



UNIVERSIDAD POLITÉCNICA DE VICTORIA

UNIVERSIDAD POLITÉCNICA  
DE VICTORIA



**CLASIFICADOR DE NARANJAS POR TAMAÑO Y COLOR  
IMPLEMENTADO EN FPGA USANDO APRENDIZAJE  
SUPERVISADO**

T E S I S  
QUE PARA OBTENER EL GRADO DE  
**MAESTRO EN INGENIERÍA**  
P R E S E N T A

**Ismael Antonio Dávila Rodríguez**

DIRECTOR DE TESIS:  
**DR. MARCO AURELIO NUÑO MAGANDA**

CO-DIRECTOR DE TESIS:  
**DR. YAHIR HERNÁNDEZ MIER**

Cd. Victoria, Tamaulipas, México

Diciembre, 2020



© Derechos reservados por  
Ismael Antonio Dávila Rodríguez  
2020



Agradecimiento especial al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo financiero recibido durante los estudios de maestría.



La tesis presentada por Ismael Antonio Dávila Rodríguez fue aprobada por:

---

DR. MARCO AURELIO NUÑO MAGANDA, Director

---

DR. YAHIR HERNÁNDEZ MIER, Co-Director

Cd. Victoria, Tamaulipas, México, 7 de Diciembre de 2020



Este trabajo de tanto esfuerzo lo dedico enteramente a mis padres Ismael Davila y Socorroto Rodríguez por su amor incondicional en todos mis años de estudio, a mi familia por siempre confiar en mi capacidad y a mi novia Adriana Echartea por tanto apoyo y comprensión durante este grado.



# Agradecimientos

Primeramente quiero agradecer al **Ing. Arturo Martinez** por su apoyo en todas las etapas del proyecto, con herramientas, su guía y conocimientos de electrónica/maquinaria industrial.

También quiero agradecer a los Doctores que me enseñaron tanto en este grado:

**Dr. Marco Nuño** por sus grandes e interesantes conocimientos en el HDL, Android, Java; y por su guía en el desarrollo del proyecto y escrito.

**Dr. Said Polanco** por sus profundos conocimientos y experiencia en modelos de aprendizaje automático.

**Dr. Yahir Hernandez** por su apoyo referente a la ponencia del congreso internacional Reconfig 2019, producto de este grado.

**Dr. Amparo Rodríguez, Dr. Enrique Rocha y Dr. Carlos Calles** por sus amplios conocimientos de investigación que me ayudaron a realizar este trabajo.

**Dr. Benjamín Ortiz, Dr. Willian Pech y Dr. Miguel Ibarra** por sus conocimientos varios de ingeniería que me ayudaron en el transcurso del estudio de mi grado.

**Dra. Daniela Cruz y Dr. Victor Martinez** por sus conocimientos varios de investigación y administración que me ayudaron en este grado.

Estoy profundamente agradecido con todos ustedes por su apoyo en las diferentes etapas de este grado. Gracias.



# Índice General

Índice General	I
Índice de Figuras	v
Índice de Tablas	VII
Índice de Algoritmos	IX
Resumen	XI
Abstract	XIII
Nomenclatura	xv
<b>1. Introducción</b>	<b>1</b>
1.1. Antecedentes . . . . .	1
1.2. Planteamiento del problema . . . . .	3
1.3. Justificación . . . . .	4
1.4. Estado del arte . . . . .	8
1.5. Objetivos . . . . .	10
<b>2. Marco Teórico</b>	<b>11</b>
2.1. Inteligencia artificial . . . . .	11
2.2. Machine learning . . . . .	12
2.2.1. Aprendizaje supervisado . . . . .	13
2.2.2. Aprendizaje no supervisado . . . . .	14
2.2.3. Aprendizaje por refuerzo . . . . .	14
2.3. Árbol de decisión . . . . .	15
2.3.1. Algoritmo de entrenamiento ID3 . . . . .	15
2.3.2. Algoritmo de entrenamiento C4.5 . . . . .	19
2.4. Procesamiento de imágenes . . . . .	21
2.4.1. Transformaciones de color . . . . .	21
2.4.1.1. Transformación RGB a escala de grises . . . . .	22
2.4.1.2. Transformación RGB a HSV . . . . .	22
2.4.2. Segmentación . . . . .	23
2.4.2.1. Binarización por umbral . . . . .	24
2.4.3. Histogramas . . . . .	25
2.5. FPGA . . . . .	26
2.5.1. Estructura de un FPGA . . . . .	26
2.5.2. Historia de los FPGAs . . . . .	28

2.5.3. Arquitectura general de un FPGA . . . . .	29
2.5.4. Componentes y características . . . . .	30
2.5.5. Lenguajes de descripción de Hardware (HDL) . . . . .	33
2.5.6. Implementación . . . . .	34
2.5.7. Ventajas de FPGA . . . . .	35
2.6. Procesamiento de imágenes en FPGA . . . . .	36
2.6.1. Ventajas de FPGA en procesamiento de imágenes . . . . .	39
<b>3. Desarrollo</b>	<b>43</b>
3.1. Metodología . . . . .	43
3.2. Sistema propuesto . . . . .	45
3.3. Maquinaria de flujo de naranja . . . . .	45
3.4. Flujo de fruta en maquinaria . . . . .	48
3.5. Sistema eléctrico/electrónico . . . . .	49
3.5.1. Elementos mecánicos . . . . .	50
3.5.2. Procesador de video . . . . .	52
3.5.3. Elementos auxiliares . . . . .	52
3.5.4. Elementos de control y medición . . . . .	52
3.6. Modelo AD de segmentación . . . . .	55
3.6.1. Captura de video . . . . .	56
3.6.2. Fotogramas de interés . . . . .	56
3.6.3. Captura de data set de pixeles . . . . .	57
3.7. Modelo AD clasificador de color . . . . .	60
3.7.1. Análisis de color . . . . .	60
3.7.2. Captura de data set de histogramas . . . . .	61
3.8. Modelo AD clasificador por tamaño . . . . .	62
3.9. Arquitectura genérica de árbol de decisión . . . . .	64
3.9.1. Generación de un HDL a partir de un AD . . . . .	66
3.10. Arquitectura del procesador de video . . . . .	67
3.11. Módulo Clasificador . . . . .	68
3.11.1. Conversión de espacios de colores y cálculo de ROI . . . . .	69
3.11.1.1. Módulo convertidor RGB a escala de grises . . . . .	69
3.11.1.2. Módulo convertidor RGB a HSV . . . . .	70
3.11.1.3. Módulo de cálculo de posición, ROI y fin de ROI . . . . .	72
3.11.2. Segmentación . . . . .	73
3.11.3. Cálculo de histogramas y área de segmentación . . . . .	74
3.11.3.1. Módulo de cálculo de área de segmentación . . . . .	74
3.11.4. Clasificación de color y tamaño . . . . .	76
3.12. Módulos de comunicación y gestión de respuesta . . . . .	77

<b>4. Resultados y discusión</b>	<b>81</b>
4.1. Árbol de Decisión de segmentación . . . . .	83
4.2. Árbol de Decisión de clasificación de color . . . . .	84
4.3. Clasificación de tamaño . . . . .	85
4.4. Recursos utilizados . . . . .	87
<b>5. Conclusiones</b>	<b>89</b>
<b>Bibliografía</b>	<b>91</b>



# Índice de Figuras

1.1. Condiciones poco probables [1]. . . . .	8
2.1. Ejemplo de la estructura de una máquina de aprendizaje. . . . .	13
2.2. Ejemplo de clasificación en 2 dimensiones. . . . .	14
2.3. Ejemplo de la estructura de un AD. . . . .	15
2.4. Inicio de AD de ejemplo. . . . .	19
2.5. Rama completa de AD de ejemplo. . . . .	20
2.6. Representación axial del espacio de color RGB [2]. . . . .	22
2.7. Conversión de imagen a color a escala de grises. . . . .	23
2.8. Pirámide de espacio de color HSV [3]. . . . .	24
2.9. Binarización de imágenes usando un umbral $t = 127$ . . . . .	25
2.10. Histograma de imagen 2.7b. . . . .	25
2.11. Arquitectura más simple de un PAL. . . . .	29
2.12. Arquitectura genérica de un FPGA. . . . .	30
2.13. Esquemas de arquitecturas de elementos internos de FPGA. . . . .	32
2.14. Esquema básico de un IOB. . . . .	33
2.15. Lectura de imagen por posiciones de pixel. . . . .	37
2.16. Lectura de imagen por posiciones de pixel. . . . .	39
2.17. Arquitectura en tipo pipeline. . . . .	41
3.1. Diagrama de proceso de metodología empleada. . . . .	44
3.2. Diagrama general del sistema. . . . .	46
3.3. Diagrama general de maquinaria para el sistema de separación automática de naranjas. . . . .	47
3.4. Diagrama de flujo de procesamiento de fruta en la maquinaria. . . . .	49
3.5. Diagrama eléctrico/electrónico general. . . . .	50
3.6. Diagrama de conexión de eleméntos mecánicos . . . . .	51
3.7. Configuracion de terminales en MCU . . . . .	54
3.8. Diagrama de flujo del programa del MCU y los protocolos de comunicación empleados. . . . .	55
3.9. Escena de cámara dentro de cabina y ROI definida. . . . .	57
3.10. Captura para el set de datos y obtención de la mascara. Difuminación por movimiento. . . . .	58
3.11. Proceso de captura de imágenes. . . . .	58
3.12. Diagrama de flujo de captura de dataset de pixeles. . . . .	59
3.13. Resultados del análisis para cuatro posibles escenarios de color . . . . .	61
3.14. Diagrama de flujo y ejemplo de captura de data set de histogramas. . . . .	63
3.15. Modelos de AD en Hardware. . . . .	65
3.16. Ejemplo de AD generado mediante SCHAD. . . . .	67
3.17. Esquema general de la arquitectura. . . . .	68
4.1. AD usado para clasificación de tamaño. . . . .	86



# Índice de Tablas

2.1. Data set falso de prueba. . . . .	16
2.2. Muestras que son positivas en característica "Tos". . . . .	17
2.3. Muestras que son negativas en característica "Tos". . . . .	18
2.4. Data set falso de prueba. . . . .	19
2.5. Data set falso de prueba. . . . .	20
2.6. Data set falso de prueba. . . . .	21
2.7. Descripción de tiempos de sincronización de video para resoluciones 720p y 1080p. . . . .	40
3.1. Características de motores . . . . .	51
4.1. Descripción de velocidades y tipos de set para cada clip de video. . . . .	81
4.2. Distribución de capturas por prueba para el data set de imágenes. . . . .	82
4.3. Distribución de data set de pixeles por prueba obtenidos y sus resultados de entrenamiento en WEKA. . . . .	83
4.4. Precisión obtenida por plataforma de segmentación con AD por prueba. . . . .	84
4.5. Descripción de data set de histogramas por prueba y sus resultados de entrenamiento en WEKA. . . . .	84
4.6. Precisión por computadora y FPGA de modelo AD de clasificación de color. . . . .	85
4.7. Distribución de data set de imágenes para clasificación por tamaño. . . . .	85
4.8. Distribución de data set de imágenes para clasificación por tamaño. . . . .	86
4.9. Distribución recursos utilizados del FPGA. . . . .	87



# Índice de Algoritmos

1.	Algoritmo para módulo de histogramas . . . . .	75
2.	Algoritmo para cálculo del área segmentada . . . . .	76
3.	Secuencia de respuesta . . . . .	79



# Resumen

## **Clasificador de naranjas por tamaño y color implementado en FPGA usando aprendizaje supervisado**

por

**Ismael Antonio Dávila Rodríguez**

Maestría en Ingeniería

Universidad Politécnica de Victoria, 2020

DR. MARCO AURELIO NUÑO MAGANDA, Director

DR. YAHIR HERNÁNDEZ MIER, Co-Director

Se presenta un sistema de clasificación de tamaño y color mediante una cámara implementado en FPGA, donde la escena captada por la cámara se transportan naranjas mediante una banda industrial. Se utilizaron arboles de decisión para la resolución de la segmentación de la fruta para una posterior clasificación de tamaño y color. Además se desarrolló un script que convierte un archivo de texto donde se describe el comportamiento de los arboles a un lenguaje VHDL sintetizable. Se realizaron pruebas con diferentes sets de prueba, en donde cada set contiene menos prueba que la siguiente en donde se obtiene como resultado una precisión del 97 % para la segmentación de la fruta en la escena, un 94 % para la clasificación de color y un 90 % para la clasificación de tamaño, donde se denota la eficiencia del modelo al entrenarlo con pocas muestras.



# Abstract

## Citrus fruit classifier by size and color FPGA implemented using supervised machine learning

by

**Ismael Antonio Dávila Rodríguez**

Master of Engineering

University Polytechnic of Victoria, 2020

DR. MARCO AURELIO NUÑO MAGANDA, Advisor

DR. YAHIR HERNÁNDEZ MIER, Co-advisor

A color and size fruit grading system is presented, implemented in FPGA where the scene from a camera citrus fruit is transported by an industrial conveyor belt. Decision Trees were used to solve fruit segmentation to then perform a size and color grading. Also a script was developed that converts a decision tree text file to VHDL language synthesizable. Was made with different test sets where each set has fewer samples than others, was as a result the accuracy obtained where 97 % for segmentation, 94 % for color grading and 90 % for size grading, where the efficiency is maintained using fewer samples for training.



# Nomenclatura

**AD** Árbol de decisión

**FAO** Food and Agriculture Organization

**ML** Machine Learning

**RNA** Red Neuronal Artificial

**ARFF** Attribute-Relation File Format

**FPGA** Field Programmable Gate Array

**LSB** Less Significant Bit

**MSB** Most Significant Bit

**SCHADD** Script Constructor HDL de Árbol De Desición

**CONACyT** Consejo Nacional de Ciencia y Tecnología



Nota con respecto al formato:

En los primeros capítulos hay una linea en blanco para separar los párrafos, pero hay secciones en donde esta linea desaparece (debe checar todo el documento). Especificamente: Sección 1.2, 2.2.2, 2.3.1, 2.5, 2.5.1, 3.1 (demasiado grande el listado)

\*\* Sugerencia:

O lo quita o lo deja, pero que sea PAREJO para todo el documento!!

Imagen no lleva acento, pero Imágenes si lo lleva !!  
(lo aprendí a chingadazos)

## Introducción

Con respecto a las imágenes y tablas

Le recomiendo utilizar `\begin{figure}[!tbh]`, lo cual forzara a las figuras a ir en la parte superior de la página, o en su defecto en la parte inferior

### 1.1 Antecedentes

Desde la antigüedad los cítricos han sido parte fundamental de los alimentos, teniendo su origen desde las regiones tropicales y subtropicales del sureste de Asia y el archipiélago Malayo y esparcirse a todo el mundo. Partiendo históricamente por la planta del cidro vista por primera vez en Europa en 300 a.C. el cuál se utilizaba para el injerto y generación de especies híbridas de plantas y frutas, esto siendo una particularidad de los cítricos, ya que presentan mutaciones con mucha frecuencia incluso entre miembros de la misma especie [4].

Desde el siglo XVI, los cítricos fueron ganando popularidad por toda Europa aunque desde el siglo VI se cultivaban, hasta que en el siglo XIX fué el naranjo amargo el más popular entre el cultivo.

En la época actual los mayores productores de cítricos son China, Brasil, Estados Unidos, España, India, México y Argentina. De acuerdo a la Organización de Agricultura y Alimentos (FAO por sus siglas en inglés), México es uno de los 5 mayores productores de cítricos en el mundo [5].

## **1. Introducción**

---

Actualmente, en el estado de Tamaulipas es el segundo productor de cítricos en México. En el 2011, la producción total fue de 500,000 toneladas aproximadamente [6].

A inicios de 1960, la visión por computadora iniciaba en las universidades de Estados Unidos, herramienta que hasta el día de hoy busca emular la visión de los humanos a través de computadoras para el proceso de interpretación de imágenes, las cuales pueden producir estímulos en procesos industriales de clasificación. Esta herramienta se ha utilizado en diferentes ámbitos industriales, incluyendo la alimentaria.

Hoy en día se cuenta con sistemas de visión en la industria alimentaria para la clasificación de frutas como los *cítricos*. Los cuales cuentan con bandas transportadoras las cuáles hacen pasar la fruta por el sistema clasificador, el sistema normalmente se comunica con un sistema de movimiento de actuadores que realiza la tarea de separación. Por lo general estos sistemas sirven para determinar el tamaño, color, maduración y calidad de la fruta. Ya que una maquinaria trabaja más rápido que un sistema manejado enteramente por humanos la producción puede ser elevada, abasteciendo la gran demanda mundial antes mencionada.

Sin embargo en México no se tienen tecnologías muy desarrolladas en el ámbito, contando solamente con tecnologías extranjeras y obsoletas. Por eso la necesidad de desarrollar un proyecto que mediante tecnologías actuales y nacionales, solventar la producción de cítricos.

En las secciones posteriores de éste capítulo se aborda una pequeña introducción teórica acerca de los sistemas de clasificación actuales y su problemática nacional. En el Capítulo 2 se muestran todos los conceptos técnicos referentes a la realización del proyecto para poder comprender el principio y naturaleza de la solución propuesta. En el Capítulo 3 se muestra el desarrollo de la plataforma general del sistema propuesto, donde se explica el sistema mecánico, el flujo de información y la arquitectura propuesta del sistema. En el Capítulo 4 se demuestran los resultados obtenidos y la eficiencia detrás de la naturaleza de la solución propuesta desde los AD. En el Capítulo 5, de encuentran las conclusiones

del uso de modelos de AD en un FPGA.

### 1.2 Planteamiento del problema

Las clasificadoras de fruta son aquellas capaces de ordenar al producto en diferentes grupos tales como separarlas por color, tamaño o maduración.

En México existen clasificadoras que pueden separar estas clases mencionadas. Generalmente la mayoría de las veces se utilizan clasificadores manuales o mecánicos, ya que no cuentan con tecnología electrónica actual por lo que presentan muchos inconvenientes, partiendo que la mayoría solamente desempeñan 1 sola clasificación a la vez. Existen de varios tipos de clasificadores:

✓ **Clasificadora manual.** Es literalmente un proceso artesanal en el cual un grupo de personas se encarga de seleccionar la fruta conforme a su criterio. Las mismas personas se encargan de llevar a cabo cada etapa en la selección de la fruta, desde la limpieza hasta el empacado, convirtiendo este proceso en largas jornadas de trabajo arduo y dependencia en un 100 % de la intervención del hombre. Algunos de los problemas que presenta es la necesidad de empleados y todas las responsabilidades humanas, fiscales y legales que esto conlleva, también, la velocidad de producción no es estable ya que depende del personal y el clima laboral, así como la precisión del criterio del personal que puede ser variada, ocasionando errores en la clasificación.

✓ **Clasificadora mecánica.** Este tipo de clasificadora comprende una estructura, que debido a su forma geométrica puede clasificar la fruta solamente por tamaño, ya que no cuenta con ningún dispositivo electrónico de procesamiento de datos. El movimiento de la fruta puede ser por gravedad o movida por motores. Algunos de los inconvenientes es que debido a la construcción de estos clasificadores, es imposible la degradación de la calidad de la fruta, debido a golpes y cortes en producidos en el proceso, así como también son propensas a atascamientos y generar más desperdicios en el proceso.

## 1. Introducción

---

✓ **Clasificadora automática.** En los últimos años ha aumentado el uso de tecnología en la selección de fruta, donde predominan los métodos de inspección ópticos (cámaras) el cuál es un tipo de inspección no destructiva, aplicado a diversas frutas como manzanas, plátanos, fresas, limones, duraznos, naranjas, peras, papayas, tomates, papas, melones, sandía, etc como se verá en las secciones posteriores. En la figura se puede ver una máquina que cuenta con la capacidad de inspección por cámaras asistido con visión por computadora. Entre los problemas mas frecuentes de estos clasificadores es su obsolescencia, ya que por lo regular sus componentes no son actualizables, tanto hardware como software. Debido a esto también las reparaciones son difíciles, ya que el hardware necesario para reparaciones es escaso al pasar del tiempo, además que la clasificación es de menor velocidad ya que existen tecnologías muy avanzadas hoy día.

Con el fin de mejorar el hardware existente en velocidad, usar un arreglo de compuertas lógicas programables (**FPGA** por sus siglas en inglés), parece una opción prometedora, ya que estos se destacan por su procesamiento en paralelo, y flexibilidad de construcción de hardware.

### 1.3 Justificación

De acuerdo a la Organización de Agricultura y Alimentos (**FAO** por sus siglas en inglés), México es uno del los 5 mayores productores de cítricos en el mundo [5]. Actualmente, en el estado de Tamaulipas es el segundo productor de cítricos en México. En el 2011, la producción total fue de 500,000 toneladas aproximadamente [6].

El tratamiento post-cosecha se ha convertido en una etapa esencial en el mercado de fruta fresca, para mantener la frescura de los cítricos y proveer al consumidor en las mejores condiciones posibles. Comúnmente, la calidad de la clasificación de cítricos es realizado por humanos, basado en inspección visual de un criterio externamente visible de la fruta, como su tamaño, forma y color. Esta inspección manual recae mucho en la experiencia del observador, para mantener la consistencia y uniformidad en la clasificación.

Algunos autores han reportado algunos sistemas de clasificación. En [7], las técnicas de clasificación son revisadas. El autor menciona las características mas usadas para identificar pudrición, maduración y clase de frutas, en modelos de machine learning (**ML**) usadas para la tarea. Dos grupos grandes de enfoques fueron identificados:

1. Sistemas enfocados a la identificación de múltiples frutas (Por lo que no se revisa la calidad de la misma), alimentada con sets de miles de imágenes de una serie de diferentes frutas
2. Sistemas enfocados en una fruta específica, usando un set de imágenes de un solo tipo de fruta para entrenar y probar el clasificador. Aun y cuando el primer enfoque es más general, el segundo es una mejor opción para máquinas clasificadoras, diseñadas para clasificar una sola fruta.

Para sistemas específicamente diseñadas para clasificación de cítricos, una evaluación de diferentes clasificadores automáticos para naranja es reportado en [8], donde 5 arboles de decisión (AD) diferentes, tres redes neuronales artificiales (RNA) y un clasificador por reglas fueron probados. En [9], un clasificador de tipo árbol de reglas fue propuesto para clasificar la maduración de las naranjas. En [10], un algoritmo de segmentación multiespectral basada en una estructura de datos de arboles cuadruples es presentada.

Sistemas para clasificación de frutas que no son cítricos también han sido propuestos. En [11], un clasificador Bayes y un clasificador de manzanas basado en el espacio de colores de Ohta es presentado. En [12], un método para reconocer las regiones de tallo o cáliz en manzanas es propuesto. Eliminación de fondo y segmentación de objetos por umbralización son usadas para aislar el objetivo. Después por características estadísticas, texturas y forma son extraídas de cada objeto segmentado y usado para el entrenamiento de los modelos: lineal discriminante, vecino más cercano (KNN), vecino más cercano difuso (KNNF), máquinas de soporte vectorial (SVM) y clasificador adaboost.

