz contiene las coordenadas x e y de uno de los puntos. El formato del vector es [arriba-izquierda arriba-derecha-arriba-derecha-abajo-derecha-abajo-izquierda-izquierda-abajo-arriba]. Esta figura ilustra los extremos de dos regiones diferentes. En la región de la izquierda, cada punto extremo es distinto. En la región de la derecha, ciertos puntos extremos (por ejemplo, arriba a la izquierda y arriba a la izquierda) son idénticos. 
'FilledArea'	Devuelve un escalar que especifica el número de píxeles en FilledImage.
'FilledImage'	Devuelve una imagen binaria (lógica) del mismo tamaño que el cuadro delimitador de la región. Los píxeles de encendido corresponden a la región, con todos los orificios rellenados, como se muestra en esta figura.

	
'Image'	Devuelve una imagen binaria (lógica) del mismo tamaño que el cuadro delimitador de la región. Los píxeles activados corresponden a la región, y todos los demás píxeles están desactivados.
'MajorAxisLength'	Devuelve un escalar que especifica la longitud (en píxeles) del eje mayor de la elipse que tiene los mismos segundos momentos centrales normalizados que la región.
'MinorAxisLength'	Devuelve un escalar que especifica la longitud (en píxeles) del eje menor de la elipse que tiene los mismos segundos momentos centrales normalizados que la región.
'Orientation'	Devuelve un escalar que especifica el ángulo entre el eje x y el eje mayor de la elipse que tiene los mismos segundos momentos que la región. El valor está en grados, que van desde -90 a 90 grados. Esta figura ilustra los ejes y la orientación de la elipse. El lado izquierdo de la figura muestra una región de imagen y su elipse correspondiente. El lado derecho muestra la misma elipse con las líneas azules sólidas que representan los ejes, los puntos rojos son los focos y la orientación es el ángulo entre la línea punteada horizontal y el eje mayor.

	
'Perimeter'	Devuelve un escalar que especifica la distancia alrededor del límite de la región. regionprops calcula el perímetro calculando la distancia entre cada par de píxeles adyacentes alrededor del borde de la región. Si la imagen contiene regiones no contiguas, regionprops devuelve resultados inesperados. Esta figura ilustra los píxeles incluidos en el cálculo del perímetro para este objeto. 
'PixelIdxList'	Devuelve un vector p-elemento que contiene los índices lineales de los píxeles en la región.
'PixelList'	Devuelve una matriz p-by-Q que especifica las ubicaciones de los píxeles en la región. Cada fila de la matriz tiene la forma [x y z ...] y especifica las coordenadas de un píxel en la región.
'Solidity'	Devuelve un escalar que especifica la proporción de píxeles en el casco convexo que también se encuentran en la región. Calculado como Área / ConvexArea.
'SubarrayIdx'	Devuelve una matriz de celdas que contiene índices tales que L (idx {:}) extrae los elementos de L dentro del cuadro delimitador de objetos.

4.2.2 Lectura y binarización de la imagen

Para comenzar el algoritmo se debe leer la imagen, la cual se obtiene por medio de una interfaz visual programada en C#, proporcionada por INTESC, que se encarga de abrir la comunicación RS232 entre la computadora y la tarjeta ASSERTA. En dicha interfaz se solicita un frame a la tarjeta, previamente capturado por la cámara, al recibir éste frame se crea una carpeta en donde se guarda un archivo de texto con extensión .txt el cual contiene con todos los valores de los pixeles que componen la imagen, junto con la imagen en un archivo .jpg. Esta interfaz se presenta en la Figura 4.6.

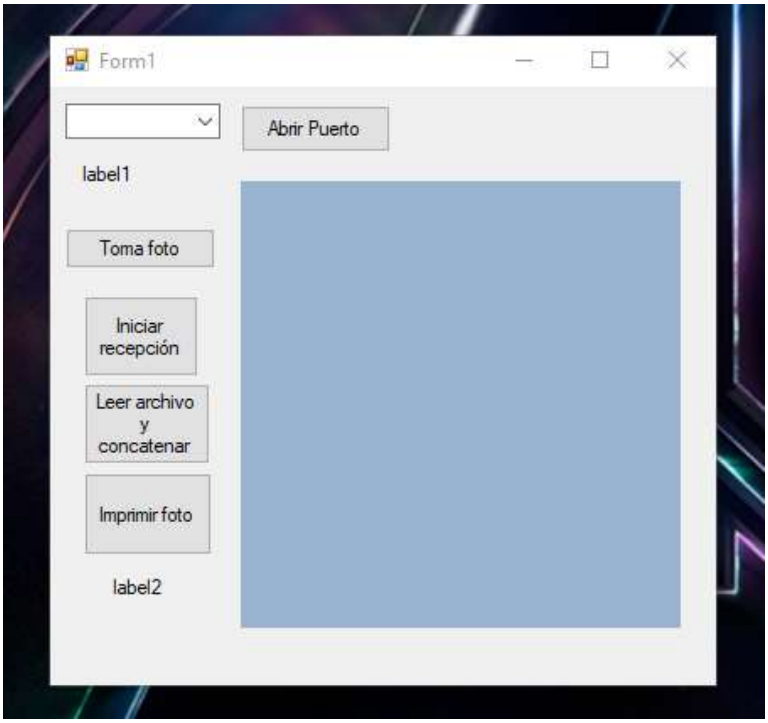


Figura 4.8 Interfaz de la aplicación de escritorio para la comunicación RS232.

Para leer la imagen, que se envía a través de la comunicación RS232, se trabaja en Matlab con el bloc de notas. Dado que las imágenes tienen un buen contraste se binariza la imagen original por medio de umbrales. El código en Matlab de este proceso se muestra en la Figura 4.7 y consiste en lo siguiente:

1. Abrir el archivo .txt donde se guardaron los valores de los pixeles.
2. Leer los datos del archivo de texto abierto con un formato de entero sin signo y guardar los valores en un vector.

3. Cerrar el archivo leído.
4. Ordenar en una matriz especificando sus dimensiones, el vector donde se guardaron los valores de los pixeles.
5. Obtener el tamaño (filas y columnas) de la matriz anterior.
6. Comparar cada valor de pixel con un umbral idóneo para crear la imagen binarizada.
7. Mostrar la imagen binarizada.

```
%% Lectura de la imagen y binarización
fileID = fopen('foto2.txt','r');
formatSpec='%u';
A = fscanf(fileID,formatSpec);
fclose(fileID);
A=reshape(A,[],260);
[f,c]=size(A);

for i=1:f
    for j=1:c
        if A(i,j)<=75
            Imnew(i,j) = 0;
        else
            Imnew(i,j)=1;
        end
    end
end
```

Figura 4.9 Código en Matlab, lectura y binarización.

4.2.3 Cálculo del área y los momentos de inercia de primer y segundo orden

Después de binarizar la imagen, se procede a extraer sus características. Primero se obtiene el área, ya que, para determinar los momentos de inercia y el centro de gravedad, éste parámetro es indispensable. Usando la ecuación (4.1) para calcular el área se ejecuta una sumatoria de todos los pixeles con valor a 1. Esto último, traducido a Matlab, se lleva a cabo en un ciclo for, donde cada iteración representa una fila, anidado con otro ciclo for, donde cada iteración representa una columna. En conjunto, ambos ciclos tienen la función de hacer un barrido de todos los pixeles de la imagen binarizada e ir sumando los pixeles con valor a 1, por lo que ese total representa el área del objeto binarizado.

Para calcular los momentos de primer orden se usa la ecuación (4.2). Se puede observar que al igual que la ecuación del área es una sumatoria, por lo tanto, en Matlab se necesita un ciclo for donde se vaya sumando el número de filas y columnas, si se encuentra un pixel con valor a 1. Como resultado se obtiene el momento de Inercia de primer orden en el eje x y en el eje y, representados por las columnas y filas respectivamente, al no depender de otra característica se puede calcular en el mismo ciclo for que el área.

Obteniendo los datos anteriores el paso siguiente es obtener la posición del centro de gravedad del objeto, para ello se utiliza la ecuación (4.3), lo que nos dice que los momentos de inercia de primer orden son divididos por el área, lo que nos da como resultado el punto fijo donde la distribución de los pixeles es equivalente.

```
%% Calculo del Área y los Momentos de primer orden
%  para el calculo del centroide
area = 0;
Ix = 0;
Iy = 0;

for i=1:f
    for j=1:c
          if (imB(i,j)~=0)
              area=area + 1;
              Ix = Ix + j;
              Iy = Iy + i;
          end
    end
end

Cx = Ix/area;
Cy = Iy/area;
Centroide = [Cx, Cy]
```

Figura 4.10 Código Matalab, cálculo del área, del primer momento y el centroide.

Empleando la ecuación (4.9) observamos que de nuevo se hace uso de una sumatoria, pero para el cálculo de los momentos de segundo orden se requieren los datos del centroide, por lo que en Matlab se hace uso de un ciclo for independiente.

```
%% Calculo de los momentos de segundo orden centralizados
Ixx = 0;
Iyy = 0;
Ixy = 0;

for i = 1:f
    for j = 1: c
        if imB(i,j) ~= 0
            Ixx = Ixx + (j - Centroide(1))^2;
            Iyy = Iyy + (i - Centroide(2))^2;
            Ixy = Ixy + (j - Centroide(1))*(i - Centroide(2));
        end
    end
end
```

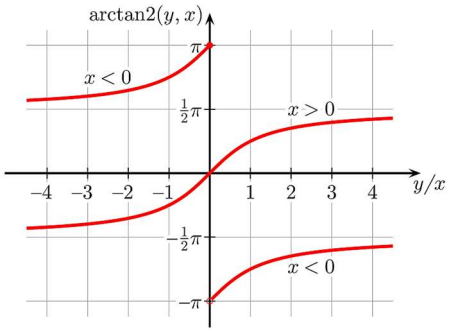
Figura 4.11 Código Matlab, cálculo del segundo momento.

4.2.4 Cálculo de la Orientación

Para determinar la orientación de los objetos captados se hace uso de la ecuación (4.12), pero dicha fórmula puede generar divisiones por cero lo que ocasionaría indeterminaciones en los resultados, para evitar eso se recomienda usar la siguiente formula:

$$\theta = \frac{atan2(y,x)}{2} \tag{4.19}$$

La función atan2, ver Figura 4.12, tiene la ventaja de tomar en cuenta los 4 cuadrantes por lo que los resultados son más fiables, se comienza con el cálculo de los valores x,y para después ser usados en la ecuación (4.19) [27], debido a que la función $\mathrm{Atan2}$ no es posible encontrarla en todos los lenguaje de programación y menos para VHDL se desarrolla en Matlab con sentencias IF acatando las siguientes sentencias:



$$\mathrm{atan2}(y,x) = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0 \\ \arctan(\frac{y}{x}) + \pi & \text{if } x < 0 \text{ and } y \geq 0 \\ \arctan(\frac{y}{x}) - \pi & \text{if } x < 0 \text{ and } y < 0 \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0 \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0 \end{cases}$$

Figura 4.12 Función Atan2.

Como se analizó anteriormente es necesario verificar si el ángulo que se obtiene es del eje del momento de inercia mínimo o su perpendicular, por lo que se realiza una comparación entre la sumatoria de las distancias de los puntos al eje y a su recta perpendicular como lo muestra la ecuación (4.10), se debe tener en cuenta que el ángulo que se forma será positivo en sentido antihorario.

```
%% Cálculo de la Orientación
%  Función ATAN2
x=Iyy-Ixx;
y=2*Ixy;

if x > 0
    tyr=(atan(y/x))/2;
elseif x < 0 && y >= 0
    tyr=(atan(y/x)+pi)/2;
elseif x < 0 && y < 0
    tyr=(atan(y/x)-pi)/2;
elseif x == 0 && y > 0
   tyr= (pi/2)/2;
elseif x == 0 && y < 0
   tyr= (-pi/2)/2 ;
else
    tyr=0;
end
```

Figura 4.13 Código Matlab, función atan2.

```
%verificar si tyr o su perpendicular tyrp=tyr+pi/2 es el eje de momento mín
tyrp=tyr+pi/2;
%Sumas d y dp de las distancias de los puntos al eje y a su perpendicular
d=Iyy*cos(tyr)*cos(tyr)+Ixx*sin(tyr)*sin(tyr)+2*Ixy*sin(tyr)*cos(tyr);
dp=Iyy*cos(tyrp)*cos(tyrp)+Ixx*sin(tyrp)*sin(tyrp)+2*Ixy*sin(tyrp)*cos(tyrp)

if d <= dp
    teta = tyr*180/pi
else
    teta = tyrp*180/pi
end
tetay=teta-90

if tetay < 0 && tetay >= -90
    tetax = tetay+90
else
    tetax = tetay-90
end
```

Figura 4.14 Código Matlab, cálculo de la orientación.

CAPÍTULO 5. RESULTADOS

En este capítulo se presentan los resultados finales del algoritmo implementado, así como el funcionamiento logrado de la tarjeta FPGA y el módulo OV7670.

5.1 Montaje del Hardware y área de trabajo

Las imágenes de las Figuras 5.1 y 5.2 muestran el montaje de la cámara y la Figura 5.3 muestra el montaje completo con la tarjeta FPGA, tal como se desarrollaron las pruebas. Como se observa, la cámara y la tarjeta deben de estar fijas y con una iluminación adecuada, por ello se incorporan LEDs, tal como se muestra en la Figura 5.2. Las pruebas se realizaron bajo un ambiente controlado, por lo que los componentes se ubican sobre un fondo blanco. Esto último es con la finalidad de diferenciarlos y facilitar la binarización por medio de umbral.

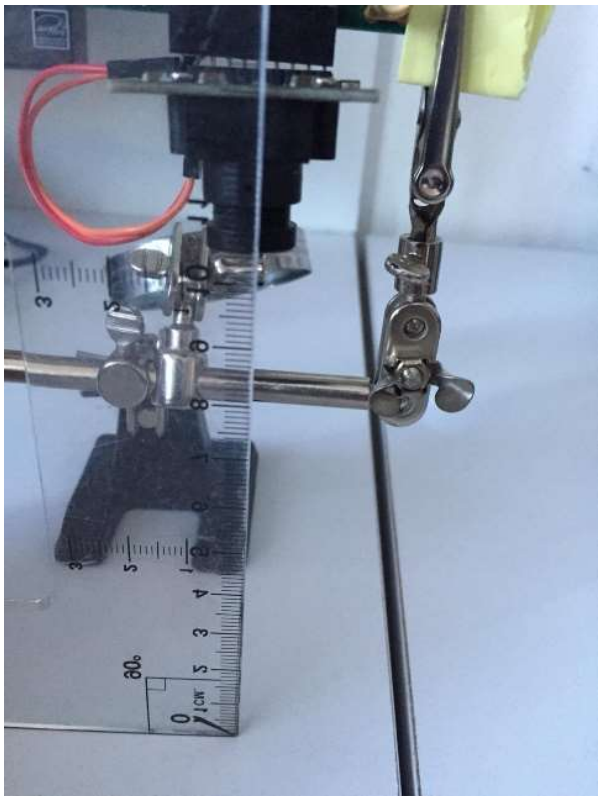


Figura 5.1 Fijación del módulo OV7670.

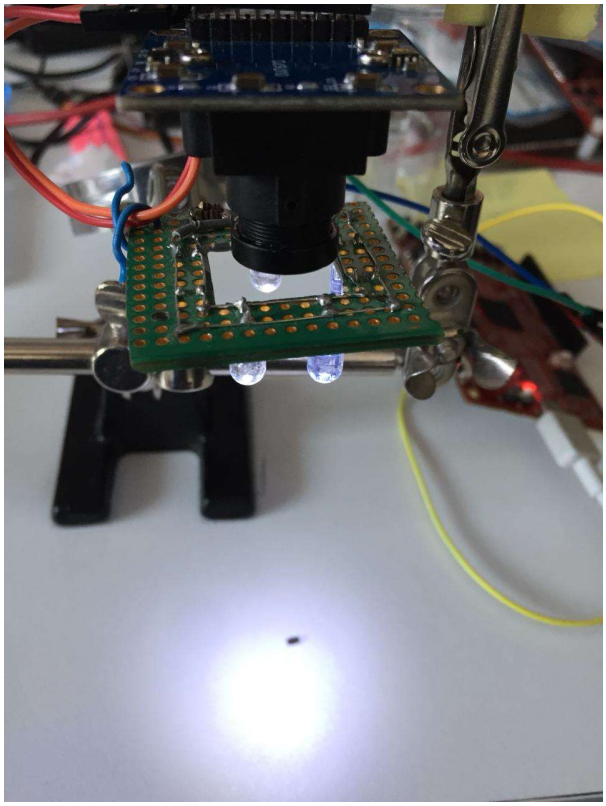


Figura 5.2 Iluminación del área de trabajo.