## 1. Introducción

---

En [13], un prototipo para clasificar fruta en base a las fecha de cosecha es propuesto. RNAs son entrenadas con el set de datos de 1800 frutas para obtener características como debilidad, tamaño, forma, intensidad y defectos.

Relacionados a sistemas de múltiples frutas, hay algunas aplicaciones en la literatura. En [14], una transformación curvelet para la clasificación de guayaba y limón es propuesto. Medidas de texturas basadas en la transformación curvelet como la energía, entropía, media y desviación estandar es usada para caracterizar textura de la superficie de la fruta. Modelos como SVM, y Redes Neuronales Artificiales Probabilísticas (RNAP) fueron entrenadas para distinguir entre frutas en buen estado y con defectos. En [15], un sistema para clasificar manzanas, frutas y naranjas es presentado. Utilizando una Transformación de Características Invariantes a Escalamiento (SIFT) fue usado para obtener características de las frutas presentadas, y un clasificador de tipo Bosque aleatorio fue aplicado de igual manera. En [16], un algoritmo KNN para clasificar frutas basado en su color, forma y tamaño es reportado. El enfoque propuesto fue aplicado para manzanas, limones, plátanos y fresas. En [17], descriptores de imágenes basadas en apariencia, color, textura, y forma fueron obtenidos y distinguidos de las técnicas de ML como SVM, LDA, árboles de clasificación, KNN, y ensambles de árboles fueron entrenados.

Recientemente, modelos de Deep Learning (DL) han sido intensivamente desarrolladas para la clasificación de problemas complejos. Esos modelos son mostrados para entender mejor las características en imágenes, y han sido usadas para afrontar con el rápido crecimiento de datos (big data). En [18], un clasificador basado en DL es propuesto. Algoritmos de DL requieren mucho tiempo entrenamiento de datos, una gran cantidad de datos de entrenamiento (set de datos), y un poder de procesamiento enorme. Generan una solución muy especializada en un dominio específico, donde la revaloración es necesaria para resolver problemas fuera del dominio.

El diseño de un sistema de visión para clasificación automática de cítricos es altamente deseada para reducir el costo de la fuerza laboral y problemas de mantener la consistencia y uniformidad

en la clasificación. Un clasificador automático de cítricos basada en visión de computadora afronta algunos problemas técnicos como, el procesamiento de imágenes en tiempo real, baja precisión de clasificación y alto costo computacional. Un FPGA es ideal para esta tarea, ya que provee de procesamiento paralelo. En el enfoque presentado en [19], una arquitectura para clasificación de cítricos es presentada. El pre-procesamiento requerido para la clasificación es implementada en hardware. La clasificación de cítricos fue basada en una umbralización global, convirtiendo al sistema sensible al ruido generado por los objetos en movimiento y un modelo de ML más robusto no fueron explorados.

Algunos algoritmos de clasificación para frutas como ANN, SVN requieren mucha cantidad de memoria para almacenar diferentes coeficientes y complejos cálculos no lineales. Los arboles de decisión (AD), son uno de los mejores algoritmos para implementación de hardware, ya que no requieren cálculos aritméticos, que son computacionalmente costosos, si no que solamente usa comparadores [20]. Los AD han sido aplicados exitosamente en algunos dominios, como reconocimiento de objetos [21], identificación de gas [22], y reconocimiento de cantar de aves [23].

La comparación entre los sistemas de clasificación es difícil por su larga variedad y combinatoria de sets de imágenes de frutas. Precisión, especificidad y sensibilidad de mediciones son a menudo reportadas en diversos trabajos, pero no hay una medida real para medir su desempeño en el area, ya que existen sets de imágenes que son capturadas en entornos estáticos.

En esta tesis se propone una metodología para llegar al desarrollo de un clasificador de naranjas basado en AD, usado para la segmentación de la fruta y su clasificación. El enfoque está dirigido para realizar una implementación sencilla en hardware, donde las arquitecturas de los modelos AD se generan por medio de software.

## 1. Introducción

---

### 1.4 Estado del arte

Por lo general los experimentos que se han realizado se usan fotografías (no video), de los cuales se extraen características, sin embargo las condiciones en las que se toman las fotografías son escenarios poco probables y poco prácticos para un problema real como lo es en un escenario industrial tal y como se ve en la figura 1.1 [1], donde la fruta está inmóvil en una escena fácil de segmentar.

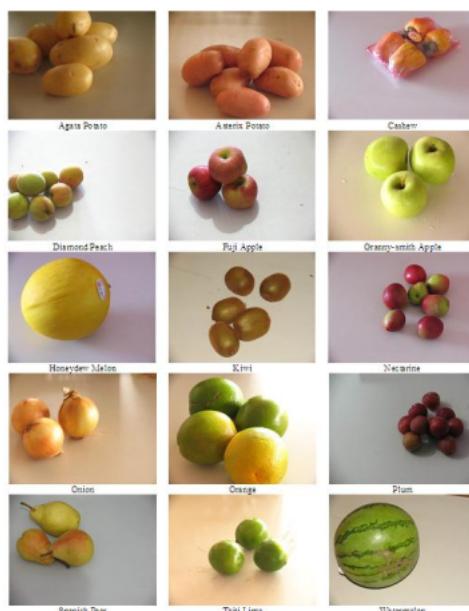


Figura 1.1. Condiciones poco probables [1].

El canal de colores Tono, Saturación y Valor (HSV) es pieza clave en la clasificación por color [8, 24, 25, 26, 27] ya que se obtiene mucha información de uno de los canales ya que tanto H como S no son muy afectados por la iluminación.

La clasificación por umbrales fijos es difícil de lograr ya que se necesita mucha experimentación y desarrollo [26, 19], por lo que la implementación de un clasificador es necesaria.

Para la extracción de características para saber el estado de la fruta se utilizan ciertas características, tal es el caso de obtener desde el espacio de colores HSV de la imagen a procesar para

obtener características sobre el color y la iluminación específica. De esto anterior obtener la media, y el histograma de los 3 canales correspondientes [8, 24], por otra parte el análisis es solamente para detectar que se aprecia en mal estado, pero no se clasifica el color.

El obtener características complicadas de computar no siempre es la mejor opción para un problema como este, ya que el obtener una precisión de 90 % es aceptable, sin embargo el tiempo de procesamiento o el desarrollo de un algoritmo sofisticado puede ser costoso en recursos y tiempo de desarrollo [14]. Sin embargo, existen sistemas similares, los cuales están montados en FPGA, donde la velocidad de computo se mejora hasta 100 veces [28] además de también poder ser implementados clasificadores por arboles de decisión [29].

El uso de diferentes clasificadores para la clasificación de naranjas están los perceptrones multicapa (analizando histogramas sobre el canal RGB) [30, 8], arboles de decisión, Random Forest [8, 31].

Por otra parte, en la mayoría de los casos se tiene un set de datos muy reducidos, es por eso la elección de preparar un hardware industrial como en [32, 33] para tomar video, y obtener un set de datos más grande, a cambio de un poco de tiempo de desarrollo para el procesamiento del mismo y obtener fotografías variadas.

Como se ha mencionado antes, un sistema como este, montado en un FPGA tiene resultados interesantes dado el poder de computo y la capacidad de funcionar en tiempo real [19], sin embargo el no utilizar algún tipo de memoria para almacenar las imágenes para un post-procesamiento, así como también el no utilizar clasificadores supervisados.

**Atiende a las indicaciones que te haga el Dr. Yahir y Said con respecto a esta sección.**

### 1.5 Objetivos

#### Objetivo general

- Desarrollar una simulación de hardware en FPGA capaz de clasificar naranjas con una precisión del 90 % con una rapidez de 1 naranja por segundo. **Suponiendo hardware ideal - Cual es tu velocidad de procesamiento limite?**  
**Con el hardware actual - Cual es tu velocidad de procesamiento limite?**

#### Objetivos específicos

- Simulación en FPGA de un clasificador por AD para segmentación y clasificación de color. **y tamaño?**
- Simulación en FPGA de un procesador de video extractor de características.
- Simulación de módulo de comunicación externo para hardware auxiliar de control de proceso.

Deben ir ligados con tu metodología, generalmente se redactan en singular "Simular", pero simular es ambiguo, deberian decir: Desarrollar, Implementar, Evaluar, Probar, etc (ojo, excluiria Diseñar una maquina separadora, pero incluiria un objetivo que involucre el hardware existente con tu propuesta de diseño)

- \* Obtener un conjunto de videos para el entrenamiento de modelos de ....
- \* Implementar las interfaces de comunicacion con una maquina existente
- \* Generar un conjunto de modelos de aprendizaje basados en arboles de decision para segmentar y clasificar frutas por tamaño y color
- \* Diseñar, codificar y evaluar un programa para generar la representacion hardware de un arbol de decision y su integracion es el modulo de vision
- \* Diseñar y probar el modulo de vision en un dispositivo FPGA
- \* Integrar la solucion propuesta con un prototipo maquina separadora propiedad de la empresa ....

# 2

## Marco Teórico

En esta sección se abordan temas relacionados al trabajo realizado.

### 2.1 Inteligencia artificial

La inteligencia artificial en ciencia, es una rama que estudia desde un punto de vista de la ingeniería, la inteligencia en elementos artificiales, y que estos sean capaz de demostrar un comportamiento inteligente. Pretende construir sistemas que presenten un comportamiento que se pueda decir que es inteligente.

La inteligencia es un concepto difícil de definir, por lo que los consensos científicos no han establecido completamente su definición e identificación de un caso de inteligencia, osease, decir cuando algo es inteligente o no, sea artificial o no.

En el año de 1950, el padre de la computación Alan Turing, publicó un articulo relacionado a la computación y la inteligencia llamado “Maquinaria computacional e inteligencia”, donde

## 2. Marco Teórico

---

él argumentaba que siempre y cuando una máquina puede actuar como un humano, entonces es inteligente [34]. En el artículo se propuso una prueba llamada Test de Turing, que permitiría afirmar si una máquina es inteligente o no. El test es compuesto por 2 partes (individuos o cosas) donde se comunicarán por un medio informático, la primera parte debe ser un humano, y la segunda puede ser una máquina o un humano, la prueba consiste en que la primera parte defina si la segunda es una máquina o un humano. Si la primera parte identifica a una máquina como humano entonces esta máquina es considerada inteligente. Hoy en día con los avances de la tecnología para que una máquina sea identificada como inteligente debe tener como características el reconocimiento del lenguaje natural, el razonamiento, el aprendizaje o la representación del conocimiento.

El **aprendizaje automático** se refiere a la capacidad de la máquina para aprender cosas nuevas, de otro modo, no será capaz de adaptarse al medio o condiciones que exige el entorno. Las líneas de investigación que se revisan hoy en día se busca hacer que las máquinas capten generalizaciones a partir de ejemplos sacados del entorno. Como ejemplo, un niño pequeño al aprender a caminar, tiene caer muchas veces antes de calcular de manera exacta las instrucciones motrices para dar un paso, hasta lograr dar pasos con soltura.

### 2.2 Machine learning

Machine learning es una rama de la inteligencia artificial. Se define como la ciencia de hacer que sistemas de cómputo, aprendan y actúen como los humanos, mejorar su aprendizaje sobre el tiempo de forma autónoma, alimentándolos con datos e información que significan interacciones del mundo real [35].

Una máquina de aprendizaje es un modelo matemático o algoritmo capaz de predecir algún comportamiento a partir de datos que caracterizan el sistema (figura 2.1), los datos por lo general son clusters de características de que definen el comportamiento de un sistema, cada característica se administra en el modelo como una entrada. Después el modelo calcula una predicción y esta es

su salida.

Mismo tamaño de letra para toda la figura...

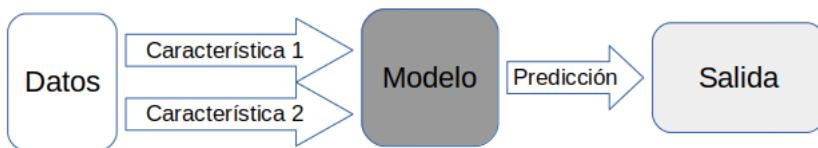


Figura 2.1. Ejemplo de la estructura de una máquina de aprendizaje.

Las máquinas de aprendizaje sirven para clasificar o realizar una regresión. En la figura 2.2 se puede ver el ejemplo de un set de datos, donde los ejes representan características de un sistema, por ejemplo: temperatura y humedad. Supongamos que se selecciona el 90 % de los datos para entrenar el modelo, el cual dependiendo del modelo se entrenará con uno u otro algoritmo. La linea punteada dibuja el resultado del entrenamiento el cual define la separación entre clases. Cuando un nuevo dato no conocido por el sistema ingresa al modelo, este será capaz de predecir la clase, entre más difícil sea de separar los datos se necesitan diferentes modelos.

Existe 3 tipos de máquinas de aprendizaje: máquinas de aprendizaje supervisado, máquinas de aprendizaje no supervisado, máquinas de aprendizaje por refuerzo [35].

### 2.2.1 Aprendizaje supervisado

En ellas la entrada y la salida deseada son administradas para el entrenamiento del modelo [36]. Los datos de entrada y salida son etiquetadas para la clasificación y con ellas se forma la base del entrenamiento para el procesamiento futuro de los datos tal y como se explicó en la sección anterior.

Algunos modelos de este tipo son el Árbol de decisión (AD), regresión lineal, regresión logística, máquinas de soporte vectorial (SVM), modelo Naive Bayes, algoritmo K-vecinos más cercanos, algoritmo de bosque aleatorio, etc.

## 2. Marco Teórico

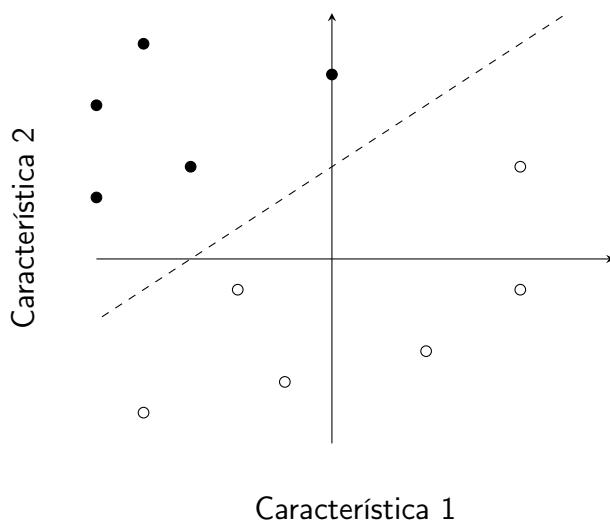


Figura 2.2. Ejemplo de clasificación en 2 dimensiones.

### 2.2.2 Aprendizaje no supervisado

En este tipo de aprendizaje, los datos de entrenamiento, no están etiquetados, por lo que el modelo aprende a distinguir instancias de otras. Los modelos más usados en este tipo de aprendizaje pueden ser K-Vecinos cercanos, Análisis de componentes principales, Apriori o Eclat.

La utilización de estos modelos, algunas veces son útiles para hacer clusters, es decir, agrupamiento de múltiples modelos de aprendizaje para una mejor clasificación; estos realizan tareas que facilitan, a un modelo siguiente, como por ejemplo: la detección de anomalías para ignorar muestras, visualización de datos para el entendimiento de los datos, reducción de dimensiones para una clasificación más sencilla, etc.

### 2.2.3 Aprendizaje por refuerzo

En estos modelos, la instancia del mismo es llamado agente y son entrenados mediante recompensas/castigos por alguna conducta, la estrategia usual es que entrenen por si mismos, y deben definir cuál es la mejor estrategia que les dé mayores recompensas con el tiempo, definiendo políticas comportamentales al paso del entrenamiento enfrentando diversas situaciones.

se ha logrado implementar estos tipos de modelos en robots para que aprendan a hablar.

Reforzar lo anterior con alguna referencia

## 2.3 Árbol de decisión

Los Arboles de decisión (AD) es una herramienta poderosa y popular para la clasificación y predicción [37]. Los AD son estructuras de arboles enraizados, con hojas representando clasificaciones y nodos representando pruebas con características que definen las clasificaciones.

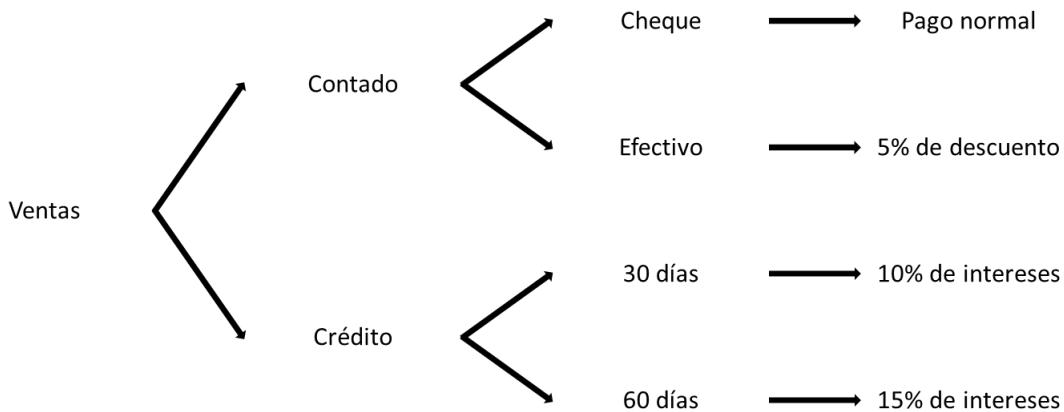


Figura 2.3. Ejemplo de la estructura de un AD.

Un árbol puede ser entrenado dividiendo la fuente del set de datos de entrenamiento en otros pequeños basados en un atributo. Algunos algoritmos populares para entrenar un AD son CART, ID3, C4.5 y Bosques aleatorios. Los AD clasifican las instancias clasificándolas por el árbol desde la raíz hasta algún nodo hoja, lo que proporciona la clasificación de la instancia [38]. El algoritmo utilizado en este trabajo es el **algoritmo C4.5** el cuál es un derivado del algoritmo IDE3.

### 2.3.1 Algoritmo de entrenamiento ID<sub>3</sub>

**Insertar referencias al algoritmo original (ver correo)**

El algoritmo de Dicotomización Iterativa 3 (en inglés Iterative Dichotomiser 3) inventado por Ross Quinlan es llamado así debido que de manera iterativa, divide las características de cada grupo de muestras. El algoritmo, realiza un árbol de decisión partiendo de las características más importantes, y separando de lo más general a lo más particular, por lo que cada iteración se genera un nuevo nodo u hoja dictaminando la clasificación o si se debe realizar nuevamente una separación entre muestras.

Para la compresión de este algoritmo veamos la tabla 2.1 la cual contiene un **data set falso** donde

Agregar cita

## 2. Marco Teórico

---

se determina si una persona padece una gripe común.

conjunto de datos hipotético original

Tabla 2.1. ~~Data set falso~~ de prueba.

ID	Tos	Escurrimiento Nasal	Fiebre	Gripa
1	No	No	Si	No
2	Si	No	Si	Si
3	Si	No	No	No
4	Si	Si	No	Si
5	No	Si	No	No
6	Si	No	Si	Si
7	No	Si	Si	Si
8	No	No	No	No
9	No	Si	Si	Si
10	No	Si	No	No
11	Si	Si	No	Si
12	No	Si	No	No
13	Si	No	No	No
14	Si	Si	Si	Si
15	Si	Si	No	Si

El algoritmo selecciona la mejor característica en cada paso cuando se construye un AD, para eso hay que encontrar la **ganancia** de cada característica, en este caso Tos, Escurrimiento nasal y Fiebre. La ganancia describe reducción en la entropía, es decir, mide que tan bien separa una característica entre los dos grupos de cada clase. Recordemos que la **entropía** mide el desorden del ~~data set~~, será cercana a 0 cuando todos los miembros de la misma clase comparten la misma cantidad de muestras.

La entropia y la ganancia son calculadas de la siguiente manera:

$$E(S) = - \sum_{i=1}^{i=n} p_i \log_2(p_i)$$

$$G(S, A) = E(S) - \sum \frac{|S_v|}{|S|} E(S_v)$$

Donde  $E$  es la entropía,  $S$  el data set,  $n$  el número de clases,  $p_i$  la probabilidad de la clase  $i$ ,  $G$  es la ganancia,  $|S_v|$  el número de muestras que tiene el valor  $v$ ,  $|S|$  el número total de muestras (o filas) del data set completo.

El algoritmo realiza el siguiente orden: Primero, se calcula la ganancia de cada característica. En segundo se consideran que todas las filas no pertenecen a la misma clase, dividiendo el data set en partes, usando la característica con mayor ganancia. En tercero, se instancia un nodo con la característica de mayor ganancia. En cuarto, si todas las filas pertenecen a la misma clase, se deja el nodo actual como una hoja del AD. Y por último, se repite para las siguientes características de manera iterativa hasta quedar con nodos hoja en el AD.

Para verlo de una mejor manera, calculemos la entropía de nuestro data set:

$$E(S) = - \left[ \frac{8}{15} \log_2 \left( \frac{8}{15} \right) \right] - \left[ \frac{7}{15} \log_2 \left( \frac{7}{15} \right) \right] = 0,4837 + 0,5131 = 0,9968$$

$n=8$ ,

Donde 8 muestras del data set son afirmativos en gripe (del primer término), 7 los que no (del segundo término), y 15 las muestras totales. Tenemos que la entropía se acerca a 1, por lo que quiere decir que tenemos casi igual de repartidas de las 2 clases.

Para calcular la ganancia de la característica "Tos" primero se divide el data set por la cantidad de clases que existan en la característica, en este caso Si y No, correspondiendo la tabla 2.2 y 2.3

Tabla 2.2. Muestras que son positivas en característica "Tos".

ID	Tos	Escurrimiento Nasal	Fiebre	Gripa
2	Si	No	Si	Si
3	Si	No	No	No
4	Si	Si	No	Si
6	Si	No	Si	Si
11	Si	Si	No	Si
13	Si	No	No	No
14	Si	Si	Si	Si
15	Si	Si	No	Si

Al separarlas calculamos la entropía por cada parte del set de la siguiente manera.

$$|S| = 15$$

## 2. Marco Teórico

---

Tabla 2.3. Muestras que son negativas en característica "Tos".