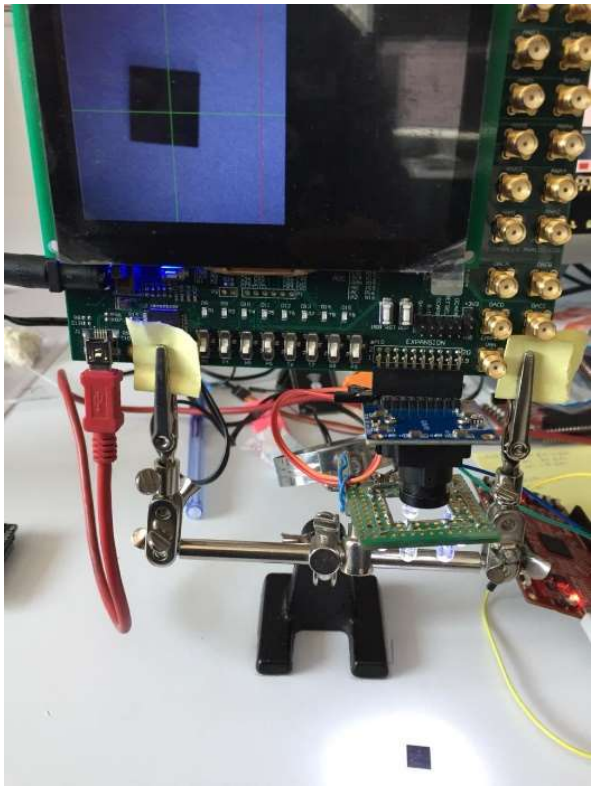


Figura 5.3 Prueba de funcionamiento.

Una vez que todo se encuentra montado y fijo, el programa se descarga a la FPGA para comenzar a grabar con la cámara y al mismo tiempo visualizarlo en la pantalla TFT. En las Figuras 5.4 y 5.5 se observa en la pantalla TFT la captura realizada por cámara.

Figura 5.4 Pantalla TFT montada sobre la tarjeta ASSERTA.

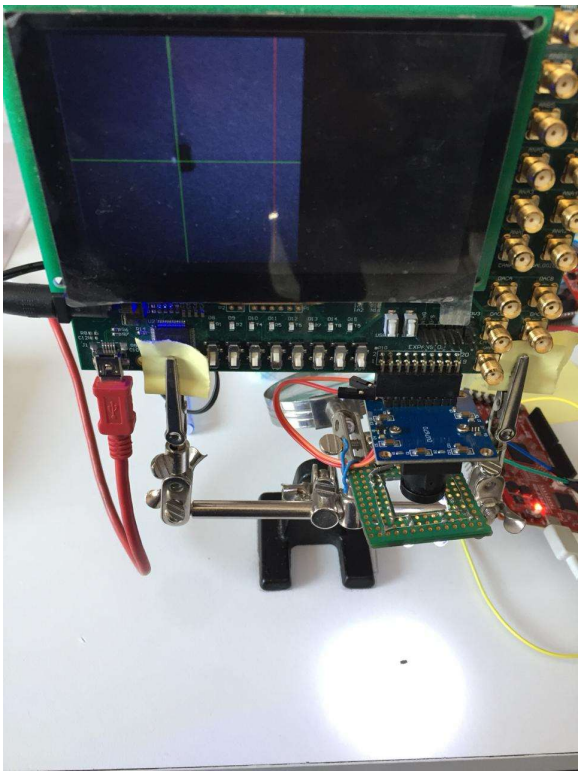


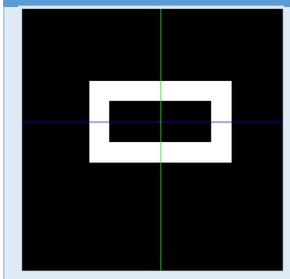
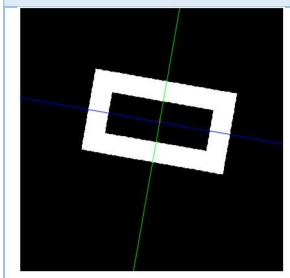
Figura 5.5 Funcionamiento de la Tarjeta ASSERTA y el módulo OV7670.

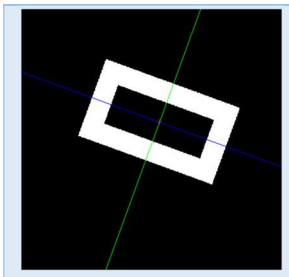
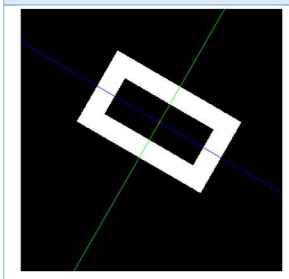
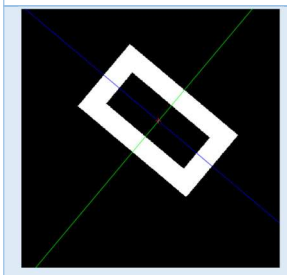
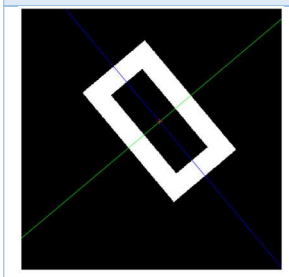
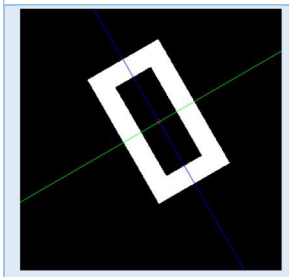
5.2 Pruebas de desempeño con imágenes de Fireworks

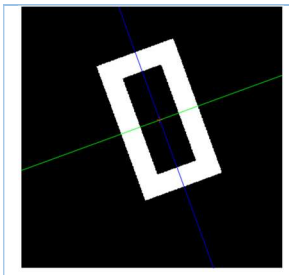
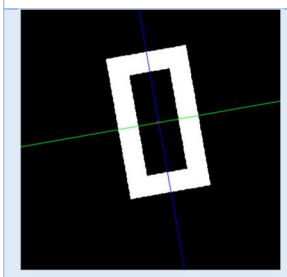
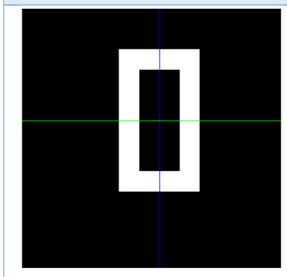
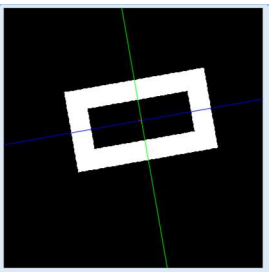
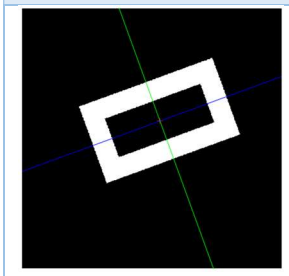
Para comprobar el desempeño del algoritmo de Matlab, se generaron algunas imágenes de prueba en el software de diseño Fireworks CS6. Si la imagen es girada, el software Fireworks nos proporciona el ángulo de rotación. Este ángulo se considera como el valor de referencia para nuestras pruebas.

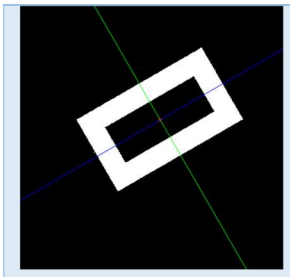
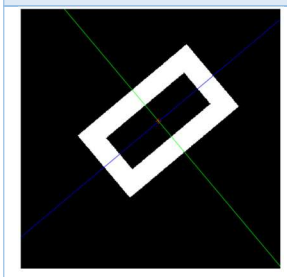
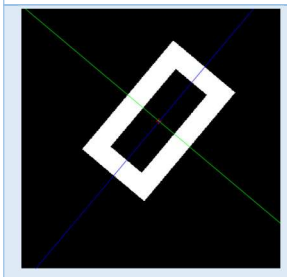
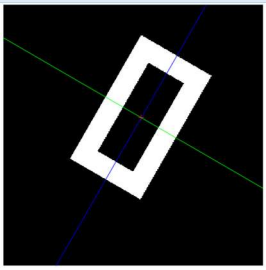
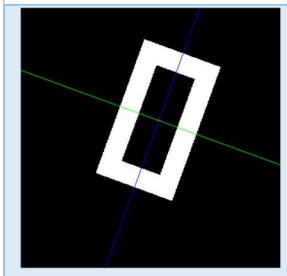
Tal como se presentó anteriormente, Matlab incluye el comando “regionprops”, el cual también nos proporciona el ángulo de rotación de una imagen. Por ello, la Tabla 5.1 presenta la comparación de los resultados obtenidos usando el software Fireworks, el algoritmo propuesto de los momentos de segundo orden y el comando “regionprops”, donde se calculó la rotación en el eje *x* y el eje *y*. Para todos los casos se presenta la precisión obtenida, generando una precisión promedio de 99.966% con un error promedio del ±0.00601 para el algoritmo y de 99.968% con un error promedio del ±0.005955 para el comando “regionprops”.