ID	Tos	Escurrimiento Nasal	Fiebre	Gripa
1	No	No	Si	No
5	No	Si	No	No
7	No	Si	Si	Si
8	No	No	No	No
9	No	Si	Si	Si
10	No	Si	No	No
12	No	Si	No	No

Para  $v = \text{Si}$ ,  $|S_{Si}| = 8$

$$E(S_{Si}) = - \left[ \frac{6}{8} \log_2 \left( \frac{6}{8} \right) \right] - \left[ \frac{2}{8} \log_2 \left( \frac{2}{8} \right) \right] = 0,3112 + 0,5 = 0,81$$

Para  $v = \text{No}$ ,  $|S_{No}| = 7$

$$E(S_{No}) = - \left[ \frac{2}{7} \log_2 \left( \frac{2}{7} \right) \right] - \left[ \frac{5}{7} \log_2 \left( \frac{5}{7} \right) \right] = 0,5163 + 0,3467 = 0,863$$

Y se calcula la ganancia de la característica:

$$G(S, Tos) = E(S) - \left[ \left( \frac{|S_{Si}|}{|S|} E(S_{Si}) \right) \right] - \left[ \left( \frac{|S_{No}|}{|S|} E(S_{No}) \right) \right]$$

$$G(S, Tos) = 0,9968 - \left[ \frac{8}{15}(0,81) \right] - \left[ \frac{7}{15}(0,863) \right] = 0,9968 - 0,432 - 0,402 = 0,1628$$

Para las otras características se obtiene:

$$G(S, Esc\_Nas) = 0,0794$$

$$G(S, Fiebre) = \mathbf{0.1859}$$

Ahora como vemos la característica "Fiebre" es la que separa mejor las clases y tomamos como nodo raíz a esa característica con lo que se tendría el árbol como se muestra en la figura 2.4.



Figura 2.4. Inicio de AD de ejemplo.

Para continuar, ahora separamos S en dos subsets de Si y No, las cuales serán nuestras nuevas S, para cada rama, veamos el caso del subset Si en la tabla 2.4, donde ahora se tendrá que calcular cual de las dos características restantes es mejor para definir la clase, por lo que se calculan de nuevo las ganancias, obteniendo la misma ganancia para ambos casos ya que hay igual muestras repartidas.

Tabla 2.4. Data set falso de prueba.

ID	Tos	Escurrimiento Nasal	Fiebre	Gripa
1	No	No	Si	No
2	Si	No	Si	Si
6	Si	No	Si	Si
7	No	Si	Si	Si
9	No	Si	Si	Si
14	Si	Si	Si	Si

Se repite el algoritmo para cada rama del árbol con su correspondiente subset sucesivamente, en el caso para la rama afirmativa se muestra la continuación del árbol en la figura 2.5, donde los nodos hoja, se muestran con óvalos.

### 2.3.2 Algoritmo de entrenamiento C4.5

El algoritmo de entrenamiento C4.5 también llamado J48, es un algoritmo para construir ADs. Es el sucesor del algoritmo ID3 creado por J. Ross Quinlan, y es uno de los algoritmos para ADs más populares [39]. En este algoritmo tiene la ventaja de aceptar valores numéricos, veamos la tabla

## 2. Marco Teórico

---

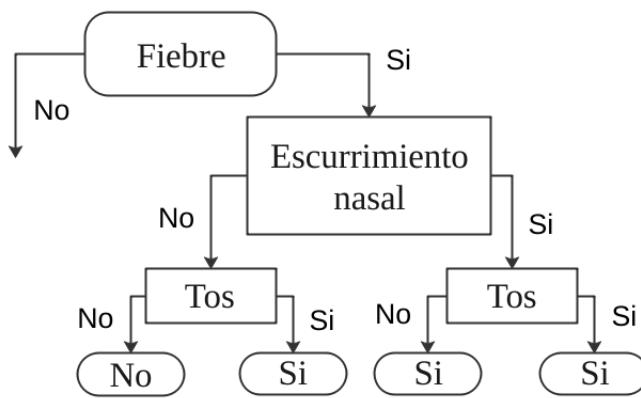


Figura 2.5. Rama completa de AD de ejemplo.

2.5 en donde la columna de "Fiebre" se ha cambiado por "Temperatura" en valores continuos. Para la realización del AD, siempre que sean valores nominales, se aplicará el algoritmo ID3, sin embargo para valores continuos realiza el siguiente procedimiento:

conjunto de datos hipotético original

Tabla 2.5. Data set falso de prueba.

ID	Tos	Escurrimiento Nasal	Temperatura(°C)	Gripa
1	No	No	40.1	No
2	Si	No	39.6	Si
3	Si	No	36.1	No
4	Si	Si	35.9	Si
5	No	Si	36.2	No
6	Si	No	39.7	Si
7	No	Si	38.8	Si
8	No	No	37.0	No
9	No	Si	40.2	Si
10	No	Si	36.4	No
11	Si	Si	36.7	Si
12	No	Si	36.1	No
13	Si	No	36.2	No
14	Si	Si	39.8	Si
15	Si	Si	36.7	Si

Para convertir los valores continuos en unos nominales, primeramente se ordenan los valores de menor a mayor, tal y como se ve en la tabla 2.6, y se tiene que encontrar el umbral con mayor ganancia

de manera iterativa, es decir, primeramente se prueba con la condicional que  $Temperatura > 35,9$ , después que  $Temperatura > 36,1$  y así sucesivamente, hasta encontrar el umbral con mayor ganancia, entonces ese es el elegido como condicional para el nodo. Teniendo esto en cuenta, se realiza el procedimiento del algoritmo ID3 normalmente.

Conjunto de datos hipotético reordenado

Tabla 2.6. Data set falso de prueba.

ID	Tos	Escurrimiento Nasal	Temperatura(°C)	Gripa
4	Si	Si	35.9	Si
3	Si	No	36.1	No
12	No	Si	36.1	No
5	No	Si	36.2	No
13	Si	No	36.2	No
10	No	Si	36.4	No
15	Si	Si	36.7	Si
11	Si	Si	36.7	Si
8	No	No	37.0	No
7	No	Si	38.8	Si
2	Si	No	39.6	Si
6	Si	No	39.7	Si
14	Si	Si	39.8	Si
1	No	No	40.1	No
9	No	Si	40.2	Si

## 2.4 Procesamiento de imágenes

Algunas operaciones de procesamiento de imágenes son necesarias para la implementación del sistema propuesto en este trabajo.

### 2.4.1 Transformaciones de color

En procesamiento de imágenes, es común que las imágenes estén en formato rojo-verde-azul (espacio de colores RGB), y se necesitan ser transformados en otros espacios de colores.

## 2. Marco Teórico

---

### 2.4.1.1. Transformación RGB a escala de grises

Una operación común para reducir los recursos computacionales en procesamiento de imágenes consiste en convertir una imagen RGB en una codificación diferencial como  $YC_bC_r$ , donde  $Y$  es la componente de luminancia,  $C_b$  y  $C_r$  son las componentes diferenciales de azul y rojo [40].

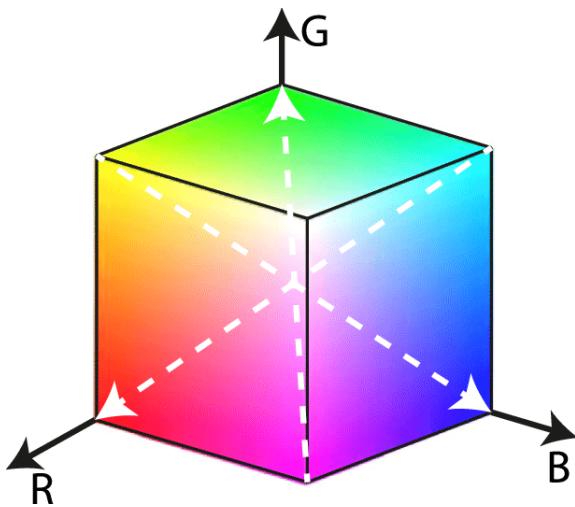


Figura 2.6. Representación axial del espacio de color RGB [2].

En los formatos digitales ITU-R BT-709, la luminancia puede ser calculado usando la ecuación 2.1, donde R, G y B son los componentes del espacio de colores RGB.

$$Y'_{709} = 0,2126R' + 0,7152G' + 0,722B' \quad (2.1)$$

Esta componente de luminancia corresponde a una imagen en escala de grises tal y como se muestra en la figura 2.7b para la imagen a color 2.7a.

### 2.4.1.2. Transformación RGB a HSV

Otro espacio de colores importante es Tono - Saturación - Valor (HSV). La particularidad de este espacio de colores es el campo del procesamiento de imágenes posible en él. Es una transformación no lineal del espacio RGB.



Figura 2.7. Conversión de imagen a color a escala de grises.

Las componentes R, G y B de un objeto en una imagen digital son todas correlacionadas a la cantidad de luz que refleja el objeto. Por otra parte las descripciones en términos de esos componentes puede hacer la discriminación del objeto complicada. En estas situaciones donde la descripción del color juega un rol importante, la descripción en términos del Tono (H), saturación (S) y brillo (V) es comúnmente usada.

Esos valores son usados como ejes de coordenadas que proyectan el espacio RGB a lo largo de las diagonales blanco a negro (figura 2.6), resulta en un hexágono que forma la parte superior de una pirámide del espacio HSV (figura 2.8). H, es indicado como un ángulo alrededor del eje vertical. El color rojo es obtenido cuando  $H=0^\circ$  o  $H=360^\circ$ , el verde cuando  $H = 120^\circ$ , y así sucesivamente [41].

## 2.4.2 Segmentación

La segmentación es la capacidad de separar o capturar una porción de una imagen, la cual contiene información de interés en el fotograma y que generalmente se extrae para procesamientos posteriores y obtener un mayor enfoque en el área. La comprenden una amplia gama de procesamientos, sin embargo en este veremos solo algunas formas utilizadas en este trabajo.

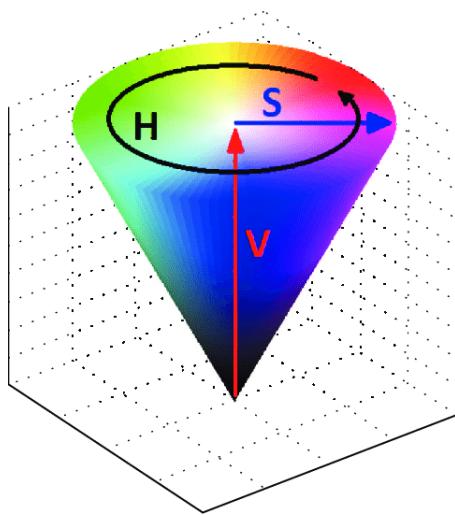


Figura 2.8. Pirámide de espacio de color HSV [3].

#### 2.4.2.1. Binarización por umbral

Se trata de un método de segmentación que se rige por pixeles individuales. De una imagen por lo general en escala de grises se analiza pixel por pixel utilizando la ecuación 2.2, donde  $T$  es la función de binarización (por sus siglas en inglés de la palabra *Thresholding*),  $p$  como el valor del pixel y  $t$  como el umbral (por sus siglas en inglés de la palabra *threshold*).

$$T(p, t) = \begin{cases} 0 & \text{si } p < t \\ 255 & \text{si } p \geq t \end{cases} \quad (2.2)$$

El efecto de este método de segmentación da un realce de todos aquellos pixeles mayores al valor del umbral, tal y como se ve en la figura 2.9a. Por otra parte la condición de la ecuación 2.2 puede ser invertida, obteniendo la ecuación 2.3, donde el efecto se muestra en la figura 2.9b, la cual es una inversión de colores de la imagen 2.9a.

$$T_{inv}(p, t) = \begin{cases} 255 & \text{si } p < t \\ 0 & \text{si } p \geq t \end{cases} \quad (2.3)$$

El cómputo de este método de segmentación es muy fácil de implementar en hardware ya que se trata de un comparador simple.



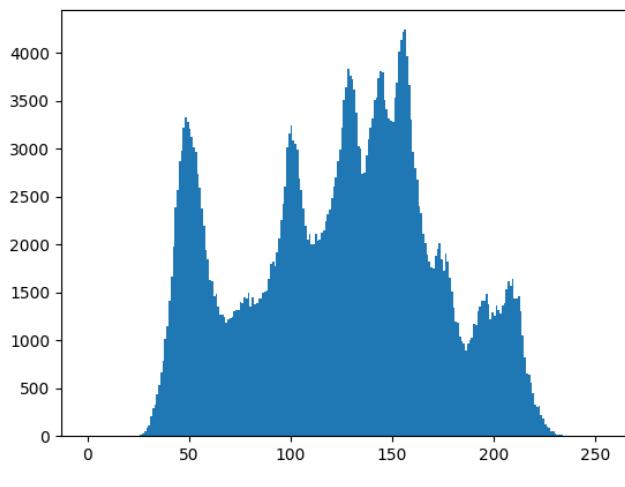
Figura 2.9. Binarización de imágenes usando un umbral  $t = 127$

### 2.4.3 Histogramas

Los histogramas en procesamiento de imágenes, representan de manera gráfica la distribución de un canal de color, es decir, en que tantas ocasiones los pixeles han tomado cierto valor, por ejemplo, en la figura 2.10, se encuentra el histograma de la figura 2.7b, y que su distribución se encuentra entre el rango (30,200) aproximadamente, por lo que en muy pocas ocasiones, se encuentran valores de pixeles en esta imagen fuera de este rango.

Porque?

\* Depende de la resolución



- \* El histograma de LENA de 512x512 se parece al histograma de LENA de 64x64?
- \* La figura 2.10 es de una imagen en escala de gris. Para una imagen a color, hay 3 histogramas (para cada canal) o hay un histograma separado por BINS (debes incluir esto en esta sección)

Figura 2.10. Histograma de imagen 2.7b.

## 2. Marco Teórico

No es relevante!!

### 2.5 FPGA

Los...

Arreglos de Compuertas Lógicas Programables llamadas en inglés Field Programmable Gate Array (llamadas FPGA de aquí en adelante), es un circuito integrado que está fabricado con silicio el cuál tiene la apariencia como cualquier otro circuito integrado. A diferencia de un IC convencional, los FPGAs son reprogramables, es decir, su funcionamiento interno o su propósito, puede ser modificado. Estos utilizan bloques pre-establecidos dispuestos de tal forma que la conexión entre ellos permiten ser variable, logrando así que el funcionamiento final pueda ser modificado dependiendo de como es que el usuario quiera esas conexiones.

Los FPGAs en el mundo de las simulaciones y en el prototipado, tiene un gran impacto desde hace algunas décadas. Los FPGAs las compañías de electrónicos las usan para poder generar su hardware digital, poderlo probar en una etapa temprana como los puertos de comunicación y los puertos de desarrollo que son usados durante todo el prototipado. Debido a su flexibilidad son muy requeridos en el área de la electrónica industrial.

FPGA => Masculino, singular  
FPGAs => Masculino, plural

#### 2.5.1 Estructura de un FPGA

El usuario puede configurar los FPGAs para implementar funciones que el mismo desee, sin tener que utilizar ICs adicionales para cambiar el comportamiento del circuito y por ende no utilizar otros dispositivos físicos que se pueden convertir en basura (circuitos de prueba que fallen por ejemplo). Por otra parte, el usuario deberá desarrollar ingeniería de computo digital en un software especificado para cada fabricante, este computo digital se desarrolla en archivos de texto en Lenguaje de Descripción de Hardware (llamado HDL a partir de aquí), el cual el software compilará y dará de salida un archivo tipo bitstream, el cual contiene toda la información necesaria que se utilizará para hacer las conexiones internas en el FPGA . Por ende, según la explicación anterior los FPGAs son completamente flexibles ya que se puede cambiar su configuración cada que el usuario lo requiera.

## Utilización en la industria

Los FPGAs en la industria han sido pilar importante en el desarrollo de nuevos semiconductores (procesadores, Application-Specific-Integrated-Circuits o ASICs, etc.), ya que los FPGAs combinan lo mejor de los ASICs y de los sistemas en los que se basan procesadores ya que estos ofrecen frecuencias de reloj administradas por hardware, sin requerir altos volúmenes de requerimientos para poder fabricar un ASIC.

A diferencia de un procesador convencional que funcionan con una serie de instrucciones o pasos, los FPGAs funcionan en paralelo ya que están construidos solo por bloques lógicos o en más bajo nivel, solo están fabricados por compuertas lógicas las cuales están dispuestas de tal modo que se pueden fabricar series de bloques que ejecutan diversas tareas al mismo tiempo tal es la función de un procesador con más de un núcleo. Por otra parte, en los procesadores modernos existen hasta 8 o más núcleos, sin embargo en un FPGA se pueden hacer tantos bloques sean posibles físicamente dentro del FPGA que equivaldrían a núcleos que se dedican específicamente a una tarea en particular y no a tareas de distintos tipos lo cual es lo que mismamente desarrolla un procesador convencional de computadora.

La diferencia entre un ASIC y un FPGA es básicamente que un ASIC no puede cambiar su funcionamiento. Un ASIC como lo dice su nombre es un IC que tiene una aplicación en específico (por lo que su estructura no es flexible), su arquitectura es definida y finita. Los ASIC son ICs que son personalizables en su construcción, pero una vez manufacturados no pueden cambiar su arquitectura, por lo que para probar y hacer diseños de ASICs con otros ASICs es algo que los FPGAs han desplazado por completo. En un principio los FPGAs eran de baja potencia de computación, para desarrollar grandes volúmenes de diseño, pero hoy en día un FPGA puede llegar a tener un total de 500 millones de operaciones por segundo (un reloj de 500MHz) sin ningún problema y son capaces de almacenar diseños mucho más grandes comparándolos con los FPGAs de hace un par de décadas.

## 2. Marco Teórico

---

### 2.5.2 Historia de los FPGAs

En 1984 Xilinx creó el primer modelo de FPGA, el modelo XC2046. Ellos no les llamaban como tal hasta que Actel popularizó el término en 1988. Desde entonces los fabricantes de FPGAs han incrementado las prestaciones increíblemente, tales **som su** capacidad, velocidad costo y consumo de energía.

Cuando los FPGAs aparecieron, eran mucho más lentos, caros y pequeños (en capacidad) que cualquier ASIC en el mercado. Los FPGAs prosperaron debido a que en la fabricación de ASICs había mucha ingeniería de por medio que podía ser reducida usando un FPGA en el campo del diseño de circuitos digitales. Esta ingeniería de por medio resultaba algo cara, ya que se tenía que investigar, contratar personal capacitado y además algunas máquinas para la manufactura. Algunos fabricantes empezaron a usar los FPGAs, y se dieron cuenta que a cierto volumen de ventas, resultaba mucho más rentable usarlos en sus proyectos ya que se gastaban muchos recursos en la ingeniería. Cuando otros fabricantes empezaron a ver estos beneficios el volumen de ventas de ASICs ya no era justificable, volviendo así una tendencia el uso de FPGAs en el desarrollo de semiconductores.

PAL (Programmable Array Logic) los cuales eran programables con memorias EPROM. Estos fueron introducidos a principios de los años 80's. Algunos otros les llamaban PLA (Programmable Logic Array).

Estos consistían en dos niveles lógicos de estructura, en la figura 2.11 se puede ver una estructura simple de un PAL, en el fondo podemos ver que están marcadas las entradas, en la parte izquierda se encuentra un arreglo AND programable, que produce operaciones AND de cualquier combinación incluyendo sus inversos. El bloque azul de la derecha corresponde a una compuerta de operación OR que completa la operación combinacional de la función lógica programada en el PAL. De esta manera es fácil de implementar máquinas de estados finitos en una versión muy primitiva.

Los PAL fueron muy eficientes desde un punto de vista de manufactura, ya que su estructura es bastante parecida a la de un arreglo de memorias EPROM. Gracias a esta arquitectura en un tiempo

se tuvieron ideas nuevas sobre como expandir las memorias, y expandir los procesos de sus líneas de producción. El problema principal de esta arquitectura es bastante evidente, ya que para expandir las funcionalidades de un PAL, este necesita convertirse en arreglos de PALs, dando como resultado una complicada programación por la cantidad exageradas de puntos que se debían de programar. Los arreglos de ANDs terminaban siendo demasiado grandes que la manufacturarlo el tamaño del dispositivo rebasaba a lo convencional.

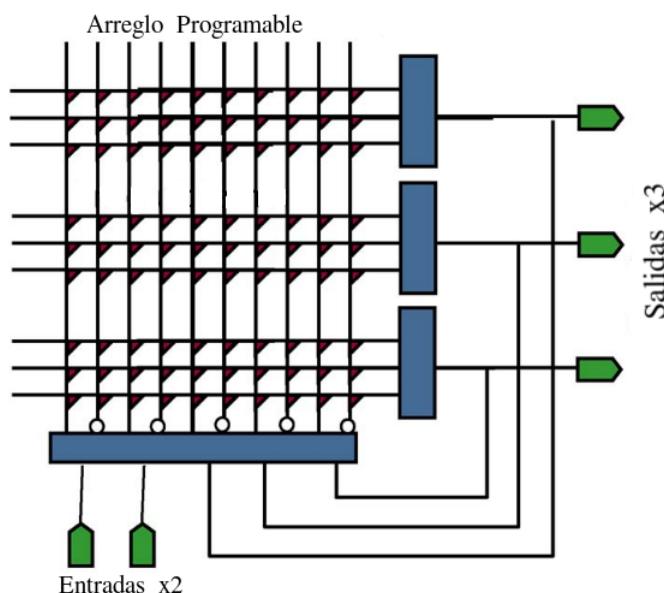


Figura 2.11. Arquitectura ~~mas simple~~ de un PAL.

### 2.5.3 Arquitectura general de un FPGA

La innovación de los FPGA fue la eliminación de la matriz de compuertas AND que se programaban, reemplazándolo por memorias de duración que fueron distribuidas entre cada nodo del arreglo de unidades lógicas. La arquitectura de un FPGA consiste en un arreglo de bloques programable los cuales se interconectan con unos switches programables entre unión de los bloques, los bloques están conectados entre si con estos switches, haciendo así el hardware reprogramable. El diagrama de la arquitectura de un FPGA convencional se puede ver en la figura 2.12, donde los

## 2. Marco Teórico

---

bloques circulares son los switches programables, los bloques cuadrados son unidades lógicas y los bloques que se encuentran en las orillas son las entradas y salidas.

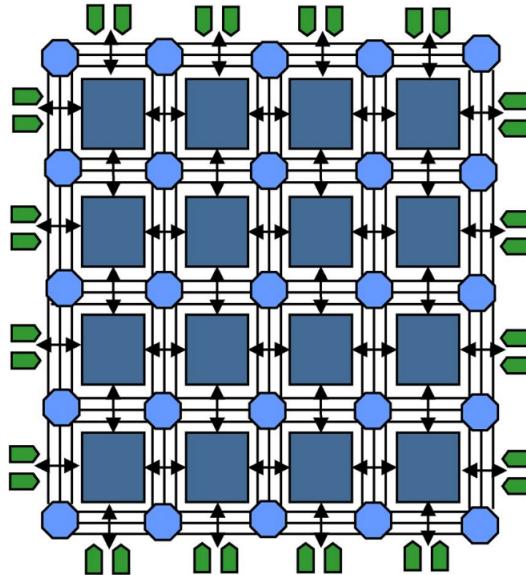


Figura 2.12. Arquitectura genérica de un FPGA.

Un FPGA no cuenta con un procesador el cual siga una serie de instrucciones (software). El programador es quien diseña un circuito digital y es posible configurar un FPGA tan simple como una compuerta AND de 2 entradas o un complejo procesador **multi núcleo** para usarse como DSP. En [42] se puede ver que los FPGAs son idóneos para los DSPs, por lo que para aplicaciones de alta velocidad son una ventaja imprescindible.