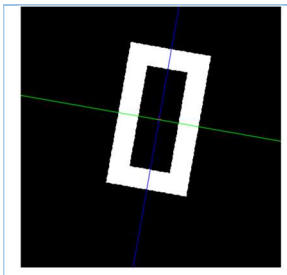
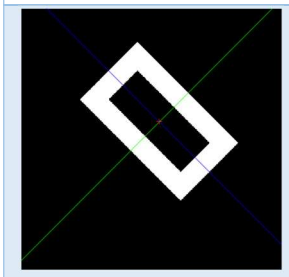
Tabla 5.1 Comparación de los resultados con el algoritmo propuesto y el comando “regionprops”.

Imagen	Ángulo referencia	Ángulo calculado por el algoritmo	Ángulo calculado por “regionprops”	Precisión (Algoritmo vs Regionprops)
	$\angle x = 0°$ $\angle y = 90°$	$\angle x = 0°$ $\angle y = 90°$	$\angle x = 0°$ $\angle y = 90°$	A = 100% E = ±0 R = 100% E = ±0
	$\angle x = -10°$ $\angle y = -80°$	$\angle x = -9.9862°$ $\angle y = -80.0138°$	$\angle x = -9.9862°$ $\angle y = -80.0138°$	A = 99.86% E = ±0.0138 R = 99.86% E = ±0.0138

	$\angle x = -20°$ $\angle y = -70°$	$\angle x = -20.0072°$ $\angle y = -69.9928°$	$\angle x = -20.007°$ $\angle y = -69.993°$	A = 99.98% E = ±0.0072 R = 99.99% E = ±0.007
	$\angle x = -30°$ $\angle y = -60°$	$\angle x = -29.9955°$ $\angle y = -60.0045°$	$\angle x = -29.991°$ $\angle y = -60.009°$	A = 99.98% E = ±0.0045 R = 99.97% E = ±0.009
	$\angle x = -40°$ $\angle y = -50°$	$\angle x = -39.9903°$ $\angle y = -50.0097°$	$\angle x = -39.995°$ $\angle y = -50.005$	A = 99.97% E = ±0.0097 R = 99.98% E = ±0.005
	$\angle x = -50°$ $\angle y = -40°$	$\angle x = -50.0124°$ $\angle y = -39.9876°$	$\angle x = -50.012°$ $\angle y = -39.988°$	A = 99.96% E = ±0.0124 R = 99.97% E = ±0.012
	$\angle x = -60°$ $\angle y = -30°$	$\angle x = -60.0022°$ $\angle y = -29.9978°$	$\angle x = -60.002°$ $\angle y = -29.998°$	A = 99.99% E = ±0.0022 R = 99.99% E = ±0.002

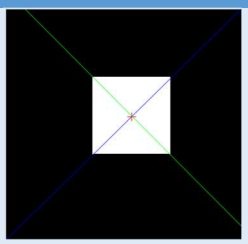
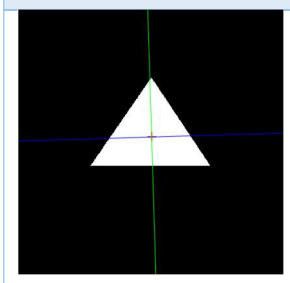
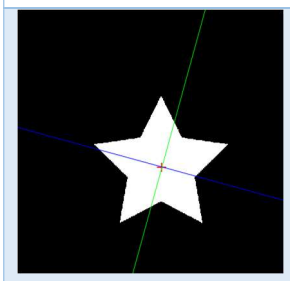
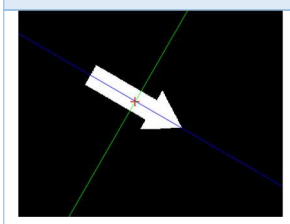
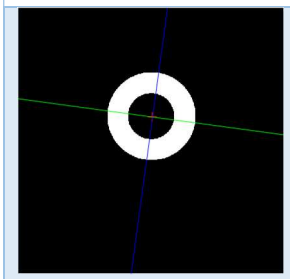
	$\angle x = -70°$ $\angle y = -20°$	$\angle x = -70.0055°$ $\angle y = -19.9945°$	$\angle x = -70.005°$ $\angle y = -19.995°$	A = 99.97% E = ±0.0055 R = 99.97% E = ±0.005
	$\angle x = -80°$ $\angle y = -10°$	$\angle x = -80.0042°$ $\angle y = -9.9958°$	$\angle x = -80.004°$ $\angle y = -9.996°$	A = 99.95% E = ±0.0042 R = 99.96% E = ±0.004
	$\angle x = 90°$ $\angle y = 0°$	$\angle x = 90°$ $\angle y = 0°$	$\angle x = 90°$ $\angle y = 0°$	A = 100% E = ±0 R = 100% E = ±0
	$\angle x = 10°$ $\angle y = 80°$	$\angle x = 9.9862°$ $\angle y = 80.0138°$	$\angle x = 9.9862°$ $\angle y = 80.0138°$	A = 99.86% E = ±0.0138 R = 99.86% E = ±0.0138
	$\angle x = 20°$ $\angle y = 70°$	$\angle x = 20.0072°$ $\angle y = 69.9928°$	$\angle x = 20.007°$ $\angle y = 69.993°$	A = 99.98% E = ±0.0072 R = 99.99% E = ±0.007

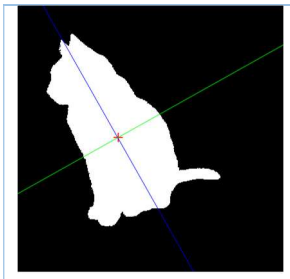
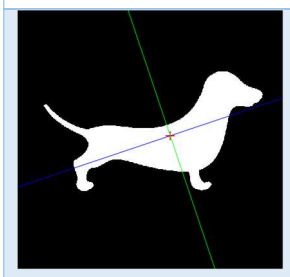
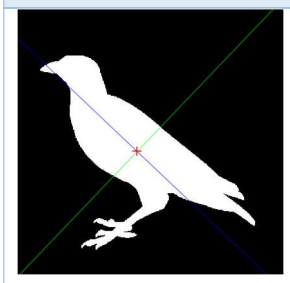
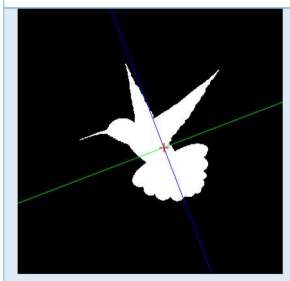
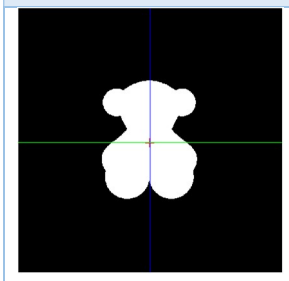
	$\angle x = 30°$ $\angle y = 60°$	$\angle x = 29.9955°$ $\angle y = 60.0045°$	$\angle x = 29.991°$ $\angle y = 60.009°$	A = 99.98% E = ±0.0045 R = 99.97% E = ±0.0045
	$\angle x = 40°$ $\angle y = 50°$	$\angle x = 39.9903°$ $\angle y = 50.0097°$	$\angle x = 39.995°$ $\angle y = 50.005$	A = 99.97% E = ±0.0097 R = 99.98% E = ±0.005
	$\angle x = 50°$ $\angle y = 40°$	$\angle x = 50.0136°$ $\angle y = 39.9864°$	$\angle x = 50.014°$ $\angle y = 39.986°$	A = 99.96% E = ±0.0136 R = 99.96% E = ±0.014
	$\angle x = 60°$ $\angle y = 30°$	$\angle x = 60.0022°$ $\angle y = 29.9978°$	$\angle x = 60.002°$ $\angle y = 29.998°$	A = 99.99% E = ±0.0022 R = 99.99% E = ±0.002
	$\angle x = 70°$ $\angle y = 20°$	$\angle x = 70.0055°$ $\angle y = 19.9945°$	$\angle x = 70.005°$ $\angle y = 19.995°$	A = 99.97% E = ±0.0055 R = 99.97% E = ±0.005