### 2.5.4 Componentes y características

En general, hoy en día hay muchas otras formas de arquitectura de FPGAs sin embargo el en el modelo **general** la arquitectura de un FPGA está compuesto de los siguientes elementos:

## CLB: Configurable-Logic-Block

La arquitectura de un CLB (figura 2.13a) de un FPGA lo componen varios módulos lógicos más pequeños. Por lo general estos contienen LUTs, los cuales son una clase de memoria ROM (Read-Only-Memory), que son volátiles, ya que una vez programado perderá su valor cuando el FPGA sea desenergizado. Además cada CLB cuenta con un bloque de interconexiones, la cual tiene como tarea comunicar los bloques lógicos.

✓ **Slices.** Algunos FPGAs en sus CLBs están divididos por slices (rebanadas) como se puede ver en la figura 2.13c, un slice esta formado por dos módulos lógicos (o más dependiendo de su arquitectura), y entre ellos mismos la lógica combinacional se asocia dependiendo del problema en particular que se trate.

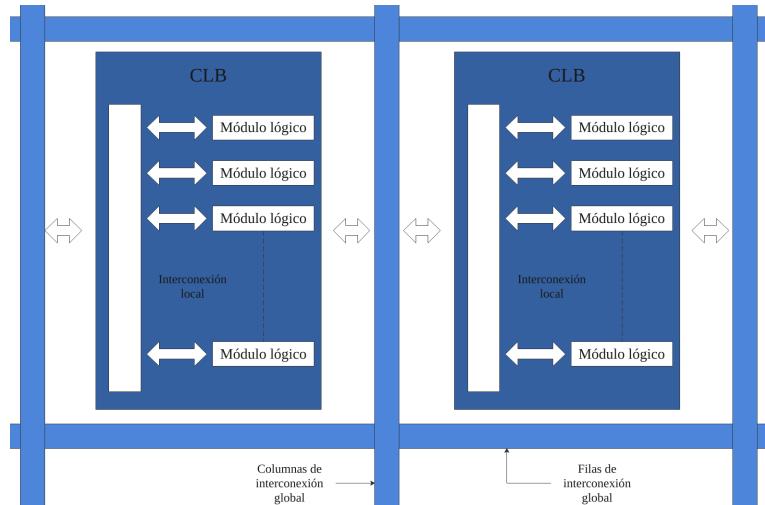
✓ **Módulo lógico.** Un módulo lógico se configura para desarrollar circuitos de lógica combinacional por medio de las LUTs, o también como registros (localidades de memoria). En si, es un modelo reducido de memoria y que en esencia una LUT realiza el mismo trabajo que una PAL.

✓ **LUTs.** Convencionalmente, la arquitectura de una LUT (figura 2.13d) consiste en un arreglo de memorias, se tiene n como el numero de entradas, y una parte se comporta como un circuito combinacional convencional.  cursiva

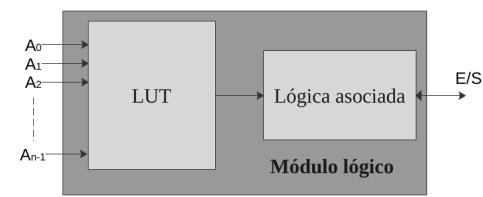
✓ **IOB: Input/Output Block.** Un IOB es un bloque de entrada o salida (figura 2.14). A diferencia de un microcontrolador, la mayoría de los pines de un FPGA puede ser configurado de diferentes maneras. En algunas tarjetas de desarrollo algunos pines están reservados, por lo regular son entradas o salidas especializadas para ciertos periféricos, o señales de alta frecuencia, tal es el caso de las señales de reloj que utilizan los FPGAs . Los microcontroladores la mayoría de las veces cuentan con diferentes modalidades, tal es el caso que una misma terminal puede servir como un

## 2. Marco Teórico

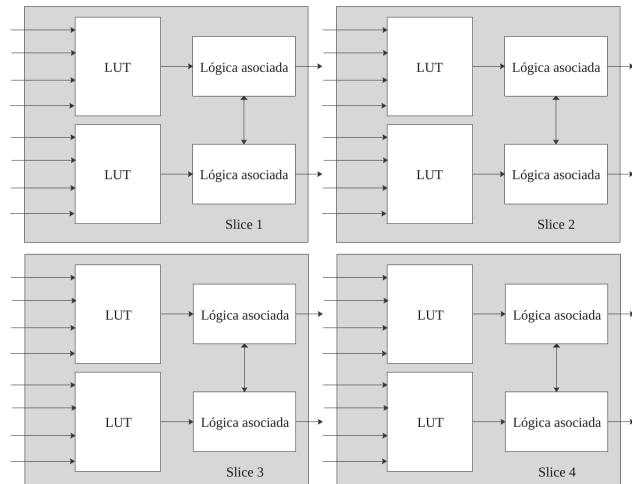
---



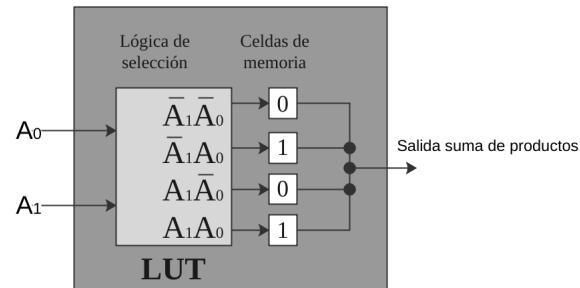
(a) Arquitectura general de un bloque Configurable Lógico (CLB).



(b) Arquitectura general de un módulo Lógico.



(c) Arquitectura general de un CLB formado con slices.



(d) Arquitectura general de una LUT.

Figura 2.13. Esquemas de arquitecturas de elementos internos de FPGA.

convertidor analógico digital (ADC), o incluso ser una salida o entrada digital, sin embargo esta función no puede ejecutarse al mismo tiempo.

implementarse

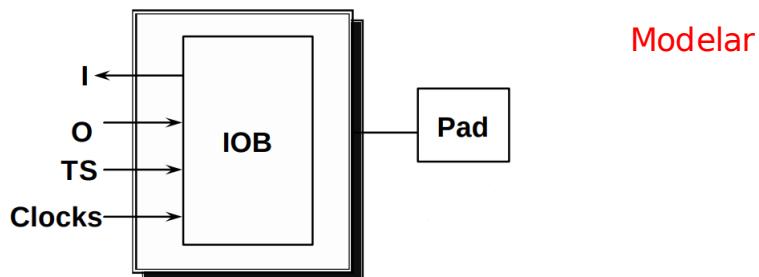


Figura 2.14. Esquema básico de un IOB.

modelar

## 2.5.5 Lenguajes de descripción de Hardware (HDL)

Como hemos visto en secciones anteriores, los FPGAs son dispositivos flexibles, ya vimos porque ~~lo son~~, sin embargo ¿cómo es que una computadora hace todo lo posible por ~~resolver~~ los circuitos que deben ~~realizarse~~ a bajo nivel?. Existe software que se usa para ~~calcular~~ todas estas cosas, pero de todo ello depende un código escrito en un HDL. El código es interpretado por un compilador y posteriormente se resuelve toda la lógica y circuitos a bajo nivel hasta convertirse e un bitstream, el cual es un sistema de archivos que sirve para programar las celdas programables dentro del FPGA .

**Existen varios HDLs, pero los 3 mas utilizados son:**

~~Existen unos cuantos HDLs, de entre los que más destacan son:~~

✓ **VHDL.** Este HDL es un estándar en Europa. Su estilo tiene un parecido al lenguaje C, sin embargo muchos de los operadores son bastante distintos. Se suelen definir los bloques con las palabras reservadas "begin" y "end". Por lo general a los principiantes les resulta más fácil entender un código en este HDL.

✓ **Verilog.** Este HDL es un estándar en América. Su estilo tiene aún mas parecido al lenguaje C ya que tiene prácticamente los mismos operadores. Se suelen declarar bloques independientes con el

## 2. Marco Teórico

---

operador “@”. Las sentencias igualmente se parecen mucho al lenguaje C.

✓ **System Verilog.** Este HDL es bastante peculiar, y como se puede intuir es una clase de “extensión” del HDL Verilog. Este HDL suele no ser sintetizable, se usa en su mayoría para declarar simulaciones de bloques temporizados. Se utiliza mucho este HDL para la depuración de grandes proyectos. Sin duda es una herramienta muy buena de alto nivel que ayuda a depurar código rápidamente.

~~Cada HDL tiene su estilo y ninguno es mejor que otro. Estos 3 son de los más usados, sin embargo existen algunos más. Por estandarización mundial, la mayoría de los desarrolladores utilizan estos 3 HDLs.~~

Recordemos que este código describe circuitos y hardware, por lo que no es nada parecido a la programación de software. El flujo de datos no es secuencial, por lo que es un error común en los principiantes tratarlo como tal. Los ciclos “for” por ejemplo, no describen una secuencia, si no un patrón de repetición para módulos, o dicho de otra manera, reproducir circuitos iguales como un arreglo.

### 2.5.6 Implementación

La implementación de un diseño básicamente consiste en 3 pasos:

✓ **Síntesis.** Este proceso compila todo el circuito lógico diseñado en un HDL, en donde se revisa la sintaxis del mismo. Después se revisa si el HDL puede ser sintetizable, es decir, que se pueda construir con Hardware. Existen algunos elementos que se pueden describir con hardware pero no pueden ser construidos físicamente, como por ejemplo pueden ser pulsos temporizados sin base de alguna señal de reloj.

✓ **Mapeo.** Al ya estar resuelto el circuito lógico puede que algunos circuitos no quepan en CLBs únicos, por lo que el circuito se divide en sub-bloques y trata de encajar con la arquitectura del FPGA. Esto se hace en una computadora con un compilador especial, esta compilación es distinta

de cada fabricante. Una vez realizado el mapeo, el usuario por medio del software puede ser capaz de realizar análisis de tiempos de señales de mapeo estático.

✓ **Posicionamiento y Ruteo.** En esta etapa se necesita una lista de red (NetList) la cual en ella, están descritos cada entrada y salida del módulo de mayor nivel y lo conecta a cada pin físico del FPGA. Una vez reestructurado todo el circuito en bloques pequeños que pueden recrearse en CLBs, el compilador busca la manera de colocar todos estos bloques de tal manera que ahorre espacio entre conexiones de bloques. Debido a que la arquitectura varía entre fabricantes e incluso entre modelos del mismo fabricante pero con modelos diferentes de FPGAs, el ruteo es específico para cada modelo, por lo que se necesita un compilador diferente para saber al final como obtener conexiones idóneas entre bloques o poder ahorrar espacio para no sufrir un sobrecargo de Hardware. Al final de este paso se obtiene un bitstream.

### 2.5.7 Ventajas de FPGA

~~Como hemos visto en las secciones anteriores~~, un FPGA tiene una estructura ~~única~~ y flexible. Gracias a estas características los FPGAs ofrecen muchos beneficios, como los son:

✓ **Rendimiento.** Dada la arquitectura de un FPGA y que puede ejecutar tareas de forma paralela, los FPGAs son candidatos excelentes para computar operaciones que realizan los procesadores digitales de señales (por sus siglas en inglés Digital-Signal-Processing llamados DSPs de aquí en adelante). Con un FPGA se pueden controlar entradas y salida a nivel de hardware propiamente, y esto ofrece tiempos de respuesta mucho más rápidos. En algunos osciloscopios se utilizan FPGAs en sus sistemas para realizar cálculos a velocidades increíbles.

✓ **Tiempo en llegar al mercado.** Gracias a las ventajas de diseño que ofrece un FPGA el experto puede probar una idea o un concepto sobre la construcción de un circuito ~~digital~~, y verificarlo por su propia cuenta, sin tener que llegar a fabricar o manufacturar el IC. Por otra parte, diseños se pueden refinar o mejorar en un FPGA y todo esto, ya que usando un FPGA se ahorra el tiempo de

## 2. Marco Teórico

---

manufactura de un IC que está en prueba.

✓ **Precio.** De primera mano comparar el precio de costo de fabricación de un ASIC contra la de un FPGA comercial es **total mente** diferente. Para un ASIC se requieren varios ingenieros tanto para desarrollar la arquitectura interna como su manufactura. Las soluciones basadas en FPGAs son bastante simples ya que en una primera parte solamente se necesita la ingeniería detrás de la arquitectura del IC, sin tener que gastar aun en la manufactura. Una vez refinado por completo el modelo puede ser manufacturado, esto puede ahorrar bastante en scrap realizado en pruebas.

✓ **Fiabilidad.** Los circuitos realizados internamente en un FPGA se pueden considerar que son implementaciones seguras, ya que no existen sistemas operativos de por medio, además el paralelismo ofrecen un mayor rendimiento en cualquier tarea.

✓ **Mantenimiento a largo plazo.** Los FPGAs son personalizables y actualizables en una implementación. No requieren el precio y el tiempo en rediseñar un nuevo chip. Algunas bloques con el tiempo pueden requerir una actualización, tal es el caso de los protocolos de comunicación los cuales con el tiempo requieren ciertos cambios, ya sean cambios de “**timing**”, o de transferencia en la secuencia. Un sistema modelado mediante un FPGA puede ser continuamente actualizable, mientras que una basada en ASICs no lo puede ser debido a que una gran parte de la electrónica requeriría ser rediseñada y verificada.

## 2.6 Procesamiento de imágenes en FPGA

El procesamiento de imágenes en un FPGA es bastante distinto de entornos computacionales, y dependiendo del procesamiento que se va a realizar se necesita cierto hardware. En imágenes de alta resolución (mayores a 720p), un **FPGA** en la mayoría de los casos, no se puede almacenar toda la imagen, debido a la cantidad necesaria de recursos que esta necesita tal y como se pudiera hacer en una computadora, ya que por lo general la imagen debe estar en formato RAW (es decir, la imagen

36      **No es del todo cierto. Que nosotros no hayamos explorado el uso de las memorias RAMS, no significa que NO SE PUEDA almacenar. Mas bien no es conveniente para no complicar el circuito (deberías considerar esta tangente), pero si seria necesario para algoritmos mas complejos**

sin ningún tipo de formato de compresión), por lo que por lo general la imagen es almacenada en alguna memoria conectada al FPGA, de ahí ser leída y procesada.

### Lectura de una imagen Detalles de sincronización de video

#### Suponiendo que se tiene una

Veamos la imagen de un solo canal que tiene una resolución de  $10 \times 10$  tal y como se muestra en la figura 2.15 donde se muestra la imagen de un carácter "u", en ella podemos ver los ejes de coordenadas  $x$  y  $y$  donde el origen está en la esquina superior izquierda de la imagen, por lo que se considera un avance positivo hacia la derecha en el eje horizontal, y hacia abajo en el eje vertical. La imagen leída en un FPGA es pixel a pixel, empezando por el pixel "0" mostrado en la figura en la posición  $(0,0)$ , en valores de 1 byte por pixel, posteriormente los píxeles hacia la derecha, y al acabar la fila, se empieza por la posición  $(0,1)$  en el pixel 10, así sucesivamente hasta terminar la imagen en el pixel 99 en la posición  $(9,9)$ .

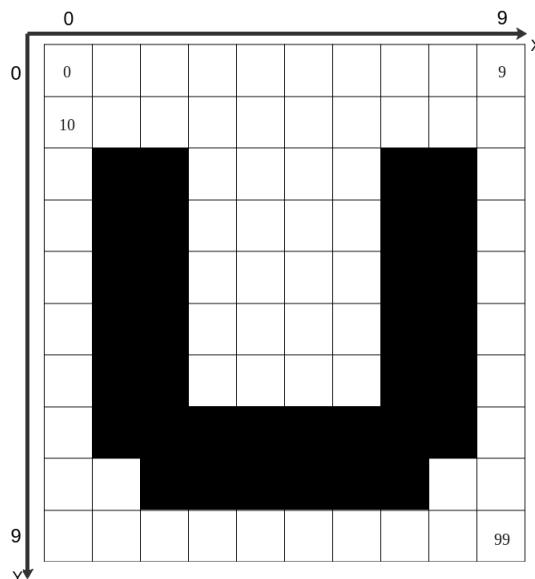


Figura 2.15. Lectura de imagen por posiciones de pixel.

La lectura de imágenes en FPGA puede obtenerse de las siguientes fuentes:

1. **Lectura desde memorias externas.** Para este caso, los datos de la imagen son leídos de una memoria, mediante buses de dirección, lectura y/o escritura, con lo que se leen bloques

## 2. Marco Teórico

---

de memoria y por consecuencia y se pueden dar dos casos en particular:

✓ **Imagen comprimida.** Existen diversos tipos de compresión, tales como PNG, JPG, JPEG, etc. Algunos con perdidas de datos al comprimir, y otras que no, en este caso la información está codificada y es necesario construir un decodificador, para poder obtener la imagen pixel a pixel.

✓ **Imagen sin comprimir (RAW).** En este caso la imagen es leída pixel a pixel byte por byte contenido en la memoria, por lo que no se necesitan pasos intermedios.

### Transferencia

2. Lectura en señales de video. La transferencia de imágenes digitales puede realizarse por medio de un conjunto de señales de sincronización, este formato es actualmente usado por protocolos como HDMI, DisplayPort, etc. En la figura 2.16, se denotan las señales de sincronización y algunas áreas, la primer área es la que compone la imagen en si, puede ser de varios canales, usualmente tres para RGB, del lado derecho tenemos a HBlank, el cual es una porción de la imagen con pixeles blancos, esta área es utilizada solamente para que se realicen pequeños cálculos en los procesadores de video, al igual que VBlank, solo que por la parte inferior de la imagen. Cabe resaltar que estas áreas no son impresas en pantalla. Las señales de sincronización se pueden apreciar en la figura 2.16 las cuales son:

✓ **Clk Video(Reloj de video).** Este es una señal de reloj que define la transferencia de un pixel nuevo.

✓ **Vsync (Sincronización Vertical).** Esta señal es un pulso al final de cada fotograma y representa cuando tiene que ser restaurado el contador de pixeles verticales del procesador de video. En ella se presentan tiempos de Back/Front porch y del pulso en si, estas tienen duraciones estándar y están en base al reloj de video.

✓ **Hsync (Sincronización horizontal).** Esta señal es un pulso al final de cada fila del fotograma y representa cuando tiene que ser restaurado el contador de pixeles horizontales del procesador de video. Así como Vsync, también se tienen tiempos estándar.

✓ **Video Activo.** Esta señal se mantiene en bajo cuando está en las áreas blank, y en alto

cuando son pixeles del fotograma a transmitir.

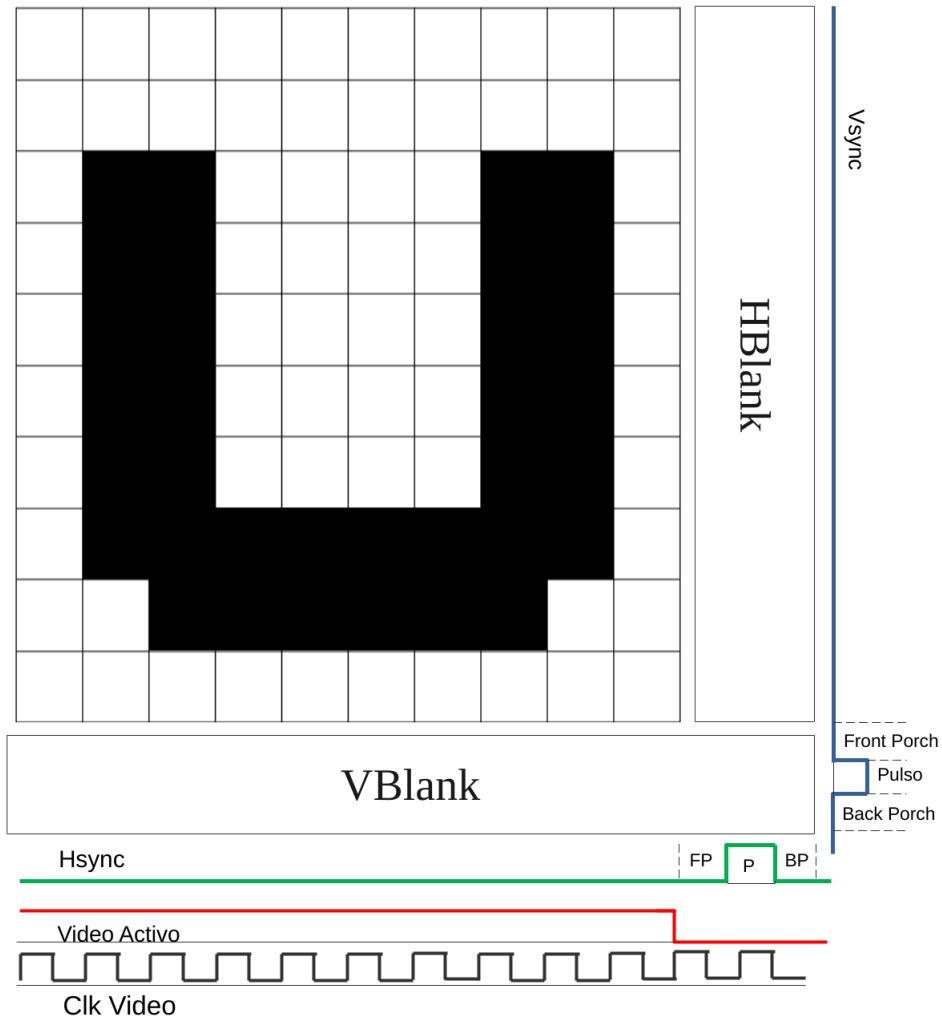


Figura 2.16. Lectura de imagen por posiciones de pixel.

Este es el tipo de lectura que ha sido utilizado en este trabajo, ya que por medio de una cámara con puerto HDMI se puede lograr una lectura y una velocidad de procesamiento muy alta. En la tabla 2.7 se denotan los tiempos standar para cada señal, obtenida de [43].

### ~~los para~~ 2.6.1 Ventajas de ~~FPGA~~ ~~procesamiento de imágenes~~

El uso de diversas tecnologías en el procesamiento de imágenes depende de su aplicación y contexto, sin embargo se puede dar una opinión acerca de las ventajas comunes que se pueden

## 2. Marco Teórico

Tabla 2.7. Descripción de tiempos de sincronización de video para resoluciones 720p y 1080p.