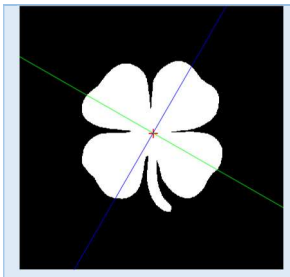
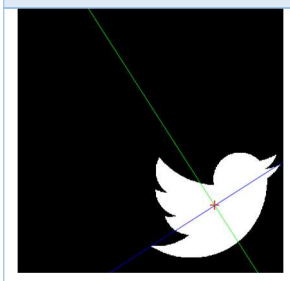
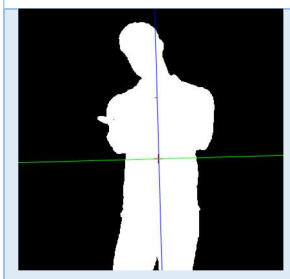
	$\angle x = 80°$ $\angle y = 10°$	$\angle x = 80.0042°$ $\angle y = 9.9958°$	$\angle x = 80.004°$ $\angle y = 9.996°$	A = 99.95% E = ±0.0042 R = 99.96% E = ±0.004
	$\angle x = -45°$ $\angle y = -45°$	$\angle x = -45°$ $\angle y = -45°$	$\angle x = -44.997°$ $\angle y = -450.003°$	A = 100% E = ±0 R = 99.99% E = ±0.003
	$\angle x = 45°$ $\angle y = 45°$	$\angle x = 45°$ $\angle y = 45°$	$\angle x = -44.997°$ $\angle y = -450.003°$	A = 100% E = ±0 R = 99.99% E = ±0.003

La prueba se repitió con 13 imágenes diferentes, de figuras geométricas y objetos más complejos. La Tabla 5.2 muestra la comparación entre los resultados obtenidos con el algoritmo y con el comando “regionprops”. Para este caso se calculó el error obtenido en cada prueba, obteniendo un error promedio de ±0.000285.

Tabla 5.2 Comparación de resultados obtenidos con el algoritmo y regionprops.

Figura	Ángulo obtenido por el algoritmo	Ángulos obtenido por "regionprops"	Error
	$\angle x = 44.7259°$ $\angle y = 45.2741°$	$\angle x = 44.726°$ $\angle y = 45.274°$	Error = ±0.0001
	$\angle x = 1.7473°$ $\angle y = 88.2527°$	$\angle x = 1.7473°$ $\angle y = 88.2527°$	Error = 0
	$\angle x = -15.3436°$ $\angle y = -74.6564°$	$\angle x = -15.344°$ $\angle y = -74.656°$	Error = ±0.0004
	$\angle x = -29.9976°$ $\angle y = -60.0024°$	$\angle x = -29.998°$ $\angle y = -60.002$	Error = ±0.0004
	$\angle x = 82.2314°$ $\angle y = 7.7686°$	$\angle x = 82.231°$ $\angle y = 7.769°$	Error = ±0.0004

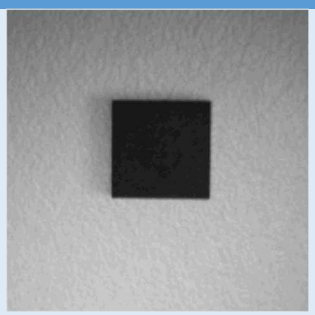
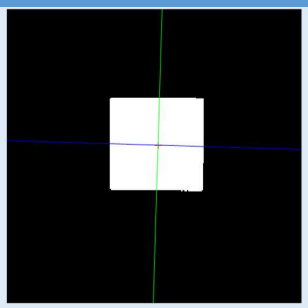
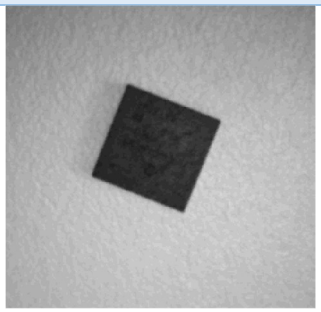
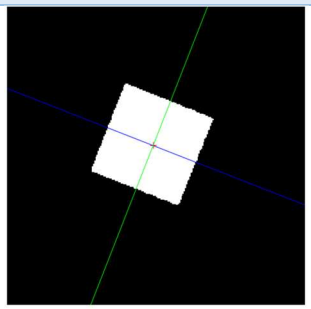
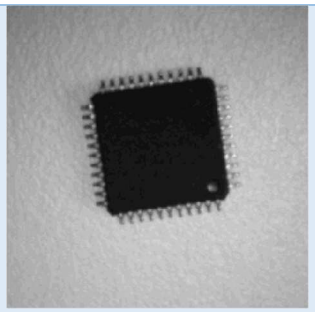
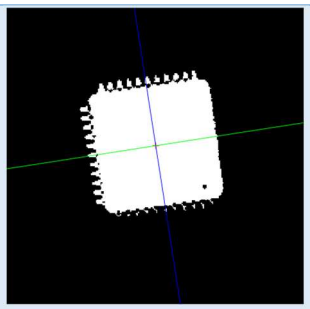
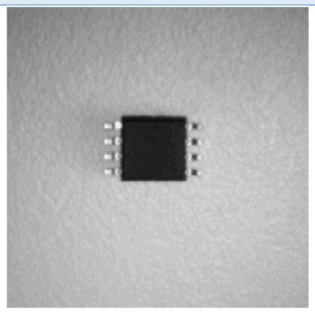
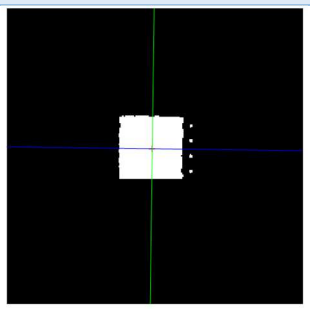
	$\angle x = -60.6202°$ $\angle y = -29.3798°$	$\angle x = -60.62°$ $\angle y = -29.38°$	Error = ±0.0002
	$\angle x = 18.5647°$ $\angle y = 71.4353°$	$\angle x = 18.565°$ $\angle y = 71.435°$	Error = ±0.0003
	$\angle x = -43.5817°$ $\angle y = -46.4183°$	$\angle x = -43.582°$ $\angle y = -46.418°$	Error = ±0.0003
	$\angle x = -69.0746°$ $\angle y = -20.9254°$	$\angle x = -69.075°$ $\angle y = -20.925°$	Error = ±0.0004
	$\angle x = -89.9925°$ $\angle y = -0.0075°$	$\angle x = -89.992°$ $\angle y = -0.008°$	Error = ±0.0005

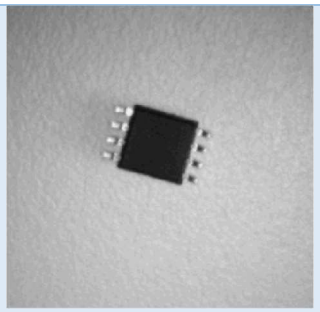
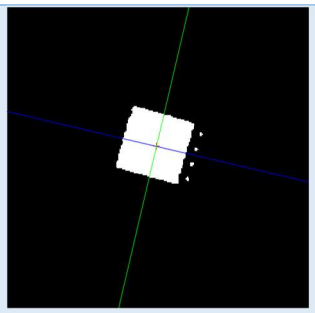
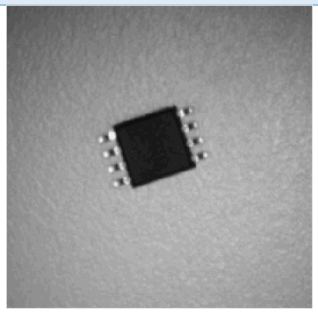
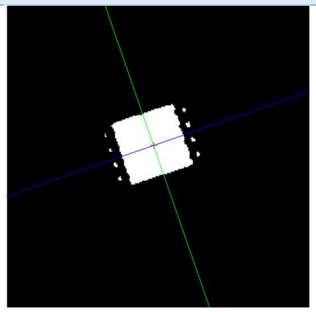
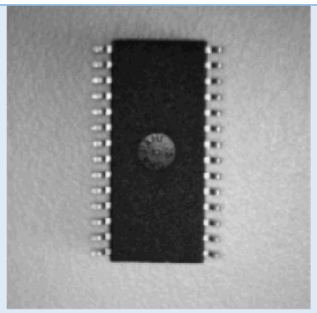
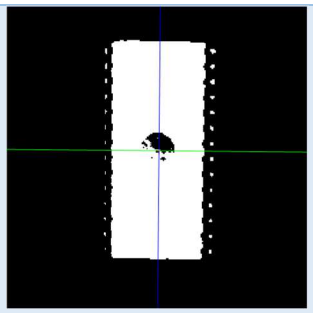
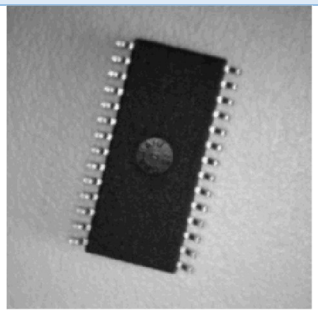
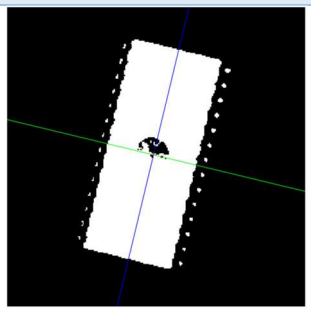
	$\angle x = 60.1072°$ $\angle y = 29.8928°$	$\angle x = 60.107°$ $\angle y = 29.893°$	Error = ±0.0002
	$\angle x = 32.6269°$ $\angle y = 57.3731°$	$\angle x = 32.627$ $\angle y = 57.373°$	Error = ±0.0001
	$\angle x = -88.4094°$ $\angle y = -1.5906°$	$\angle x = -88.409°$ $\angle y = -1.591°$	Error = ±0.0004