	<i>Caracteristica</i>	<b>720p 60Hz</b>	<b>1080p 60Hz</b>
Tiempos Horizontales	<i>Clk video (Hz)</i>	74.25	148.5
	<i>Pixeles activos</i>	1280	1920
	<i>Front Porch</i>	110	88
	<i>Pulso</i>	40	44
	<i>Back Porch</i>	220	148
Tiempos Verticales	<i>Pixeles totales</i>	1650	2200
	<i>Pixeles activos</i>	720	1080
	<i>Front Porch</i>	5	4
	<i>Pulso</i>	5	5
	<i>Back Porch</i>	20	36
	<i>Pixeles totales</i>	750	1125

encontrar en implementaciones hechas en FPGA. **Basicamente hay dos ventajas:**  
A) Procesamiento de imágenes en tiempo real  
B) Inclusion de diversos grados de paralelismo

### Procesamiento de imágenes en tiempo real

Un sistema de tiempo real es uno donde la respuesta debida a un evento debe ocurrir en un tiempo específico, de otra manera el sistema es considerado fallido según [44], en el contexto de procesamiento de imágenes entonces, se obtiene una imagen, se procesa para obtener ciertos datos, y con ello tomar el control de cierta actividad. Generalmente estos sistemas son usados para inspección o control de un proceso. Existen dos tipos de sistemas en tiempo real, el sistema duro y el sistema suave. El primero se refiere a que la muestra en proceso debe terminar antes de obtener la siguiente muestra, si esto no ocurre entonces se dice que el sistema falla. Por otra parte el segundo, se refiere a que si la siguiente muestra llega y aún no se ha terminado de procesar la anterior entonces el rendimiento del sistema baja, sin embargo sigue funcionando. En el caso de este trabajo es un sistema duro, ya que se debe ser procesada la fruta por cada muestra. Cuando se requiere que el procesamiento se realice en un periodo muy corto (dependiendo del sistema y su complejidad) es entonces que es requerido implementarse en hardware, para realizar una aceleración del procesamiento.

The Real time image processing is always in high demand for many applications used in security system, remote sensing, manufacturing process and multimedia, those require to have high performance.

Real time image processing systems need to have high performance because of requirement of handling huge image data and computationally intensive algorithms. This problem can be attempted using a heterogeneous platform consisting of an FPGA chip and a DSP processor.

The merits of FPGA chips are its flexible logic and high speed concurrence executions.

Hence real time image processing systems with heterogeneous platform can assumed to exhibit real time performance

nota pagina anterior

~~Paralelismo~~

\*\* Ver articulo que te inclui en el correo. Describir de manera breve los tipos de paralelismo (entre ellos el Pipeline)

En principio cada parte de un algoritmo debería ser corrido por un procesador dedicado, sin embargo esto en un sistema por computadora es poco aplicable. Aunque las computadoras de hoy manejen un procesador multinucleo, muchos de estos núcleos se utilizan para tareas dedicadas por el sistema operativo. Por otra parte en un FPGA debido a su flexibilidad puede ser implementado un procesador pequeño por cada paso del algoritmo, creando una estructura de tipo pipeline. El procesamiento con un sistema de tipo pipeline se puede ver en la figura 2.17, en donde todos los procesadores corren al mismo tiempo, y que, aunque sea un proceso secuencial, todos los procesadores trabajan a la vez, que al contrario del paralelismo, que el procesador realice cada tarea por separado.

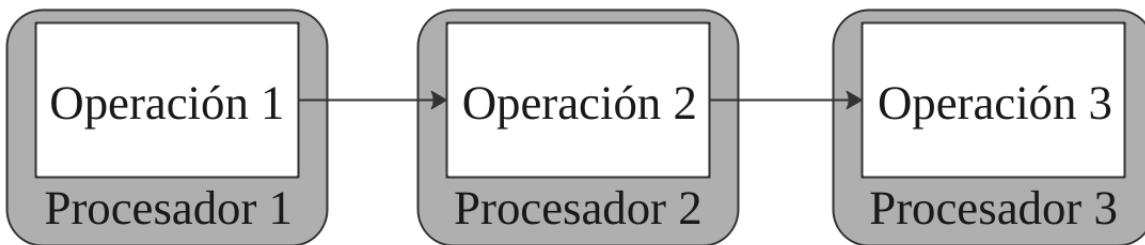


Figura 2.17. Arquitectura en tipo pipeline.

El paralelismo de la arquitectura en pipeline para procesamiento de imágenes funciona de la siguiente manera: En una primera instancia el procesador 1, realiza la operación 1, entonces está realizando una operación a una imagen. Al terminar la operación, el procesador 2 recibe los datos y realiza la operación 2, mientras el procesador 1 está recibiendo la siguiente imagen y realizando la operación 1 a la segunda imagen, así es repetido sucesivamente hasta que el procesador 3 procesa la primera imagen, mientras el primero recibe la tercera imagen.



# 3

Sugiero en adelante nombrar:

Dataset => Conjunto de datos

Train dataset => Conjunto de entrenamiento

Test => Conjunto de prueba

Testeo => Prueba

clips => Video

Conjunto de datos principal (A las imágenes de las naranjas)

Conjunto de datos de pixeles

Conjunto de datos de histogramas

Conjunto de datos de

Desarrollo

En este capítulo se muestran los componentes empleados y la solución para el proyecto. Se empieza hablando por la idea general, y el flujo de la solución ya que el proyecto está compuesto de diferentes temas, tales como ingeniería mecánica, eléctrica, electrónica y computacional. Después se explican particularmente cada sección que compone el proyecto, para el entendimiento de cada una de las consideraciones de diseño tomadas.

## 3.1 Metodología

Para este trabajo se siguió la metodología mostrada en la figura 3.1, donde se muestran los productos de cada fase, dejando como pendiente la última de ellas.

El orden de las fases se decidió a partir de lo que la fase anterior necesita para continuar. A continuación es explicada cada fase:

**Fase 1. Captura de video.** En el se obtienen clips de video en donde la escena es fruta pasando frente a ella. A partir de los clips se obtiene un data set de imágenes, y el algoritmo de segmentación se genera el modelo para la

de la fruta.

- \* Debe utilizar el verbo generar cuando implique una transformación a los datos de entrada (del conjunto de imágenes hacia el conjunto de pixeles, por ejemplo)
- \* Debe utilizar el verbo obtener cuando solo implique lectura de datos

### 3. Desarrollo

generan

**Fase 2. Generación de Data set.** Del ~~data set~~ de imágenes, se obtienen tres nuevos ~~data sets~~, uno de pixeles para la segmentación, uno de histogramas para la clasificación de color y otro de fruta por tamaño para la clasificación por tamaño.

**Fase 3. Entrenamiento de ADs.** Con ~~los data sets~~, se generan ADs para segmentación, clasificación de color y tamaño, por medio de la herramienta WEKA.

**Fase 4. Creación de HDL base.** Se crea todo el HDL necesario para el manejo de señales de video, para la extracción de características y el procesamiento de imágenes, además, se dejan los huecos que clasifican cada parte. \* Vale la pena aclarar que un HDL es un modelo de hardware que ...

**Fase 5. Script Constructor HDL de Árbol de Decisión (SCHADD).** Se genera un script de python que convierte un AD de weka (texto) en HDL (VHDL) implementable.

**Fase 6. Prueba en FPGA.** Prueba del modelo completo, donde una imagen en señales de video es procesada por el modelo, y se obtiene la clasificación por tamaño y color.

**Fase 7. Prueba en marcha con actuadores.** Prueba de la máquina completa puesta en marcha. La fase no se pudo realizar por problemas externos al proyecto. Decirlo tal cual. Por causa de la pandemia no fue posible probar esta integración en la empresa utilizar una fuente mas grande

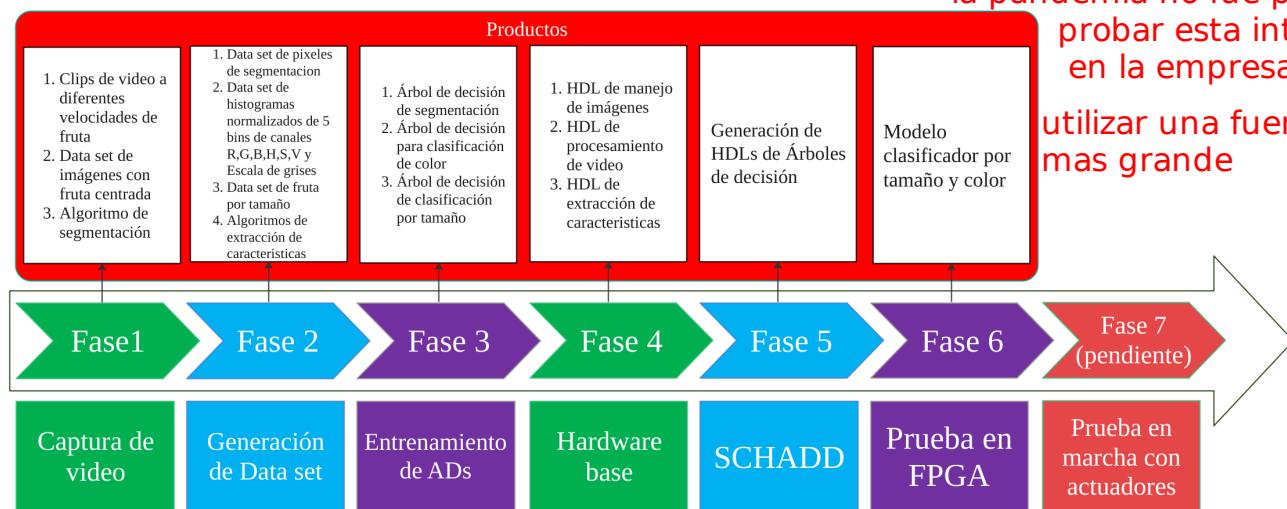


Figura 3.1. Diagrama de proceso de metodología empleada.

Yo creo que puede hacer esta figura mucho mas grande. Sugiero ponerla en vertical, algo asi:

Fase1      Captura de Video      Productos  
(Flecha hacia abajo)  
Fase 2      Generacion de....

## 3.2 Sistema propuesto

La idea general del trabajo se puede apreciar en la figura 3.2, el cual es un diagrama general del sistema, donde de manera general se compone de los siguientes componentes descritos a continuación.

- **Microcontrolador.** Encargado de todo el control de la maquinaria, tanto sensores como actuadores, encendido y secuencias.
- **FPGA.** Encargado de las tareas relacionadas con el procesamiento de video y clasificación, ya que cuenta con un poder de computo superior a los demás componentes. Este será ~~comandado~~ sincronizado/controlado por el microcontrolador.
- **Computadora.** Utilizada para analizar y desarrollar el proyecto, sin embargo no forma parte de la maquinaria final.
- **Circuitos auxiliares.** Encargados de interactuar con el usuario y la interconexión entre dispositivos.
- **Actuadores.** Se encargan de mover la fruta (motores y actuadores).
- **Cámara de video.** Encargada de capturar la escena del paso de la fruta en tiempo real. Cuenta con un puerto HDMI para la transferencia de video.

## 3.3 Maquinaria de flujo de naranja

La

mecanico

Esta maquinaria es la encargada de mover toda la fruta por un circuito para ser procesada. En la figura 3.3 se muestra un diagrama general ~~de la misma, en ella~~ donde se aprecian todos los componentes de que componen el flujo de naranja, empezando por el embudo de alimentación y terminando en la zona de fruta clasificada. Las flechas, muestran el sentido y la dirección de el flujo de las naranjas y en el caso de la flecha con textura punteada se trata de una realimentación de fruta que puede que no haya sido procesada correctamente para dar una nueva oportunidad de ser clasificada.

### 3. Desarrollo

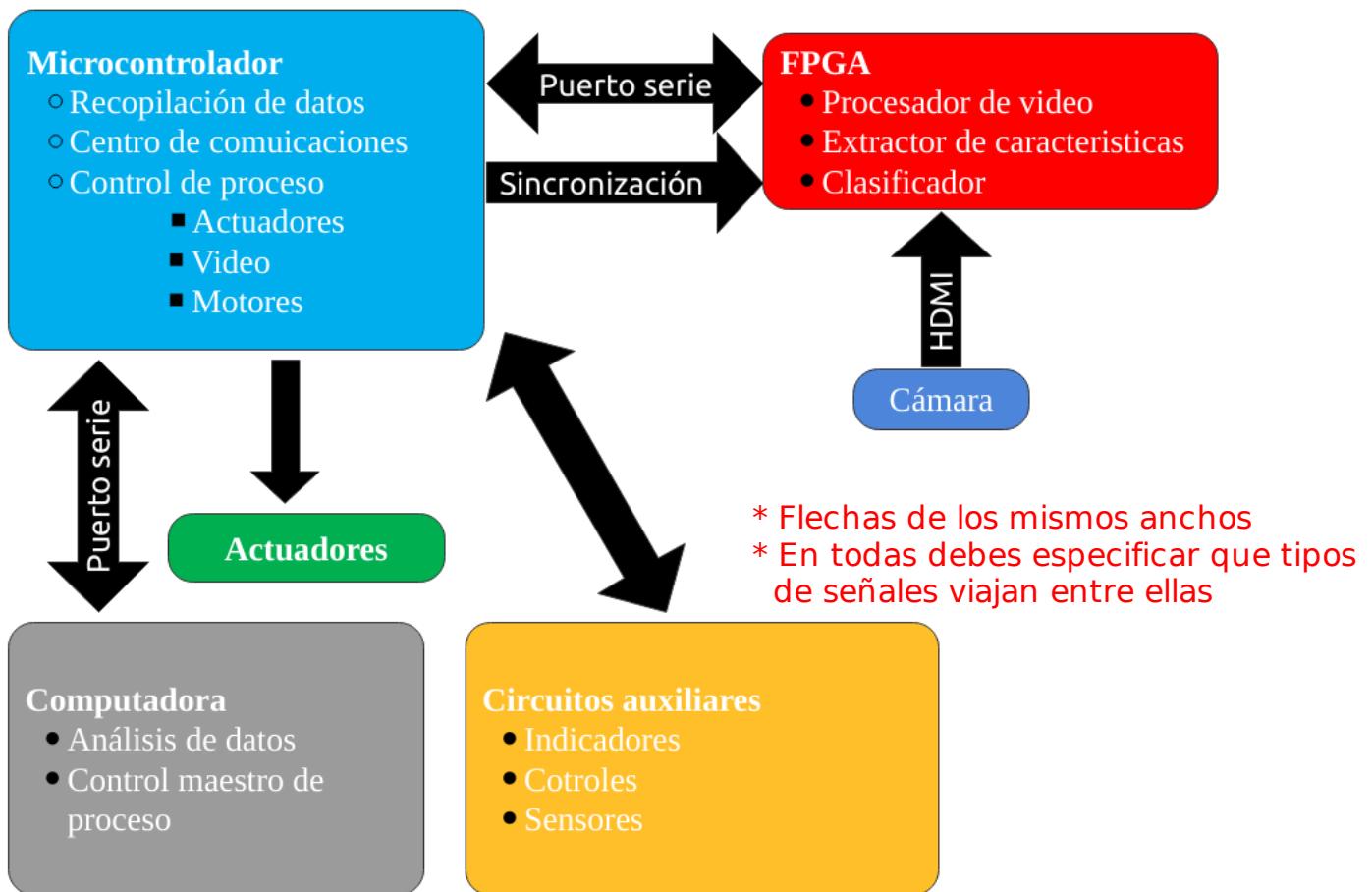


Figura 3.2. Diagrama general del sistema.

A continuación se explica de manera particular cada parte de la maquinaria.

✓ **Embudo de alimentación.** Es el inicio de del circuito de flujo de naranja, en el se dispondrá la fruta, y gracias a su forma, la cual es un embudo con una pendiente negativa, deposita la fruta en la banda de giro. Su diseño no es final, y puede mejorar bastante, ya que puede ser más larga y mucho mas grande, dependiendo de la capacidad a la que se quiera clasificar. Es un elemento puramente mecánico y no cuenta con elementos eléctricos.

✓ **Banda de giro.** Es una banda pequeña, movida por un motor DC de 90V, la cual es controlada por un driver de 0.5HP por lo que la velocidad de giro puede ser controlada para suministrar un empuje diferente a la fruta. La transferencia del giro es por medio de una banda mecánica genérica. Al final

Creo haberle solicitado una figura como la del artículo de la fruta de la palma...  
 \*\* Le encargo por favor!!!

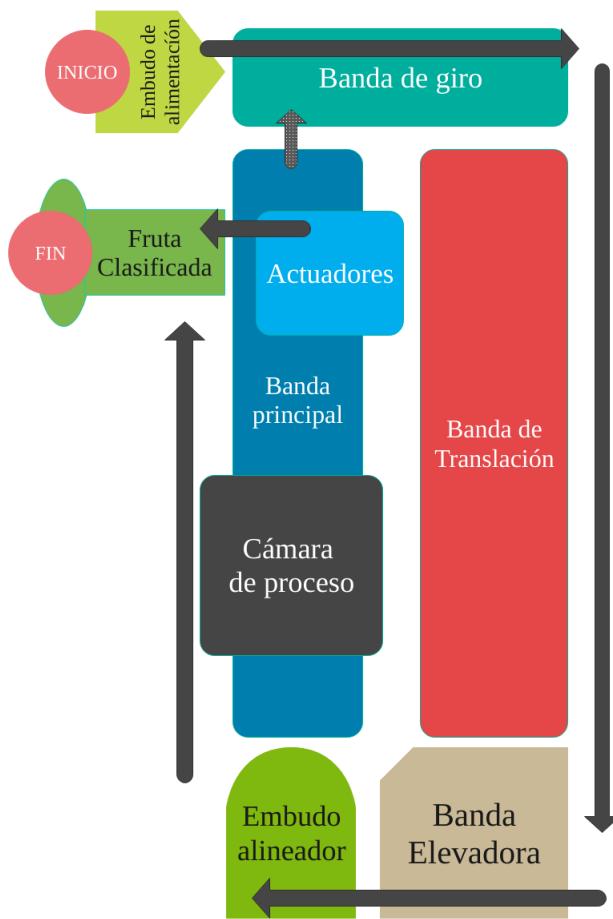


Figura 3.3. Diagrama general de maquinaria para el sistema de separación automática de naranjas.

de la banda cuenta con un riel el cual proporciona un giro de 90 grados para posteriormente la fruta caer a la siguiente etapa.

✓ **Banda de translación.** Es una banda larga movida por un motor AC monofásico. Este motor tiene una velocidad constante, sin embargo la banda cuenta con mucho espacio para la fruta y el motor es de carga pesada.

✓ **Banda elevadora.** Es una banda con paletas la cual se encarga de subir la fruta a la siguiente etapa. Es movida por un Motor AC monofásico. El diseño de la banda permite subir al rededor de 5 naranjas por paleta (esto dependiendo del tamaño de la fruta), posteriormente la fruta cae en un riel que tiene la misma estructura de la banda, la cual sirve para suministrar la siguiente etapa.

✓ **Embudo alineador.** Es una estructura metálica la cual tiene forma de embudo con una

### 3. Desarrollo

---

pendiente negativa, y unos rieles intermedios la cual se muestra en la figura X. Esta estructura es la encargada de hacer que la fruta sea alineada o colocada en hilera para posteriormente alimentar la banda principal, esto con el fin de evitar atascos en el flujo de la naranja. Es un elemento puramente mecánico por lo que no lleva motores.

✓ **Banda principal.** Es una estructura con un riel compuesta de rodillos capaces de crear zócalos de fruta, los cuales sirven para transportar fruta por toda la banda principal. El riel es movido por un motor DC de 90V, la cual es controlada por un driver de 0.5HP por lo que la velocidad de giro puede ser controlada para suministrar un empuje de diferente velocidad a la fruta. En la figura X se puede ver una fotografía da la banda, y en ella esta montada la zona de actuadores y la cámara de proceso que se describirán a continuación.

✓ **Cámara de proceso.** Es una estructura metálica en la cual se contiene un ambiente con iluminación constante con LEDs de color blanco, toda la iluminación externa es eliminada en este lugar. en la entrada de la cámara de proceso se encuentra un sensor de presencia, para detectar si una naranja está por entrar, por otra parte en la parte central y superior se encuentra la cámara digital que se conecta al FPGA.

✓ **Bloque de actuadores.** Se trata de una estructura pequeña montada sobre la banda principal, la cual alberga actuadores eléctricos los cuales empujan la naranja hacia la siguiente etapa, estos serán controlados por el microcontrolador mediante relevadores de estado sólido. La estructura es puramente ajustable, para colocarse cada actuador como sea conveniente en la posición conveniente.

✓ **Bloque de fruta Clasificada.** Es una zona que aun no se construye. Su función será recolectar la fruta ya clasificada, en diferentes rieles, se tiene un diseño preliminar.

### 3.4 Flujo de fruta en maquinaria

En la figura 3.4 se muestra un diagrama de flujo el cual explica el proceso realizado por la maquinaria en general. Primeramente se recibe la fruta no clasificada, la cual debe haber sido lavada con anterioridad, puede tener polvo, sin embargo no puede estar manchada con lodo o algún agente que haga que cambie de color la misma, ya que esto puede afectar el rendimiento de la clasificación.



Figura 3.4. Diagrama de flujo de procesamiento de fruta en la maquinaria.

Posteriormente, la fruta es suministrada al circuito de bandas la cual va a ser la encargada de alinear y acomodar la fruta para su procesamiento de video y decidir la clasificación que se esté tomando. Si la clasificación es válida, pasará a la zona de actuadores y al final del proceso, de otra manera la fruta será suministrada nuevamente al circuito de bandas para ser procesada nuevamente.

## 3.5 Sistema eléctrico/electrónico

Se explican los componentes eléctricos y electrónicos y su conexión propuesta. El sistema eléctrico/electrónico general está compuesto por diversas partes las cuales serán explicadas a detalle en las siguientes secciones. Primeramente veamos un esquema general del sistema (figura 3.5) en donde se puede apreciar los componentes. Las flechas describen el flujo de señales que son emitidas o recibidas en el microcontrolador (MCU), el cuál es el que maneja los estímulos proporcionados por los elementos de control y medición, además maneja las salidas para estimular los componentes mecánicos y de procesamiento de video. Se puede apreciar también que algunos componentes solo

### 3. Desarrollo

se usarán como auxiliares en el desarrollo (DEBUG), por lo que no forman parte del sistema final y cuando estos son eliminados las flechas con textura punteadas son utilizadas.

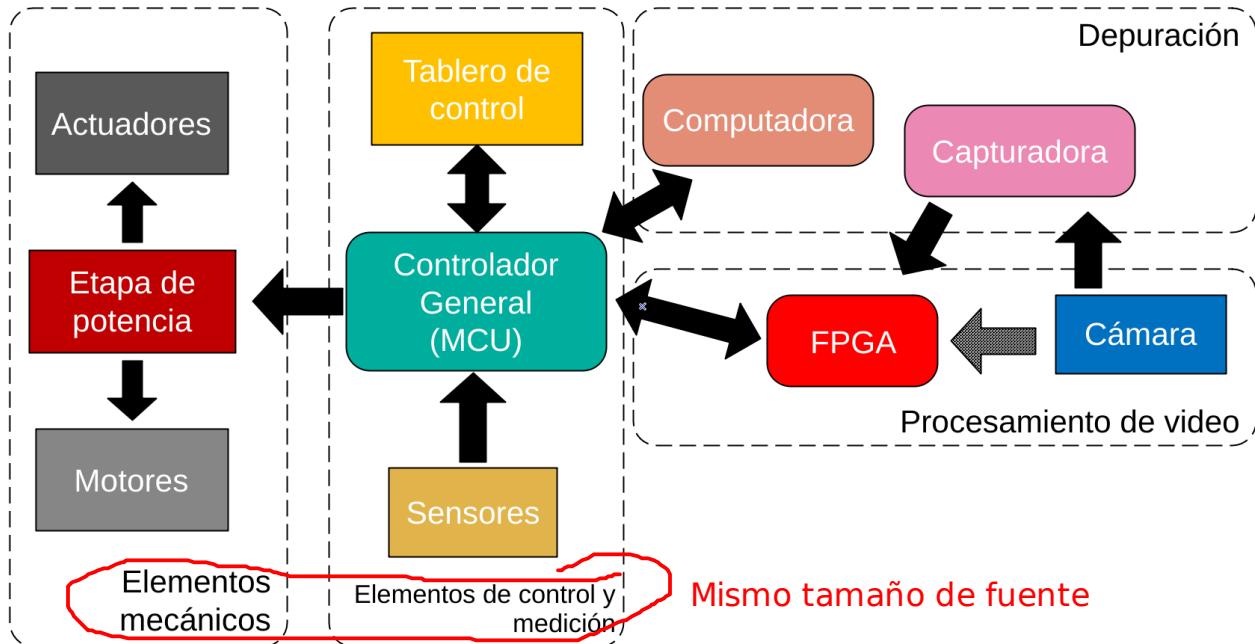


Figura 3.5. Diagrama eléctrico/electrónico general.

A continuación será explicada cada parte en particular.

#### 3.5.1 Elementos mecánicos

Los elementos mecánicos se describen a continuación:

✓ **Actuadores.** Son actuadores eléctricos de AC controlados mediante actuadores de estado sólido (incluidos en la etapa de potencia), esto para mejorar la velocidad de conmutación ya que necesitan ser rápidos.

✓ **Motores.** Las características de los motores se pueden ver en la tabla 3.1. Los drivers de motores DC cuentan con rampa de aceleración para una transición suave y el contacto es controlado por un relevador mecánico activado por el MCU.

✓ **Etapa de potencia.** La etapa de potencia está encargada de el accionamiento de los elementos anteriores, sus conexiones se pueen observar en la figura 3.6, en donde se puede ver el uso de los relevadores mecánicos y de estado sólido.

Tabla 3.1. Características de motores

Posición	Características eléctricas	Controlado por
Banda principal	90VDC 0.5HP	Driver de contacto seco
Banda de giro	90VDC 0.5HP	Driver de contacto seco
Banda elevadora	120 VAC 1HP	Relevador mecánico activado por MCU
Banda de translación	120 VAC 1 HP	Relevador mecánico activado por MCU

No mezcles leyendas con diferentes tamaños de fuente, que toda la figura sea del mismo tamaño!

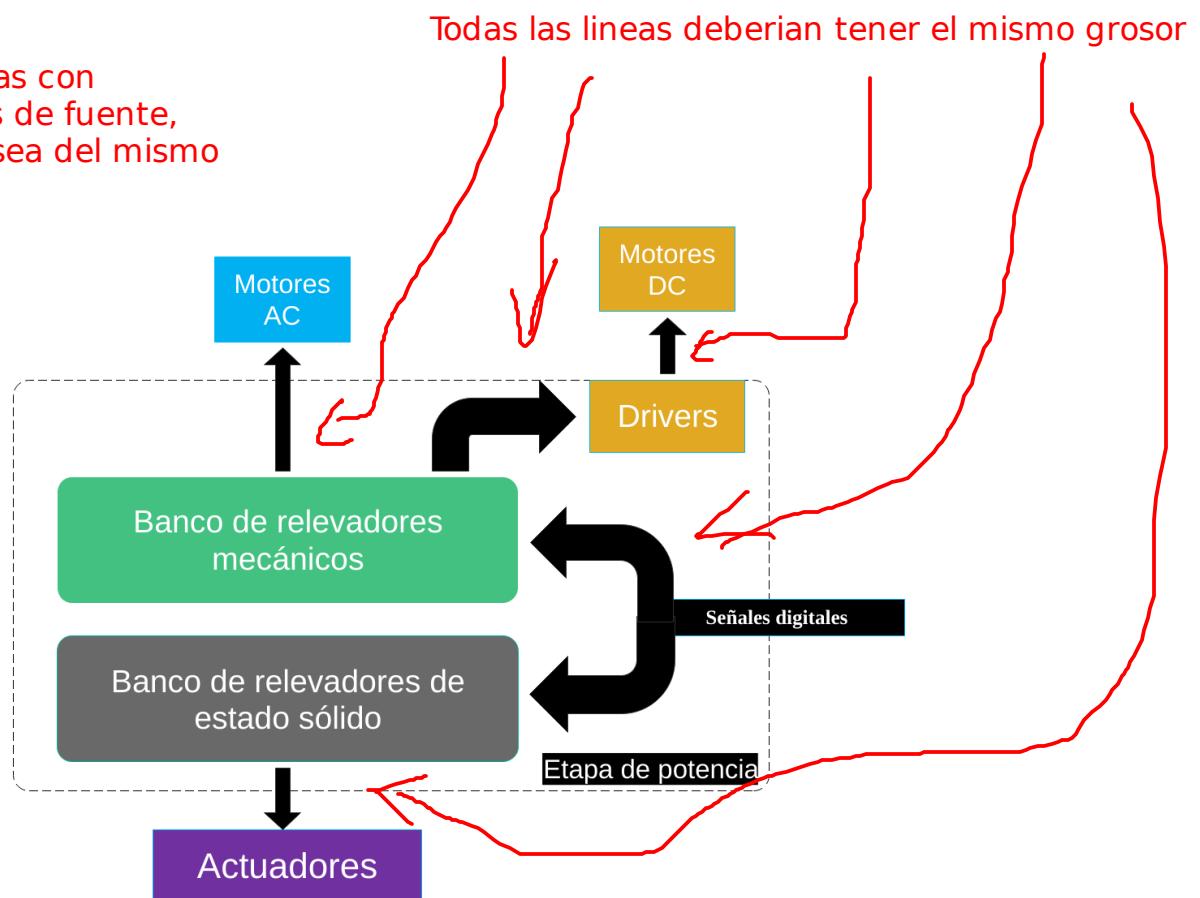


Figura 3.6. Diagrama de conexión de elementos mecánicos

### 3. Desarrollo

#### 3.5.2 Procesador de video Este modulo tiene dos componentes principales:

Es posible utilizar

- ✓ Cámara digital. Es requerida cualquier cámara digital que pueda capturar video usando una resolución de 1080p, a 60 fps, con salida de HDMI. Está colocada en la cámara de proceso y alimenta con video al FPGA.

capture

- ✓ FPGA. Se trata de la tarjeta de desarrollo Industrial Video Processing Kit (IVPK) de Xilinx, con un FPGA Spartan-6 de nivel industrial, además cuenta con una tarjeta I/O de video con puertos HDMI. Su capacidad es mayor a la requerida en el proyecto.<sup>1</sup> Cuenta con su propia fuente de alimentación externa. Actualmente está descontinuada<sup>2</sup> y otros puertos de E/S.

1. Aun no lo sabe. Mejor no lo diga

2. No lo diga ni de broma. Si es cierto que es tecnología vieja, a lo mejor resaltar que es mejorable

#### 3.5.3 Elementos auxiliares o de Depuración

Estos elementos no forman parte del proyecto final, sin embargo son parte fundamental del desarrollo de etapas computacionales y para conocer algunos datos de la maquinaria.

- ✓ Capturadora. Utilizada para la captura de los datasets necesarios para generar los modelos de ADs Puede poner el modelo de la capturadora,

- ✓ Computadora. La computadora es utilizada para recabar cierta información útil, tal como la frecuencia del sensor de sincronía y el resultado de la clasificación.

Puede poner las specs de la computadora (Proc, Mem)

Pero fueron utilizados para la adquisicion de los datos para la generacion de los modelos de ADs y para depuracion

#### 3.5.4 Elementos de control y medición

Estos elementos serán los encargados de la sincronización y activación de secuencias de encendido de la maquinaria.

- ✓ Tablero de control. Este se encarga del control general del sistema y de lámparas de señalización para el usuario el cual se puede ver en la figura X. Cuenta con los siguientes componentes:

##### ENTRADAS

- Botón de Inicio

- Botón de paro de emergencia

- Selector de 3 posiciones.
- x2 Botones de propósito general

## SALIDAS

- x3 Lamparas de señalización.

Para el proyecto solo se utilizaron los botones de inicio y paro. Los demás componentes están disponibles para desarrollos futuros a este trabajo.

✓ **Sensores.** Los sensores en este proyecto solo son 2, los cuales son de proximidad fotoeléctrico Infrarrojo (E18-D80NK), su funcionamiento es detectar si existe un obstáculo frente al sensor, el obstáculo debe estar dentro del umbral de distancia el cual es ajustable desde 1cm hasta 100cm, si existe el obstáculo dentro del umbral, entonces el sensor mandará una señal de 5V. Los sensores tienen los siguientes propósitos:

### 1. Detección de naranja.

Está colocado dentro de la cámara de proceso, justamente en el primer zócalo donde entra la fruta. Su actividad principal es detectar si en el zócalo de entrada hay una naranja, el zócalo de detección está desplazado por 3 zócalos de donde se encuentra el objetivo central de la cámara digital, esto con el objetivo de no interferir en la imagen suministrada al procesador de video.

### 2. Sincronía con banda principal.

Está colocado en la banda principal cerca de la zona de actuadores, aunque puede ser instalado en cualquier parte de la banda principal. Este se encarga de sensar cada zócalo de naranja, y con ello obtener una onda cuadrada, la cual servirá como disparador para el inicio de una clasificación.

En conjunto los sensores definen en que momento procesar la imagen obtenida por la cámara y si es necesario realizar el proceso, ya que si no hay una naranja en el objetivo del zócalo no es necesario realizarlo y así ahorrar recursos computacionales.

### 3. Desarrollo

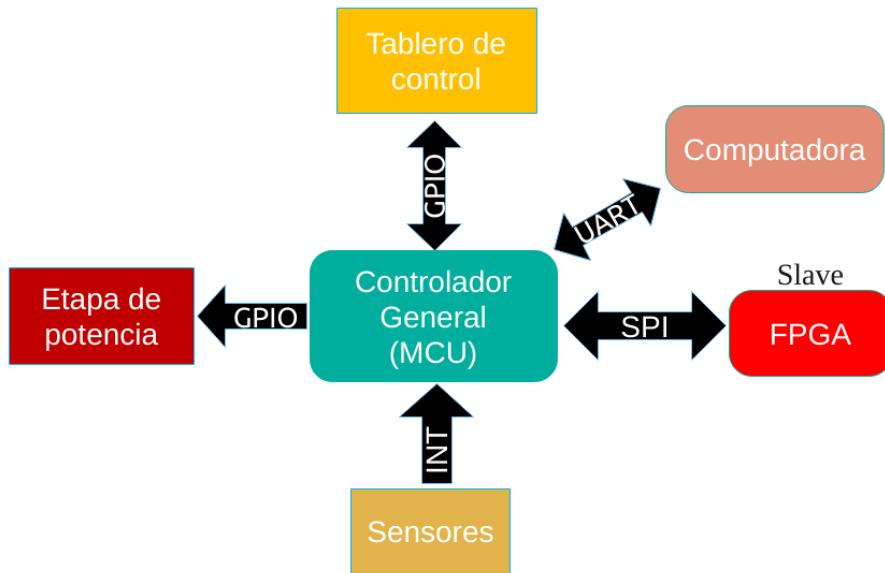


Figura 3.7. Configuración de terminales en MCU

✓ **Controlador General.** El MCU usado para el proyecto es un ATMEGA 2560, el cual cuenta con los GPIO necesarios con los pins de interrupciones necesarias. En la figura 3.7 se puede apreciar un diagrama con los tipos de conexiones utilizadas.

Se optó por un puerto SPI en el FPGA debido a la cantidad reducida de GPIOs en la tarjeta IVPK, debido a que el MCU es el maestro de la conexión, se pueden disparar instrucciones desde el microcontrolador hacia el FPGA, para iniciar una clasificación, esperar un momento y recibir la respuesta.

**De lo que te hable en el primer comentario!!!**

**muestra**

En la figura 3.8 se puede ver un diagrama de flujo del programa que contiene el MCU. Es un diagrama de flujo reducido, debido a que faltan la parte de la comunicación con la computadora y el cálculo de frecuencia la señal de sincronía, sin embargo es un diagrama muy cercano a la realidad. En una primera instancia se detecta el pulso de inicio y paro para dar a lugar las secuencias de encendido y apagado, ya que la maquinaria es de alta potencia se necesitan encender motores a tiempos espaciados, ya que de otra manera se producen transitorios de alta corriente que pueden dañar la instalación eléctrica. Después se detecta el pulso de sincronía, y al ser detectado, se sensa si hay una naranja en el zócalo de la cámara de proceso, de haberlo, se manda al FPGA una

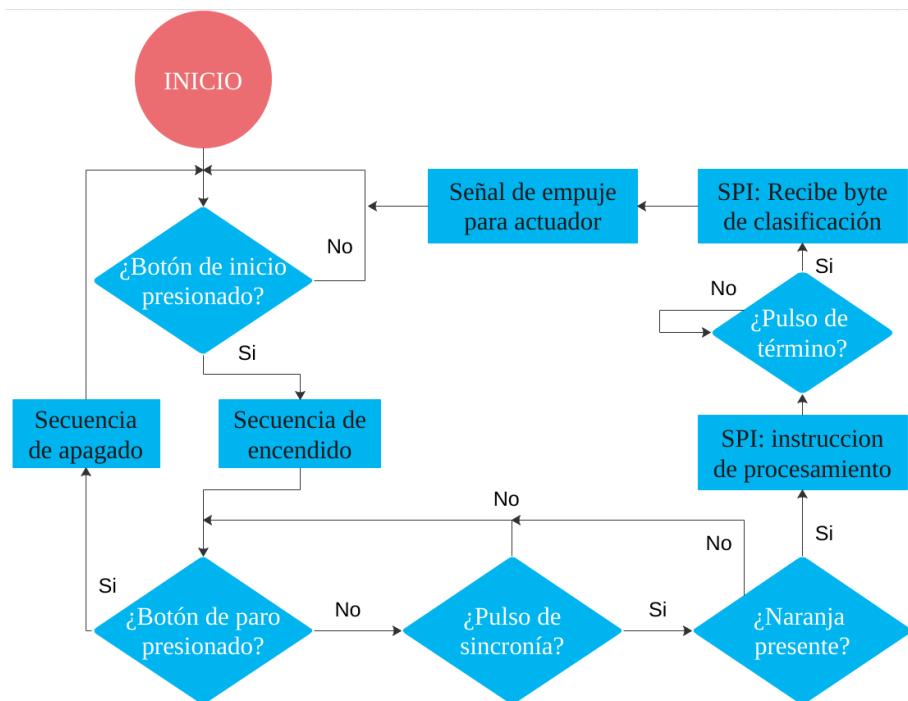


Figura 3.8. Diagrama de flujo del programa del MCU y los protocolos de comunicación empleados.

instrucción de procesamiento, se espera un pulso proveniente del FPGA para saber que ya ha acabado el procesamiento, esto debe de durar alrededor de ~~64ms~~ ya que es la tasa de refresco que tiene la cámara es de ~~16ms~~ por cuadro y el retraso es de 4 cuadros posteriores, y debe de analizar la imagen en el momento justo. Al recibir el pulso, el MCU recibe el resultado de la clasificación por medio de un byte, el cual contiene la información de tamaño y color de la fruta. Dependiendo de la respuesta, se ~~computa~~ el movimiento de empuje para el actuador.

~~genera/calcula~~

## 3.6 Modelo AD de segmentación

En esta sección se describen los pasos necesarios para construir el AD de segmentación hecho por computadora y en una sección posterior se explica como se realiza este modelo en un módulo sintetizable de HDL.

### 3. Desarrollo

---

#### 3.6.1 Captura de video

Para el desarrollo del modelo, primeramente se obtuvieron los datos para ~~poderlos manejar~~ y alimentar el modelo. Para ello se accionó la maquinaria con fruta en los circuitos de bandas sin activar los actuadores, para que estas giraran entre el circuito de bandas, con esto y los elementos auxiliares vistos en la sección anterior y con ayuda de la cámara se tomaron videos de la fruta pasando por el objetivo de la cámara para de ellos obtener ~~los datos necesarios para fabricar el modelo~~.

los videos, que posteriormente fueron divididos en imágenes individuales (fotogramas), y generar...

#### 3.6.2 Fotogramas de interés

Dado que en no todos los fotogramas de los videos capturados se encuentran imágenes de la fruta, y además solo se tomará una fracción del fotograma para realizar la clasificación se decidió construir un algoritmo de detección de la fruta para que de esta manera se lograra obtener un ~~dataset~~ de imágenes de fruta.

Primeramente se decidió la región de interés (ROI), la cual se puede ver en la figura 3.9 como un recuadro de color verde en la parte central. La región tiene una medida de 600x600 pixeles en una región conveniente del fotograma ya que en ella se encuentra el zócalo de captura de manera centrada. Dentro de ROI, se encuentran unos límites centrales las cuales son unas barras verticales de color verde. Entre estos límites se tomará el fotograma una vez que sea detectado el centroide de la fruta.

Para la segmentación de la naranja, ~~simplemente~~ la imagen de entrada para cada cuadro fue convertida a escala de grises y binarizada con un <sup>\*\*</sup> umbral. Posteriormente encontrar el contorno con mayor área y llenar el contorno, con esto se obtiene una máscara tal como se ve en la figura 3.10b. Con ésta máscara, se calcula el centroide y el área de la misma, si el área es mayor a un sexto del área total de pixeles (10000), quiere decir que se encuentra una fruta dentro de ROI. Por otra parte, se calcula si el centroide en el eje horizontal está entre los límites centrales, y de esa manera se toma la captura de la imagen tal y como se ve en la figura 3.10a. Debido a las diferentes velocidades de la banda las capturas pueden tener una difuminación tal y como se ve en la figura 3.10, sin embargo

<sup>\*\*</sup> Se utilizo el metodo de OTSU para detectar el umbral automatico o fue umbral fijo? Deberias decirlo

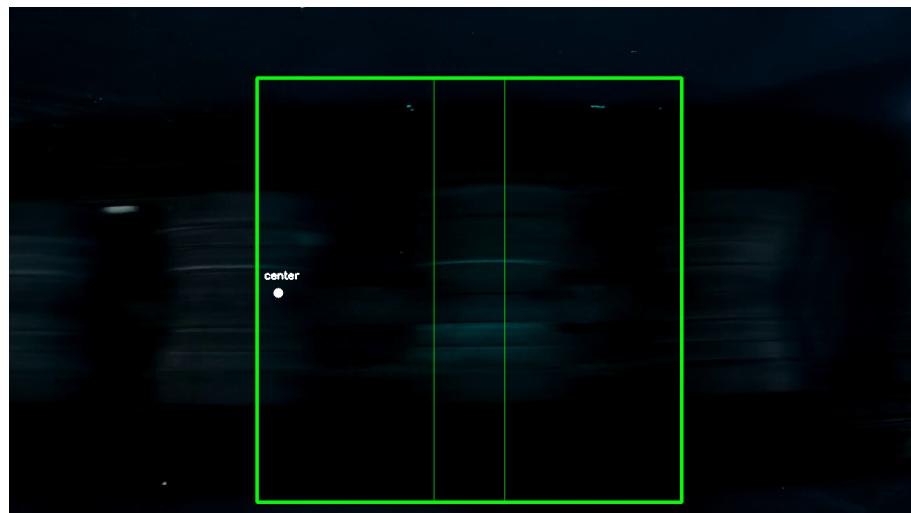


Figura 3.9. Escena de cámara dentro de cabina y ROI definida.

esto no afecta en la toma del set de imágenes. (ver figuras 3.10 c y 3.10 d)

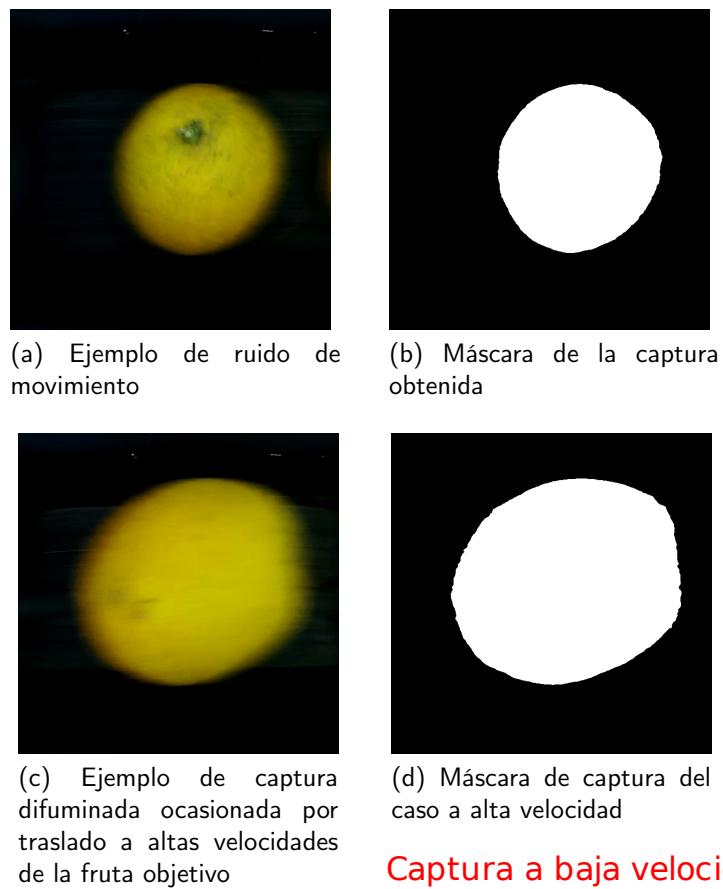
En la figura 3.11 se puede ver una imagen completa del proceso antes descrito, donde la parte enrojecida es la máscara, además se encuentra indicado tanto el ROI, como los límites centrales y el centroide de la máscara.

#### 3.6.3 Captura de data set de pixeles

Una vez obtenidas las imágenes visto en la sección anterior, estas se utilizan para realizar el data set de pixeles. En la figura 3.12 se muestra un diagrama de flujo el cual explica el proceso de la captura del data set. Para cada imagen obtenida, se obtiene la segmentación, la conversión a espacios de color a escala de grises y HSV. Una vez obtenidos, cada canal es separado y guardado como imágenes independientes, para ser convertidas de matrices a vectores planos verticales. Por último, estos vectores son apilados horizontalmente para crear una matriz donde se tiene cada canal como una columna. Realizar esto para cada imagen e ir apilando los resultados en una matriz donde se guarda el resultado de todos las imágenes, para al final remover todos los duplicados. Como paso final, se eliminaron todos los pixeles los cuales mostraran los mismos valores en los canales de color pero diferente segmentación, por ejemplo, en la figura 3.12, para el pixel 1 y 4 no son válidos, debido a que en todos los demás canales tienen los mismos datos, sin embargo en la segmentación son

### 3. Desarrollo

---



Captura a baja velocidad (a) y (c), resultado de generación de mascara (b) y (d)

Figura 3.10. Captura para el set de datos y obtención de la mascara. Difuminación por movimiento.