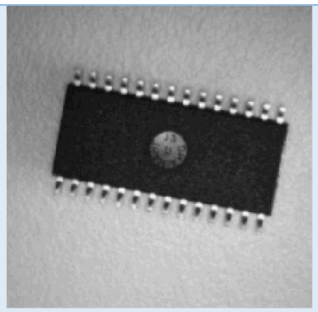
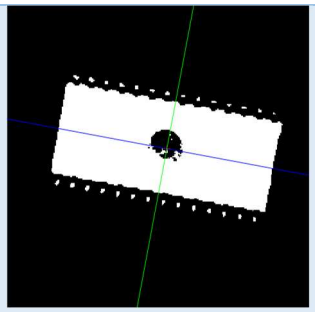
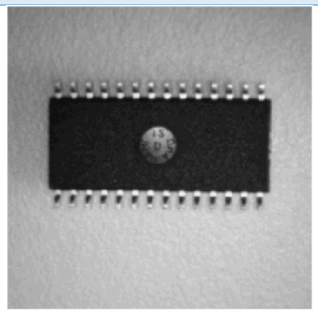
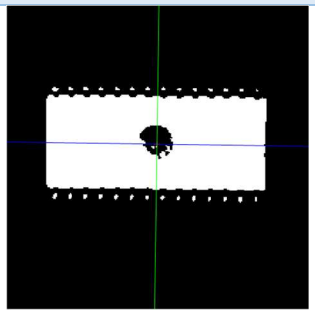
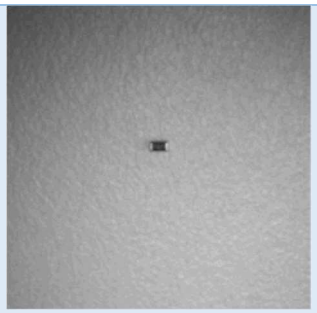
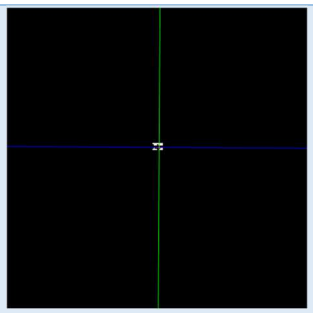
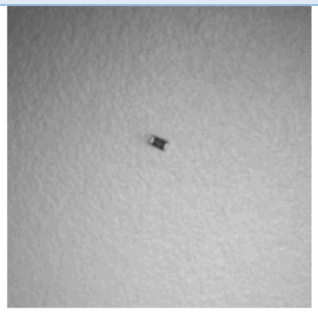
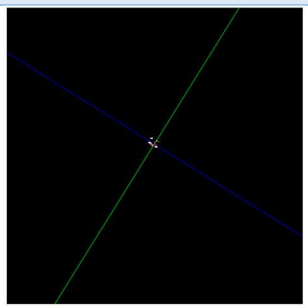
5.2 Pruebas de desempeño con imágenes capturadas por la cámara OV7670

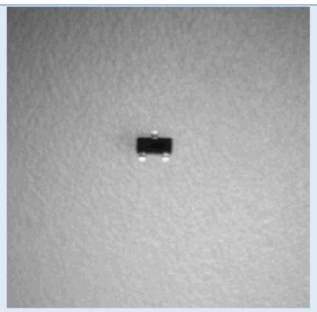
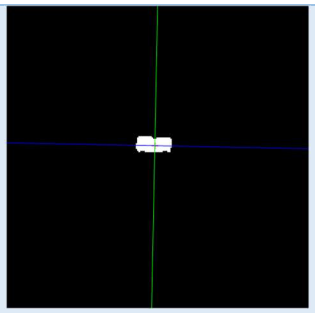
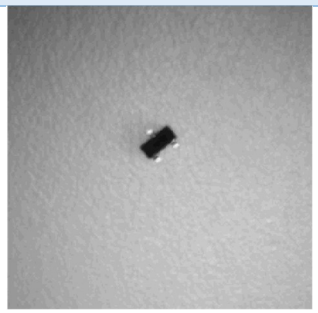
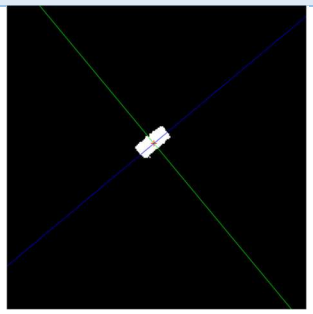
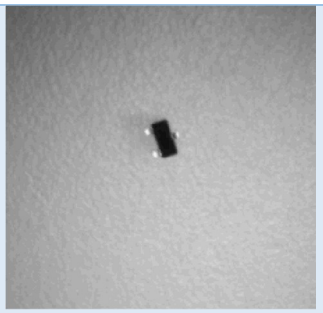
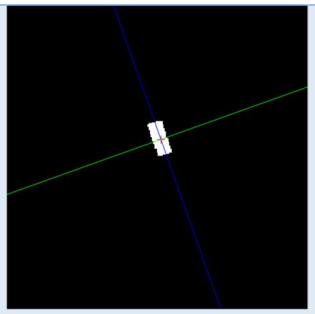
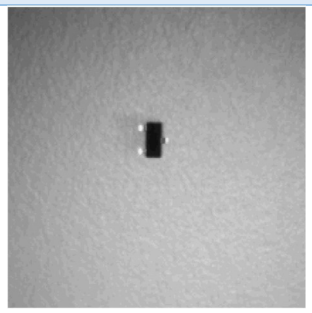
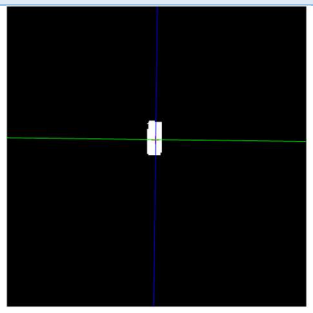
Las pruebas finales se realizaron con las imágenes de algunos circuitos integrados de montaje superficial, tales como los que son colocados en una máquina pick & place. La Tabla 5.3 muestra los resultados obtenidos con el algoritmo. Para éstas pruebas, solo se presenta el ángulo de rotación obtenido por el algoritmo al no contar con un ángulo exacto de referencia para calcular la precisión del algoritmo, deduciendo la precisión de las pruebas anteriores frente a los ángulos de referencia el error obtenido de las imágenes capturadas por la cámara de los componentes será mínimo.

Tabla 5.3 Resultados de la prueba final del algoritmo y componentes.

Imagen del componente	Imagen procesada	Resultado obtenido
		$\angle x = -1.7264°$ $\angle y = -88.2736°$
		$\angle x = -21.3809°$ $\angle y = -68.6191°$
		$\angle x = -81.1679°$ $\angle y = -8.8321°$
		$\angle x = -0.7624°$ $\angle y = -89.2376°$

		$\angle x = -13.1617\ °$ $\angle y = -76.8383°$
		$\angle x = 19.0787°$ $\angle y = 70.9213°$
		$\angle x = 89.4798°$ $\angle y = 0.5202°$
		$\angle x = 76.5440°$ $\angle y = 13.4560°$

		$\angle x = -10.6010°$ $\angle y = -79.3990°$
		$\angle x = -0.6512°$ $\angle y = -89.3488°$
		$\angle x = -0.3422°$ $\angle y = -89.6578°$
		$\angle x = -31.8802°$ $\angle y = -58.1198°$

		$\angle x = -1.1244°$ $\angle y = -88.8756°$
		$\angle x = 39.7708°$ $\angle y = 50.2292°$
		$\angle x = -69.9665°$ $\angle y = -20.0335°$
		$\angle x = 89.2955°$ $\angle y = 0.7045°$

CONCLUSIONES

Se presentó un algoritmo de visión artificial capaz de calcular la orientación de los objetos captados por una cámara OV7670 implementada en una FPGA. Esto con la finalidad de darle solución al problema sobre el montaje superficial que tiene la máquina Pick & Place de INTESC al ensamblar componentes a sus tarjetas de desarrollo. Dicho algoritmo se basa en los principios de los momentos de segundo orden de inercia vistos en Estática.