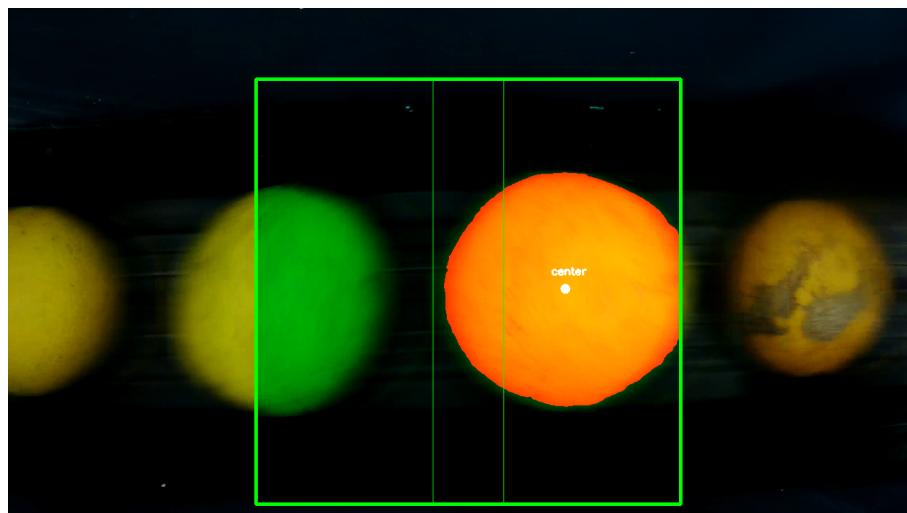


Figura 3.11. Proceso de captura de imágenes.

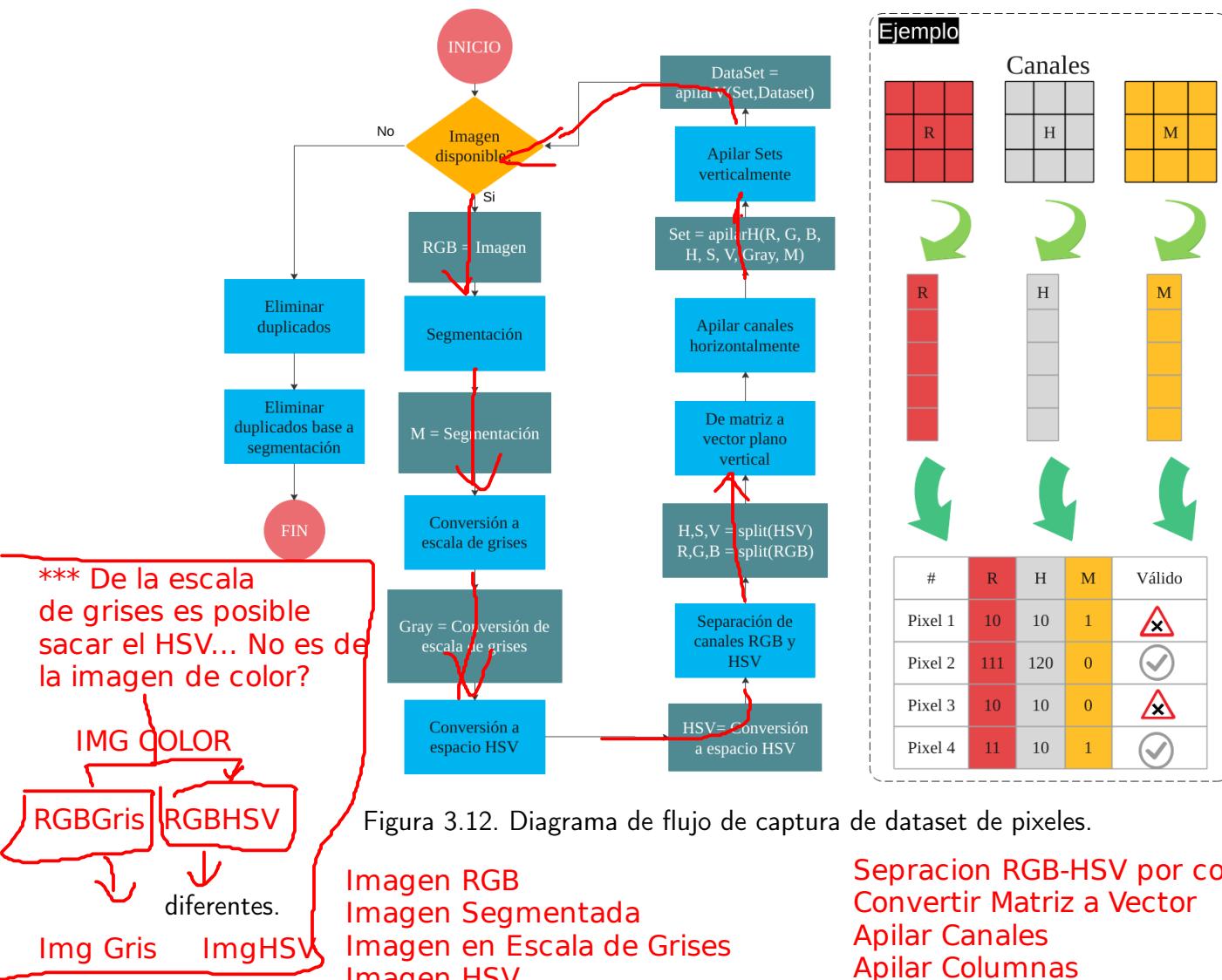


Figura 3.12. Diagrama de flujo de captura de dataset de pixeles.

Separación RGB-HSV por columnas  
Convertir Matriz a Vector  
Apilar Canales  
Apilar Columnas

Esto es un problema grave y no existe una separación espacial entre los datos. Estos pixeles son eliminados del dataset para mostrar una tendencia entre lo que es un pixel que pertenece a la fruta y uno que no.

De última instancia, se procedió al entrenamiento en WEKA, lo cual se realizó con un modelo del tipo J48 de ramas binarias con un costo  $C = 0,25$ .

\* Que significa ese costo C (va a ser lento, rápido, va a ser preciso o va ser poco preciso... QUE SIGNIFICA)....

### 3. Desarrollo

## 3.7 Modelo AD clasificador de color

Para el desarrollo de este modelo, se utilizó el dataset de imágenes anterior. Primeramente se analizó el dataset de imágenes y se clasificó de manera manual, separando entre las clases "naranja" "verde". Una vez clasificadas se realizó una captura de histogramas que se verá en las secciones siguientes.

el calculo de los

### 3.7.1 Análisis de color

Dada la naturaleza del canal H, el cual describe el color del pixel, dependiendo de su iluminación (ya que el negro se interpreta como ausencia de luz y blanco como presencia de todos los colores) se decidió realizar un pequeño análisis de colores el cual se puede ver en la figura 3.13, en donde se muestran 4 muestras de fruta, con 4 casos posibles diferentes donde 2 de ellas están muy bien definidas entre su clase (3.13a y 3.13d) y otras cerca del umbral de lo que se puede considerar como naranja o verde (3.13c y 3.13d).

En cada una de ellas, se analizó un histograma solo sobre los pixeles que son segmentados. Ya que se hace muy complejo el análisis de un histograma completo como arquitectura sintetizable, se decidió optar por uno de 5 bins. En cada una de las muestras los acompaña un histograma completo, donde las barras verticales son las divisiones, y las otras barras el resultado del histograma, también lo acompaña la representación de un histograma de 5 bins, y una representación de la muestra en el canal H. Como se puede notar, en la representación del canal H, el color verde cuenta con una intensidad más alta, que la del color naranja. Además se puede ver un patrón diferente del histograma de 5 bins para cada caso, por lo que el canal H es un buen candidato para la separación de datos, sin embargo se optó por realizar por un histograma de 5 bins de todos los canales disponibles, ya que se vio que de esta manera es un buen método, por otra parte, si los datos llegasen a ser irrelevantes, el algoritmo C.45 se encargará de eliminar todos aquellos datos que no aporten a la separación entre clases.

cursiva

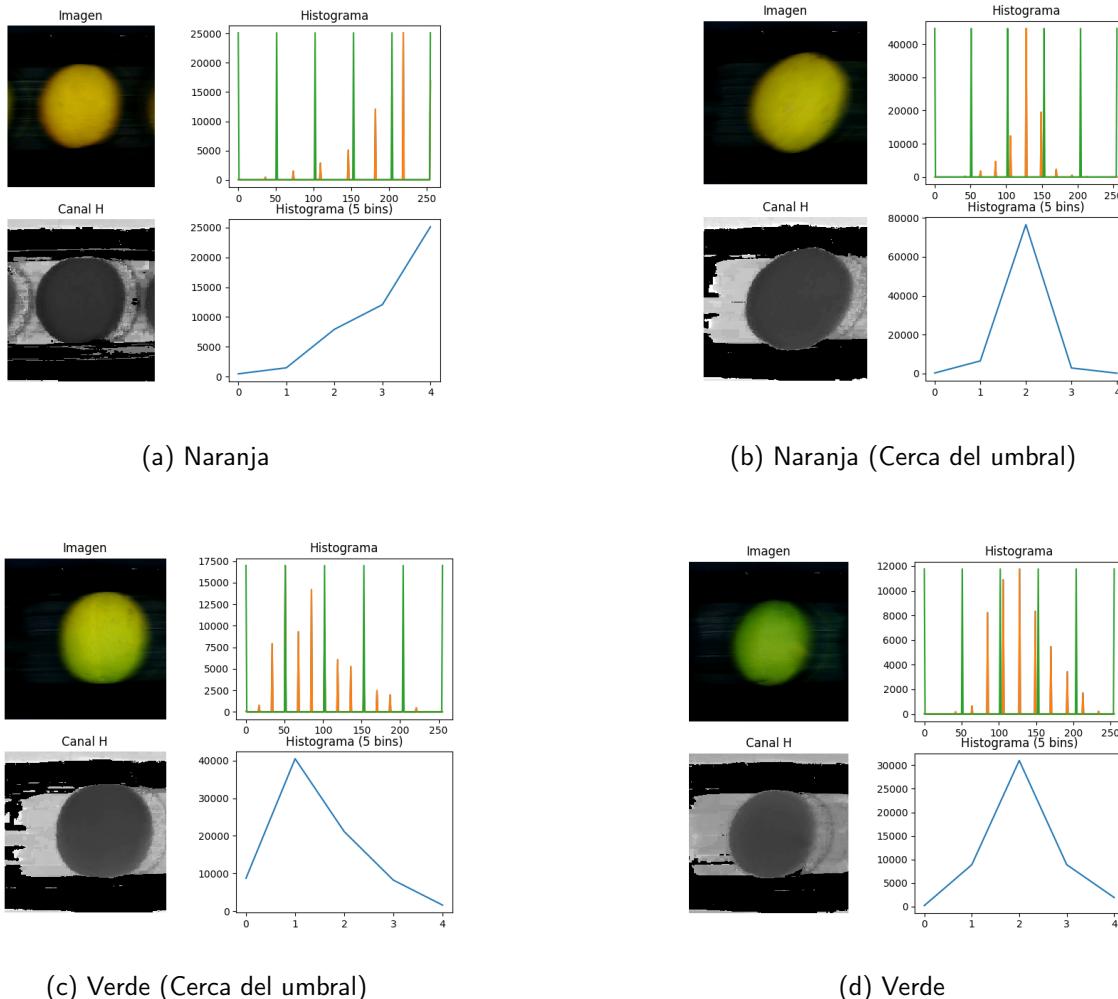


Figura 3.13. Resultados del análisis para cuatro posibles escenarios de color

### 3.7.2 Captura de data set de histogramas

Debido al tamaño distinto para cada muestra, se notó que un problema fuese el rango de valores que toman los histogramas por lo que se decidió tomar histogramas normalizados. La norma se tomaría a partir de el total de pixeles segmentados. En la figura 3.14 se puede ver un diagrama de flujo de como se capturó el **dataset**.

Primeramente, para cada imagen de cada clase primero se calcula la segmentación y el área de

### 3. Desarrollo

la misma, osease todos los pixeles de color blanco. Posteriormente se calculan todos los espacios de colores en base a la imagen de entrada, para eliminar todos los elementos de pixeles no segmentados, para solo tomar en cuenta los pixeles que pertenecen a la fruta y con ello calcular el histograma de 5 bins obteniendo un vector con los valores del histograma. Posteriormente cada elemento del histograma es multiplicado por 10000 para cambiar el valor de la precisión del histograma, esto con el fin de solamente manejar números enteros en los cálculos, ya que se tornan difíciles de manejar en un modelo HDL sintetizable. Por último para normalizar el histograma se divide cada elemento entre el área, y esto apilarlo en una matriz en donde se almacenan los datos para cada muestra.

En la figura 3.14 se muestra un ejemplo gráfico de como se efectúan los cálculos. Dado que es un ejemplo, solo se muestra un caso hipotético con un solo canal, obteniendo un dataset de 4 muestras con 5 características y una clase. Sin embargo en el experimento real, dado que los canales utilizados fueron R, G, B, H, S, V y Gray, por lo tanto se obtiene un dataset del número de imágenes con 35 características y una clase. La clase está representada por un número, siendo el número 1 para el color verde y 0 para el color naranja. **cursiva**

De última instancia, se procedió al entrenamiento en WEKA, de la misma manera que el modelo AD de segmentación.

## 3.8 Modelo AD clasificador por tamaño

Para el desarrollo de este modelo, se utilizó el dataset de imágenes anterior. Primeramente se clasificó cada imagen de manera manual, bajo el criterio del autor, obteniendo 3 clases: chico, mediano y grande. De esta manera para cada imagen de cada clase se tomó como característica principal el área de segmentación.

\* Falta esta figura, ademas de que no esta claro quienes son las caracterisitcas

De última instancia, se procedió al entrenamiento en WEKA, de la misma manera que el modelo AD de segmentación. Éste árbol se generó con el fin de únicamente obtener los umbrales de área, y debido a que es un árbol sumamente pequeño, este no se realizará con los métodos de AD sintetizables

Yo creo que debe poner ese arbol, aun cuando es pequeño!!

Mismo comentario que en figura 3.12 (es decir, especificar que entra y que sale de cada proceso)

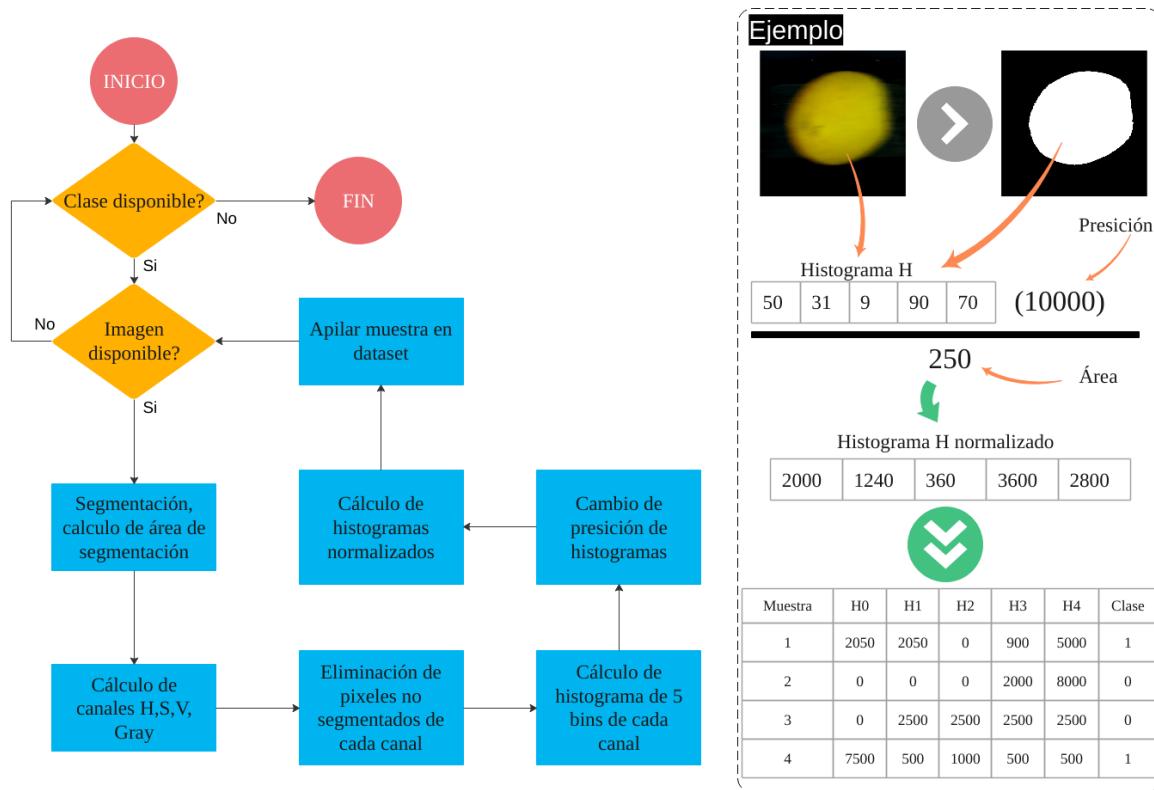


Figura 3.14. Diagrama de flujo y ejemplo de captura de datos de histogramas.

### 3. Desarrollo

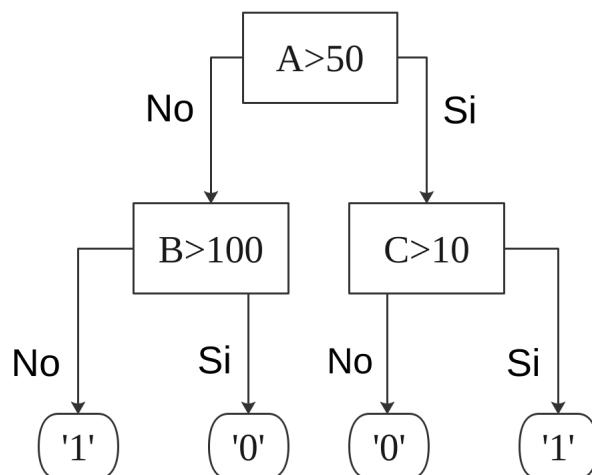
---

explicados en secciones siguientes, si no, con comparadores fijos.

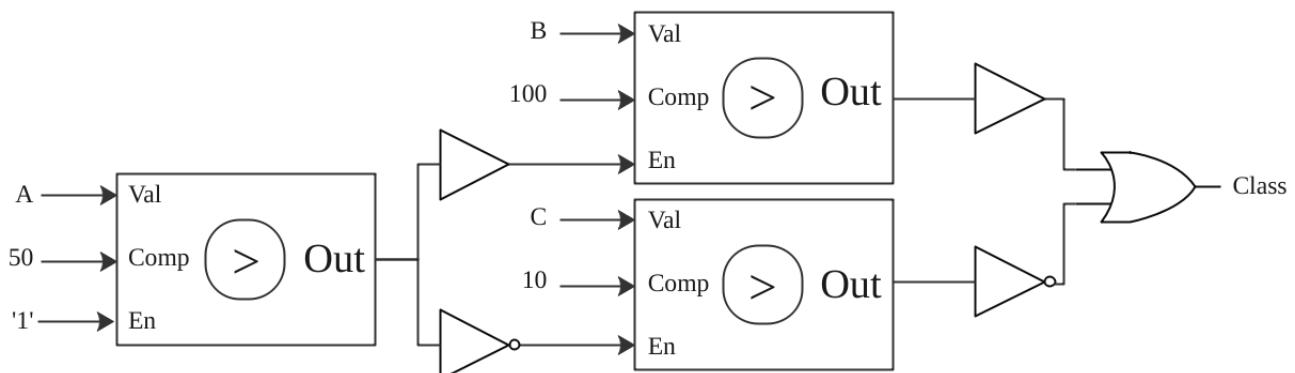
## 3.9 Arquitectura genérica de árbol de decisión

Un AD puede ser fácilmente aplicado a hardware mediante HDL. Para explicar la arquitectura se tomará como referencia el AD de la figura 3.15a, además, se tomará que todos los componentes sencillos tales como buffers, y compuertas lógicas comprenden solamente 1 unidad de retardo (UDR). El modelo encontrado en la literatura corresponde al de la figura 3.15b, en donde se puede ver bloques generales los cuales corresponden a comparadores con habilitación (el pin 'En'), donde sus entradas corresponden el valor a comparar (Val) y el valor fijo de comparación (Comp), estos, comprenden 2 UDR ya que primeramente se calcula si debe realizarse la comparación, y si esta se va a realizar se computa la salida. Sabiendo esto, se puede calcular que se tiene en el circuito completo un retardo de 7 UDR desde la entrada hasta la salida. Se puede reducir el circuito en 1 UDR reemplazando el primer comparador por uno sin habilitación, sin embargo, entonces un componente del módulo sería diferente y habría que realizar el cambio manualmente y correr riesgo de equivocación por parte del programador.

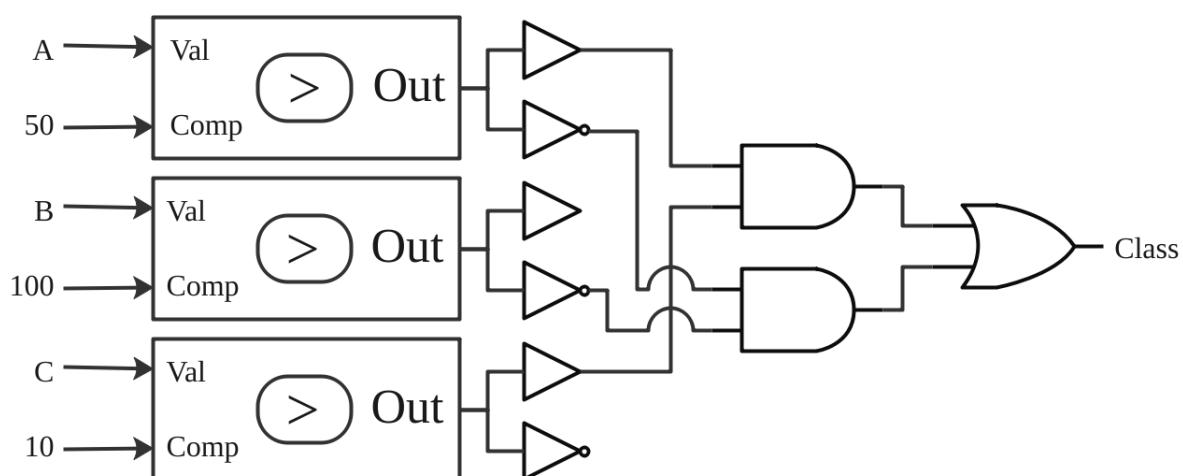
Agregar referencia al articulo de la literatura.  
Juan Perez propuso hacer esto.... Nos dimos cuenta  
que era una gran pdx, pero nosotros propusimos  
una modificacion para hacer esto, esto y esto otro...