El trabajo se dividió en 2 etapas; la primera consistió en la programación en lenguaje VHDL para implementar el módulo de la cámara OV7670 en un FPGA y la segunda en el desarrollo del algoritmo en MATLAB para determinar los grados de orientación de los objetos vistos por la cámara.

En la primera etapa se realizó en el software de ISE el módulo principal que contiene los IP Cores compartidos por INTESC para posteriormente programar la tarjeta ASSERTA y realizar las pruebas necesarias para su correcto funcionamiento.

En la segunda etapa se trabajó en MATLAB para desarrollar el algoritmo para calcular la orientación por medio de los momentos de inercia de segundo orden en imágenes binarias, los cuales tienen cierta similitud con los momentos de inercia estudiados en Estática.

En ambas etapas se presentaron ciertas complicaciones, pues no había mucha experiencia en la programación de VHDL, por lo que se tuvo que investigar acerca de la programación avanzada. Otra complicación fue el desconocimiento de la cámara OV7670 y su forma de operar, además de la escasa y simple información que hay sobre los temas abordados en este proyecto.

En las pruebas realizadas se observó que el umbral a la hora de realizar el procesamiento de imágenes tiene un papel muy importante para el cálculo de la orientación, cualquier variación puede dar otro resultado distante al real, para evitar estos problemas se pretende desarrollar a futuro un algoritmo de binarización por medio de un umbral automático.

El proyecto está pensado para implementarlo en un futuro en un FPGA, pero por su compleja forma de programar operaciones sencillas como lo es una simple división o implementar funciones trigonométricas las cuales son indispensables para el cálculo de la orientación, no se completó en este trabajo, lo cual se queda para trabajo futuro.

En base a los resultados obtenidos con el algoritmo desarrollado se puede concluir que se cumplieron con todos los objetivos planteados al comienzo de éste documento, cabe mencionar que éste proyecto queda como referente y motivación para que los estudiantes de mecatrónica del Instituto Tecnológico Superior de Atlixco colaboren con empresas para innovar y desarrollar nuevas tecnologías dando solución a los problemas con los que se enfrentan, de esa forma amplían sus conocimientos en su formación profesional.

REFERENCIAS

[1]. INTESC “Electronic & Embedded”. Visitado 01-04-2018. On line: http://www.intesc.mx/.

[2]. Prasad, R. “Surface mount technology: principles and practice”. Springer Science & Business Media, 2013.

[3]. Ganeswara-Rao M. V., Panakala, R. K. and Mallikarjuna-Prasad A. “Image Processing using FPGAs: a Framework”. IJCTA International Science Press, vol. 9, no. 19, pp. 9191-9197, 2016.

[4]. AliExpress. Búsqueda: Pick and place machine. Visitado 01-04-2018. On line: https://mx.aliexpress.com/

[5]. Protoplace S. User Manual Data Sheet. Visitado: 01-04-2018. On line: https://www.lpkfusa.com/datasheets/prototyping/ProtoPlace%20S%20Specifications%20'16.pdf

[6]. Duquette, D. W., et al. “Image analysis for pick and place machines with in situ component placement inspection”. U.S. Patent No 7813559, 12 Oct. 2010.

[7]. Arroyo, J. R., Cifuentes, G, “Pick and Place para montaje de componentes de montaje superficial SMD”, Tesis de Licenciatura, Instituto Politécnico Nacional, México, 2015.

[8]. Verstegen, P. P. H., Van Gastel, J. M. M., & Spronck, J. W., “Design of a novel single camera vision system for both component and board alignment in pick and place machines”. IFAC Proceedings Volumes, vol. 39, no 16, p. 706-712, 2006.

[9]. Ray P. Prasat, “Surface Mount Technology: Principles and Practice”, 2nd Edition, 1997.

[10]. Micro Consulting Inc, Surface Mount Technology. Visitado 20-05-2018. On line: http://www.microconsulting.ca/special-processes/smt/

[11]. Phil Zarrow, Debra Kopp, “Surface Mount Technology terms and concepts”, Elsevier, 2008.

[12]. Sze S. M. “Semicondutor Devices physics and technology”, 2nd Edition, John Wiley & Sons, Inc., 2002.

[13]. Glenn R. Blackwell, James K. Hollomon, “Surface-mount Technology for PC Boards”, Thomson Delmar Learning, 2006.

[14]. Pérez L. Serafín, Soto C. Enrique, Fernández G. Santiago, “Diseño de sistemas digitales con VHDL”, 1ra Edición, Thomson, 2002.

[15]. Maxinez David G. “Programación de sistemas digitales con VHDL”, 1era Edición, Grupo editorial Patria, 2013.

[16]. Simon J. Bleiker, “Performance Analysis of Nanoelectromechanical Relay-Based Field-Programmable Gate Arrays”. Vistado 20-05-2018. On line: https://www.researchgate.net/publication/323820898_Performance_Analysis_of_Nanoelectromechanical_Relay-Based_Field-Programmable_Gate_Arrays.

[17]. ISE Desing suite, “Herramienta de software”. Visitado 20-05-2018. On line: https://en.wikipedia.org/wiki/Xilinx_ISE

[18]. ISE Desing suite. “Logo del software ISE Desing Suite”. Visitado 20-05-2018. On line: https://upload.wikimedia.org/wikipedia/en/0/0a/XilinxISE_DS_Logo.jpg.

[19]. INTESC, Manual de usuario Tarjeta FPGA ASSERTA. Visitado 30-06-2018. On line: https://intesc.mx/wp-content/uploads/2019/01/ManualAssertaRevB.pdf.

[20]. NAYLAMP MECHATRONICS, Módulo Cámara VGA OV7670. Visitado 30-06-2018. On line: https://naylampmechatronics.com/sensores-luz-y-sonido/140-modulo-camara-vga-ov7670.html

[21]. Pablo M. Blanco, “Desarrollo de un sistema de transmisión de vídeo con módulo WiFi”, Trabajo de Fin de grado, Universidad Politécnica de Madrid, septiembre, 2016.

[22]. OmniVision, User Manual Data Sheet OV7670. Visitado: 30-06-2018. On line: http://www.haoyuelectronics.com/Attachment/OV7670%20+%20AL422B%28FIFO%29%20Camera%20Module%28V2.0%29/OV7670%20datasheet%28V1.4%29.pdf

[23]. Eusebio de la F. Lopez, Félix M. Trespaderne. “Visión artificial industrial: Procesamiento de imágenes para inspección automática y robótica”, 1era Edición, Universidad de Valladolid, 2012.

[24]. Ana G. Marcos… [etal.] (integrantes del Grupo de Investigaciones EDMANS). “Técnicas y algoritmos básicos de visión artificial”, 1era Edición, Universidad de la Rioja, 2006.

[25]. Ferdinand P. Beer, E. Russell, David F. Mazurek, Elliot R. Eisenberg. “Mecánica vectorial para ingenieros: Estática”, 9na Edición, Mc Graw Hill, 2010.

[26]. MATLAB, Measure propieties of image regions. Regionprops. Visitado 30-06-2018. On line: https://www.mathworks.com/help/images/ref/regionprops.html

[27]. Atan2, “Definition the function atan2”. Visitado: 20-07-2018. On line: https://en.wikipedia.org/wiki/Atan2

ANEXO

CÓDIGO EN VHDL DEL MÓDULO PRINCIPAL “CAPTURA PIXEL”

Proceso del funcionamiento de la cámara.

```
--DE LA CAMARA--
GUARDAR <= '1' WHEN PIXEL >= 100 AND PIXEL < (100+RES) AND FILA >= 100 AND FILA < (100+RES) ELSE '0';
--DE TFT---
IMPRIMIR <= '1' WHEN CONTA_PIXEL >= 10 AND CONTA_PIXEL < (10+RES) AND CONTA_FILA >= 1 AND CONTA_FILA < 1+ RES ELSE '0';
```

```
PROCESS(PCLK)
BEGIN
   IF RISING_EDGE(PCLK) THEN
      IF EDO = 0 THEN--ESPERAMOS A QUE SE CONFIGURE LA CÁMARA
         IF config_finished = '1' THEN
            EDO <= 1;
            LEDS <= "1100";
         ELSE
            EDO <= 0;
         END IF;

      ELSIF EDO = 1 THEN
         IF VSYNC = '1' THEN--ESPERAMOS FRAME NUEVO
            EDO <= 2;
            LEDS <= "1110";
         ELSE
            EDO <= 1;
         END IF;

      ELSIF EDO = 2 THEN--CAPTURAMOS PRIMER BYTE
         IF HREF = '1' THEN
            EDO <= 3;
            DO1 <= DATA_CAM;
         ELSE
            EDO <= 2;
            LEDS <= "1100";
         END IF;

      ELSIF EDO = 3 THEN
         IF HREF = '1' THEN
            IF GUARDAR = '1' THEN
               DIA <= DO1(2 DOWNTO 0)&DATA_CAM(7 DOWNTO 6);
               CONTA_ADD_A <= CONTA_ADD_A +1;

      --Proceso de la binarización de la imagen de entrada
               IF  DIA > conv_std_logic_vector(200,8) THEN
                  PIX_BIN <= "00000";
               ELSE
                  PIX_BIN <= "11111";
               END IF;

               IF CONTA_ADD_A = MAX_ADD THEN
                  CONTA_ADD_A <= 0;
               END IF;
            END IF;
            PIXEL <= PIXEL+1;
               IF PIXEL = 639 THEN
                  PIXEL <= 0;
                  FILA <= FILA+1;
                  IF FILA = 479 THEN
                     FILA <= 0;
                     EDO <= 4;
                  ELSE
                     EDO <= 2;
                  END IF;
```

```
                ELSE
                   EDO <= 2;
                END IF;
             ELSE
                EDO <= 3;
                LEDS <= "1100";
             END IF;

          ELSIF EDO = 4 THEN
             IF MANDAR = '1' THEN
                EDO <= 5;
                GO <= '1';
             ELSE
                EDO <= 1;
             END IF;

          ELSIF EDO = 5 THEN
             IF FIN_ENVIO = '1' THEN
                EDO <= 1;
                GO <= '0';
             ELSE
                EDO <= 5;
             END IF;

          END IF;
       END IF;
    END PROCESS;
```

Proceso del funcionamiento de la pantalla TFT.

```
PROCESS(TFTCLK_S)
BEGIN
   IF RISING_EDGE(TFTCLK_S) THEN
      IF IMPRIMIR = '1' THEN
            R_IN <= DOB&"000";
            G_IN <= DOB&"000";
            B_IN <= DOB&"000";
         --end if;
         CONTA_ADD_B1 <= CONTA_ADD_B1+1;
         IF CONTA_ADD_B1 = MAX_ADD THEN
            CONTA_ADD_B1 <= 0;
         END IF;
      ELSE
         R_IN <= "00000000";
         G_IN <= "00000000";
         B_IN <= "00000000";
      END IF;
   END IF;
END PROCESS;
```

Proceso de la comunicación RS232.

```
PROCESS(CLK)
BEGIN
   IF RISING_EDGE(CLK) THEN
      IF EDO_ENV = 0 THEN
         IF RX_IN = '1' THEN
            IF DOUT = X"55" THEN
               FIN_ENVIO <= '0';
               EDO_ENV <= 4;
               MANDAR <= '1';
            ELSE
               EDO_ENV <= 0;
               MANDAR <= '0';
            END IF;
         ELSE
            EDO_ENV <= 0;
         END IF;
```

```
            ELSIF EDO_ENV = 1 THEN
               TX_INI <= '1';
               EDO_ENV <= 2;
               DATAIN <= DOB&"000";

            ELSIF EDO_ENV = 2 THEN
               IF TX_FIN = '1' THEN
                  EDO_ENV <= 3;
                  TX_INI <= '0';
                  ADD_ENV <= ADD_ENV+1;
                  IF ADD_ENV = MAX_ADD THEN
                     ADD_ENV <= 0;
                     FIN_ENVIO <= '1';
                  END IF;
               ELSE
                  EDO_ENV <= 2;
               END IF;

            ELSIF EDO_ENV = 3 THEN
               IF CONTA_ENV = MAX_ENV THEN
                  CONTA_ENV <= 0;
                  IF ADD_ENV = MAX_ADD THEN
                     FIN_ENVIO <= '1';
                     EDO_ENV <= 0;
                  ELSE
                     EDO_ENV <= 1;
                  END IF;

               ELSE
                  CONTA_ENV <= CONTA_ENV +1;
                  EDO_ENV <= 3;
               END IF;

            ELSIF EDO_ENV = 4 THEN
               IF GO = '1' THEN
                  EDO_ENV <= 1;
               ELSE
                  EDO_ENV <= 4;
               END IF;

            END IF;
         END IF;
      END PROCESS;

      CONTA_ADD_B <= ADD_ENV WHEN MANDAR = '1' ELSE CONTA_ADD_B1;
```

ÍNDICE DE FIGURAS

CAPÍTULO 2

Figura 2.1 Máquina Pick & Place ProtoPlace S [5]..... 6
Figura 2.2 Tecnología de Montaje Superficial (SMT) [10]…..... 8
Figura 2.3 Arquitectura global de un FPGA [16]…..... 11
Figura 2.4 Fabricantes de FPGAs…..... 13

CAPÍTULO 3

Figura 3.1 Logotipo del software ISE Design Suite de Xilinx [18]…… 14
Figura 3.2 Interfaz de Usuario del software ISE Desing Suite…….. 15
Figura 3.3 Tarjeta de desarrollo FPGA ASSERTA [19]……..... 16
Figura 3.4 Diagrama a bloques de la Tarjeta de desarrollo ASSERTA [19]……... 17
Figura 3.5 Módulo OV7670 [21]…….... 18
Figura 3.6 Diagrama de bloques funcional [22]…….. 19
Figura 3.7 Pines del módulo OV7670…….. 20
Figura 3.8 Diagrama de temporización de los Frame VGA [22]……... 21
Figura 3.9 Diagrama de temporización del formato de salida RGB 565 [22]…….. 22
Figura 3.10 Diagrama general del Proyecto……... 23
Figura 3.11 Diagrama a bloques del modulo principal (TOP)……... 24
Figura 3.12 Submódulos implementados en el módulo principal (TOP)…… 25
Figura 3.13 Diagrama de transición de estados de la recepción del vídeo…… 26
Figura 3.14 Diagrama de transición de estados de la comunicación RS232…… 28
Figura 3.15 Imagen binarizada, donde el pixel en blanco tiene el valor a 1 y el negro valor a 0 [23]….. 29

CAPÍTULO 4

Figura 4.1 Determinación del área de una imagen [24]…… 33
Figura 4.2 Centro de gravedad de engrane [23]…… 34
Figura 4.3 Ejes máximo y mínimo de un objeto [24]…….. 37
Figura 4.4 Orientación de una imagen [23]…….... 37
Figura 4.5 Circularidad de una imagen [23]…….... 38
Figura 4.6 Envolvente convexa [23]…….... 39
Figura 4.7 Rectángulo envolvente de tamaño mínimo [23]…….. 39

Figura 4.8 Interfaz de la aplicación de escritorio para la comunicación RS232……..... 46
Figura 4.9 Código en Matlab, lectura y binarización…….. 47
Figura 4.10 Código Matlab, cálculo del área, del primer momento y el centroide…….. 48
Figura 4.11 Código Matlab, cálculo del segundo momento……..... 49
Figura 4.12 Función Atan2…… 49
Figura 4.13 Código Matlab, función atan2…….. 50
Figura 4.14 Código Matlab, cálculo de la orientación……..... 50

CAPÍTULO 5

Figura 5.1 Fijación del módulo OV7670…….. 51
Figura 5.2 Iluminación del área de trabajo…….. 52
Figura 5.3 Prueba de funcionamiento…….. 52
Figura 5.4 Pantalla TFT montada sobre la tarjeta ASSERTA…….. 53
Figura 5.5 Funcionamiento de la Tarjeta ASSERTA y el módulo OV7670……..... 53

ÍNDICE DE TABLAS

Tabla 4.1 Propiedades medidas con el comando “regionprops”…….. 41
Tabla 5.1 Comparación de los resultados con el algoritmo propuesto y el comando “regionprops”…... 54
Tabla 5.2 Comparación de resultados obtenidos con el algoritmo y regionprops…… 59
Tabla 5.3 Resultados de la prueba final del algoritmo y componentes…… 62