(a) Ejemplo de AD



(b) Ejemplo de arquitectura de AD en serie



(c) Ejemplo de arquitectura de AD en paralelo (propuesto)

Figura 3.15. Modelos de AD en Hardware.

### 3. Desarrollo

---

Por otra parte la arquitectura que se propone una forma más eficiente de realizar el cálculo de la clasificación, colocando algunos bloques en paralelo. Ésta se puede ver en la figura 3.15c la cual ~~se comprende~~ de 3 etapas. La primera, se compone de todos los comparadores, ya que están en paralelo y no cuentan con un pin de habilitación estos solo tienen 1 UDR, después se tiene la etapa de compuertas AND, el cual resuelve la ramificación del AD, y por último, la etapa de compuertas OR donde se adjuntan todas las posibilidades donde la salida es la clase '1'. Con esta forma, se tienen 4 UDR en el circuito, reduciendo el computo de la clase en 3 UDR.

#### En caso de requerir la implementación de

~~De tenerse~~ un AD más profundo ~~en el caso del sistema propuesto~~, la etapa de comparación no retrasaría más el computo, ya que la comparación se resuelve en la primera etapa de manera paralela, por otra parte la solución de la ramificación de la segunda etapa se puede reducir las UDR siempre y cuando se conecten la mayor cantidad de compuertas AND posible en paralelo, igualmente para la última etapa. Esto claramente no es posible en la arquitectura en serie.

#### 3.9.1 Generación de un HDL a partir de un AD

~~los ADs son diferentes para cada conjunto de datos,~~

Debido a ~~que para las pruebas cada AD es diferente~~, se decidió realizar un Script que genera el HDL ~~desde~~ a partir de un modelo de AD generado en WEKA, el cual tiene como nombre Script Constructor HDL de árbol de decisión (SCHAD). Para la explicación del algoritmo se usará como referencia la figura 3.16, donde se muestra un modelo AD de WEKA. El algoritmo funciona de la siguiente manera. Primeramente se identifican todas las ramificaciones en donde se obtiene la clase '1', como en la linea 5 y 6 del ejemplo, y se generan todas los comparadores que comprenden esas ramificaciones, junto con sus buffers e inversores, esto para ahorrar hardware en el AD excluyendo todas las compuertas restantes. Después para cada ramificación por medio de una compuerta de n entradas, donde n es el número de comparadores necesarios para resolver la ramificación, se conectan todas las salidas invertidas o no invertidas del comparador que resuelve cada ramificación por ejemplo la ramificación de la linea 5 es la inversión de la linea 1,2 y 4, Por último, las salidas de cada compuerta AND es conectada a una compuerta OR de m entradas, donde m es el número de ramificaciones donde la

n en cursiva

\* Creo que este apartado te hubiera ayudado mucho un Algoritmo. Pero si no es posible hacerlo, dejalo en su forma actual!

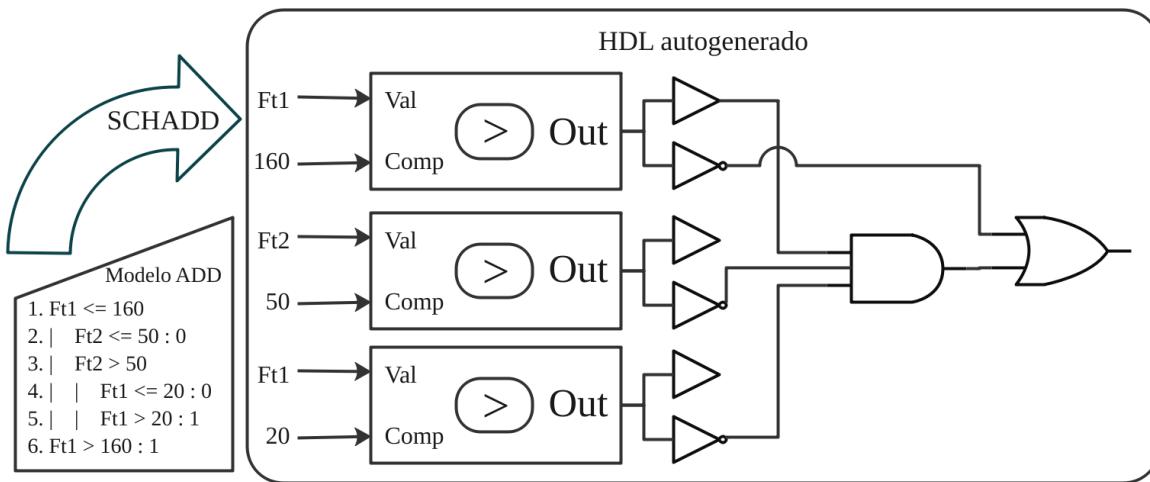
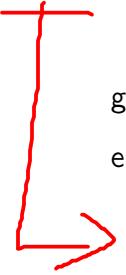


Figura 3.16. Ejemplo de AD generado mediante ~~SCHAD~~, el **SCHAD** desarrollado.

clase es '1' en el AD, y la salida de la compuerta OR es la clase computada.

## 3.10 Arquitectura del procesador de video

La arquitectura propuesta en el proyecto está compuesta de varias etapas, las cuales independientemente cumplen con una tarea en específico en donde se va resolviendo por etapas la clasificación. En la figura 3.17 se puede apreciar un esquema general de la arquitectura, en donde se aprecia las capas de comunicación y de clasificación.

 La capa de comunicación se encarga de manejar las señales entre el MCU y el FPGA, para la gestión de la clasificación, ya que la capa de clasificación estará en reposo y solo funcionará cuando el MCU lo indique.

La capa de clasificación está compuesta por 4 etapas, en las cuales se implementó una arquitectura en pipeline, por lo que entre cada capa se encuentran módulos de registro controlados por el reloj de video. Debido a las etapas de registros es que el sistema estará retrasado 4 ciclos de reloj de video, lo cual es despreciable teniendo un reloj a una frecuencia de 155MHz normalmente. Las capas y etapas de la arquitectura serán explicadas en esta sección.

### 3. Desarrollo

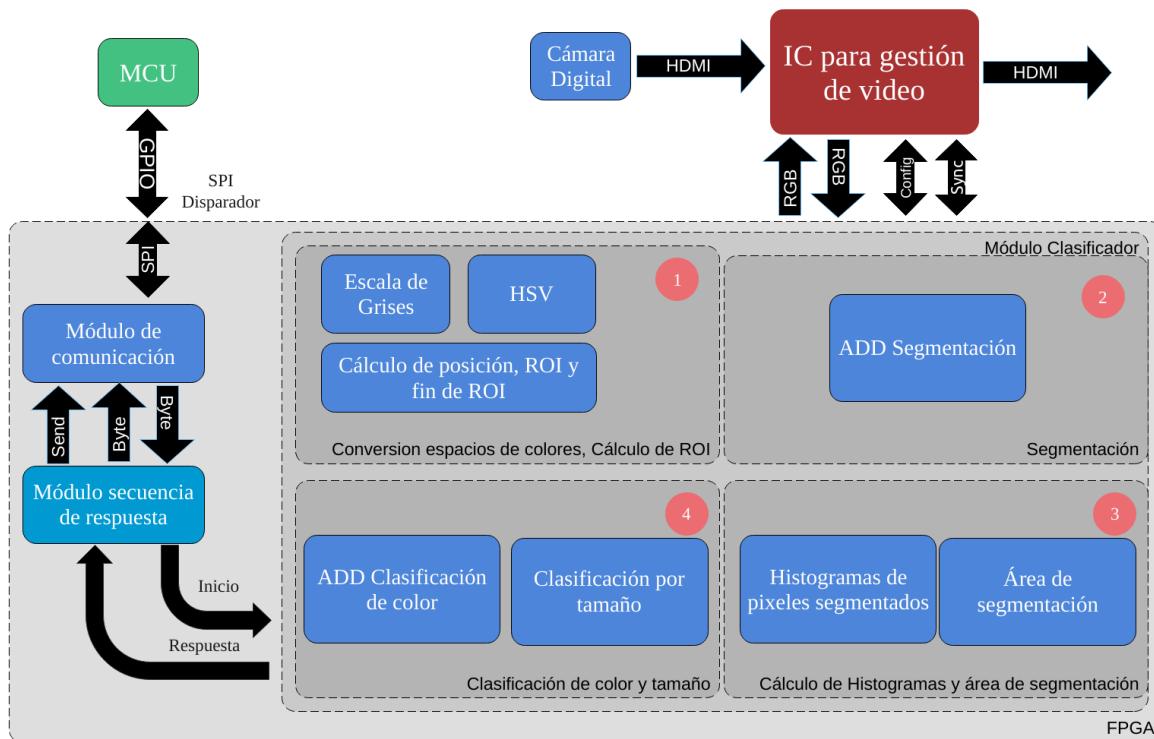


Figura 3.17. Esquema general de la arquitectura.

Mismo comentario con respecto a los tamaños de las fuentes. Incluso es posible hacer el Diagrama mas grande en caso de ser necesario

### 3.11 Módulo Clasificador

El módulo clasificador es parte de la capa de clasificación la cual está compuesta de 4 etapas que serán explicadas en esta sección.

La razón de dividirlo en etapas, es que la arquitectura de algunos módulos es compleja, aún tratándose de circuitos puramente combinacionales, por lo que la salida se vería afectada si se realizara un circuito combinacional general. Además, la naturaleza de algunos componentes son secuenciales por lo que registros entre las etapas, facilitan el manejo y orden de los datos.

La primera etapa está encargada del cálculo de algunas métricas utilizadas en etapas posteriores, la segunda está encargada puramente de la segmentación, la cual es de vital importancia para las siguientes etapas. La tercera, está encargada de la recolección de datos para la clasificación por color

En vez de compleja, puede decir, para mantener un orden estructural separando lógicamente los bloques, procurando preservar el retardo minimo en componentes combinacionales y empleando etapas de registros entre bloques combinacionales

y tamaño, y la última es la encargada de realizar la clasificación.

### 3.11.1 Conversión de espacios de colores y cálculo de ROI

En esta etapa se realizan los cálculos de espacios de colores y posicionamiento de pixeles. Está posicionado en primer lugar ya que la conversión de espacios de colores será la entrada de módulos posteriores, así como el conocimiento del posicionamiento de la ROI. La etapa está compuesta por los siguientes módulos.

#### 3.11.1.1 Módulo convertidor RGB a escala de grises

Este módulo tiene como entradas los canales del pixel entrante y la señal de reloj, y simplemente se encarga de realizar la operación que se aprecia en la ecuación 3.1 (la que corresponde como salida del módulo), la cuál es la implementada en las librerías de OpenCV ??, esto con el fin de coincidir con en cálculo de datos obtenidos desde la computadora. El módulo está sincronizado por lo que se hará el cálculo cada flanco positivo del reloj. **R, G y B son los componentes.... , y Gray es el nivel de gris ...**

$$Gray(R, G, B) = (0,299)R + (0,587)G + (0,114)B \quad (3.1)$$

Sin embargo, en este módulo no se utilizaron cálculos con números de punto flotante por lo que se calculó con números de punto fijo utilizando la ecuación 3.2. Para determinar esta ecuación primeramente se obtuvo la precisión necesaria para el cálculo, es decir, el número de bits necesarios después del punto en el formato de punto fijo. Como cada coeficiente de la ecuación 3.1 es menor que cero, entonces, no se necesitarán bits de la parte entera por lo que solo hay que poder representar los números en enteros, y para definir el número de bits es necesario poder representar el más grande de ellos, en este caso el numero 587 el cual se representa con mínimo 10 bits. Posteriormente al realizar las multiplicaciones, cada número resultante es de 18 bits, por lo que al hacer la suma, resulta en un número con la misma cantidad de bits. Finalmente se descartan todos los bits de precisión, tomando los 8 bits más significativos, o lo que es lo mismo, dividir el la suma entre  $2^{10}$ .

### 3. Desarrollo

---

$$Gray(R, G, B) = \frac{(299)R + (587)G + (114)B}{2^{10}} \quad (3.2)$$

#### 3.11.1.2. Módulo convertidor RGB a HSV

Este módulo tiene como entradas los canales del pixel entrante y la señal de reloj, y simplemente se encarga de realizar las operaciones que se aprecian en las ecuaciones 3.3, 3.4, 3.5, 3.6. Las cuales también son las implementadas en las librerías de OpenCV [45]. El módulo está sincronizado por lo que se hará el cálculo cada flanco positivo del reloj.

$$\begin{aligned} R' &= \frac{R}{255} \\ G' &= \frac{G}{255} \\ B' &= \frac{B}{255} \end{aligned} \quad (3.3)$$

$$\begin{aligned} V' &= \max(R', G', B') \\ V &= (255)V' \end{aligned} \quad (3.4)$$

$$\begin{aligned} D' &= V' - \min(R', G', B') \\ S' &= \begin{cases} 0 & \text{si } V' = 0 \\ \frac{D'}{V'} & \text{si } V' \neq 0 \end{cases} \\ S &= (255)S' \end{aligned} \quad (3.5)$$

$$\begin{aligned} H' &= \begin{cases} 0 + \frac{60(G' - B')}{D} & \text{si } V' = R' \\ 120 + \frac{60(B' - R')}{D} & \text{si } V' = G' \\ 240 + \frac{60(R' - G')}{D} & \text{si } V' = B' \end{cases} \\ &\text{si } H' < 0 \text{ entonces } H' = H' + 360 \\ H &= \frac{H'}{2} \end{aligned} \quad (3.6)$$

Sin embargo no se utilizaron cálculos con números de punto flotante ni enteros negativos. Como

se puede ver en la ecuaciones 3.7, 3.8, 3.9, no se utilizan los componentes  $R'$ ,  $G'$  y  $B'$ .

**En la ecuacion, ... para el cálculo de S es necesario multiplicar D por 255...**

Para la ecuación 3.8 **para el cálculo de S**, primeramente  $D$  tiene que ser multiplicado por 255, debido a que tanto  $D$  como  $V$  son de 8 bits, por lo que al dividirse primero resultaría en un numero de 0 bits lo cual es inconsistente, por otra parte la operación consigue obtener una relación entre los dos números con un rango  $[0, 1]$  y posteriormente multiplicarlo por 255 originalmente en la ecuación 3.5, si se realiza esta operación previamente, el resultado será de 8 bits obteniendo el resultado buscado.

Para el cálculo de  $H'$  y evitar usar números negativos se propone la ecuación 3.9. Esta se determinó debido a que dependiendo de los valores del pixel el segundo término para cada caso en la ecuación 3.6, puede tomar un valor positivo o negativo, y esta sustraería o añadiría del primer término para cada caso, por eso simplemente se reescribió la forma del segundo término para quedar de manera equitativa exceptuando de todo esto al primer caso ya que para  $D = 0$  resulta en una división indeterminada. Para finalmente calcular  $H$ , se necesita extraer el LSB de  $H'$  ya que el rango de  $H$  es  $[0, 360)$  y para poder representarse con 8 bits es necesario este descarte.

$$V = \max(R, G, B) \quad (3.7)$$

$$D = V - \min(R, G, B)$$

$$S = \begin{cases} 0 & \text{si } V = 0 \\ \frac{(255)D}{V} & \text{si } V \neq 0 \end{cases} \quad (3.8)$$

### 3. Desarrollo

---

$$H' = \begin{cases} 0 & \text{si } S = 0 \\ 0 + \frac{60(G-B)}{D} & \text{si } (V = R) \& (G > B) \\ 360 - \frac{60(B-G)}{D} & \text{si } (V = R) \& (B \geq G) \\ 120 + \frac{60(B-R)}{D} & \text{si } (V = G) \& (B > R) \\ 120 - \frac{60(R-B)}{D} & \text{si } (V = G) \& (R \geq B) \\ 240 + \frac{60(R-G)}{D} & \text{si } (V = B) \& (R > G) \\ 240 - \frac{60(G-R)}{D} & \text{si } (V = B) \& (G \geq R) \end{cases} \quad (3.9)$$
$$H = \frac{H'}{2}$$

#### 3.11.1.3. Módulo de cálculo de posición, ROI y fin de ROI

Este módulo es necesario para computar ROI, donde se define un bit que cuando es 1, significa que está dentro de ROI. Se trata de un módulo donde internamente se encuentran 2 contadores de pulsos, uno con habilitación y otro no, y además un circuito combinacional.

Uno de los contadores es el que calcula la coordenada horizontal  $X$  (definida en este trabajo como  $pX$ ), el pulso a contar es el reloj de video, la habilitación es la SVA y su reset es la SSH.

cursiva

El otro contador es el que calcula la coordenada vertical  $Y$  (definida en este trabajo como  $pY$ ), el pulso a contar es la inversion de SVA y su reset es la SSV.

Para computar el bit de ROI, es necesario hablar de los parametros  $Xm$ ,  $Xmy$ ,  $Ym$ ,  $Ymy$ ,  $Xp$ ,  $Yp$ ; los cuales simplemente son las coordenadas que representan los bordes de ROI, y la cantidad de pixeles del fotograma, donde significan lo siguiente:

- $Xm$ : Coordenada horizontal del punto izquierdo superior de ROI
- $Ym$ : Coordenada vertical del punto izquierdo superior de ROI
- $Xmy$ : Coordenada horizontal del punto derecho inferior de ROI

- $Y_{my}$ : Coordenada vertical del punto derecho inferior de ROI
- $pX$ : Coordenada horizontal actual
- $pY$ : Coordenada vertical actual
- $X_p$ : Número de pixeles en eje horizontal
- $Y_p$ : Número de pixeles en eje vertical

Los parametros de ROI deben cumplir las siguientes restricciones:

$$\begin{aligned} 0 \leq X_m < X_{my} < X_p \\ 0 \leq Y_m < Y_{my} < Y_p \end{aligned} \quad (3.10)$$

Y para el cálculo de el Bit de ROI descrita en la ecuación 3.11, solamente es necesario comparar las coordenadas actuales con los parametros de borde de ROI.

$$Bit_{ROI} = \begin{cases} 1 & \text{si } (X_m \leq pX < X_{my}) \& (Y_m \leq pY < Y_{my}) \\ 0 & \text{de otra manera} \end{cases} \quad (3.11)$$

$$FDR = \begin{cases} 1 & \text{si } (pX = X_{my}) \& (pY = Y_{my}) \\ 0 & \text{de otra manera} \end{cases} \quad (3.12)$$

### 3.11.2 Segmentación

La segmentación requiere de un AD y es la encargada de realizar la segmentación pixel a pixel y está compuesto por el siguiente módulo.

#### ✓ Módulo AD segmentador

Preparado para recibir las características computadas en la etapa anterior.

El módulo tiene como entradas los canales R, G, B, H, S, V, y Gray, donde todos ellos son de 8 bits y tiene solo una salida de 1 bit, el bit de segmentación llamado a partir de aquí como BDS. Es un circuito puramente combinacional por lo que no cuenta con registros sincronizados por el reloj de video, sin embargo estos están implementados entre capas lo que se podría decir que cuenta con

cursivas

### 3. Desarrollo

---

ellos para un manejo ordenado.

¿Porque 5 bins?

#### 3.11.3 Cálculo de histogramas y área de segmentación

En esta etapa se implementaron histogramas de 5 bins para cada canal computado en la primera etapa. Además de un módulo el cual se encarga de computar el área definida por la segmentación computada en la etapa 2.

##### ✓ Histogramas de pixeles segmentados

Para alimentar el AD para la clasificación de color es necesario el cálculo de histogramas de 5 bins para cada canal de color. En este módulo se tiene como entrada solamente el valor de 8 bits a clasificar, el reset y el reloj, y como salida 5 números de 15 bits donde se representan los 5 bins del histograma normalizado. Para ello se implementó en hardware el algoritmo 1, ~~el cual tiene un estilo de escritura del lenguaje C~~. El funcionamiento se basa por contadores con habilitación, donde su habilitación se determina por BDS y  $Bit_{ROI}$ , para al final habilitando así el contador que representa al bin correspondiente. Por otra parte, al recibir el flanco de bajada de FDR, se calcula el histograma normalizado y se activa el bit HL, el cual representa que el histograma está listo o calculado. Todo lo antes descrito sincronizado por el reloj de video y con un reset asíncrono implementado.

Los bordes determinados en la variable B, fueron los mismos indicados en el algoritmo para todos los canales, a excepción de el canal H, ya que el rango de H es [0, 180].

En esta etapa, se construyeron 7 módulos de histograma que a su vez como salida se obtienen 35 buses con los histogramas normalizados por canal. ~~Debido a que la señal HL es la misma para cada caso, se utiliza una solamente.~~

Todos los modulos tienen en comun la señal HL (lo que sea que signifique eso)

##### 3.11.3.1. Módulo de cálculo de área de segmentación

Este módulo tiene como entradas, a las señales BDS,  $Bit_{ROI}$ , reset y reloj de video, y como salida un número de 15 bits que corresponde al área normalizada. El funcionamiento del módulo se

**Algoritmo 1** Algoritmo para módulo de histogramas

---

```

B[6] = {0, 51, 102, 153, 204, 256}; // Bordes de los bins
cntpix = 0; //Contador de pixeles
Ct [5] = {0}; // Contadores de histogramas
Hs [5] = {0}; // Histogramas definidos
HL = 0; // Bit para indicar que está listo el histograma
si reset entonces
    Ct [5] = {0};
    Hs [5] = {0};
    cntpix = 0;
    HL = 0;
en otro caso
    si Flanco de subida (Reloj) entonces
        si BDS && BitROI entonces
            Leer Valor;
            para i=0 ; i<5 ; i++ hacer
                si (B[i] ≤ Valor) && (Valor < B[i+1]) entonces
                    | Ct[i] = Ct[i] + 1;
                fin
            fin
            cntpix = cntpix + 1;
        fin
    fin
    si Flanco de bajada (FDR) entonces
        para i=0 ; i<5 ; i++ hacer
            | Hs[i] = (Ct[i]*10000)/cntpix;
        fin
        HL = 1;
    fin
fin

```

---

### 3. Desarrollo

---

#### Algoritmo 2 Algoritmo para cálculo del área segmentada

---

```
cntpix = 0; //Contador de pixeles segmentados  
pixTot = (Ymy-Ym) * (Xmy - Xm); // Cálculo del área de ROI  
area = 0; //Área resultante  
si reset // SSV entonces  
    cntpix = 0;  
    area = 0;  
  
en otro caso  
    si Flanco de subida (Reloj) entonces  
        si BDS && BitROI entonces  
            | cntpix = cntpix + 1;  
        fin  
    fin  
    si Flanco de bajada (FDR) entonces  
        | area = (10000*cntpix)/pixTot;  
    fin  
fin
```

---

describe con el algoritmo 2, y está compuesto de un contador por habilitación, donde su habilitación es calculada por la operación  $BDS \&\& Bit_{ROI}$ , la cuál significa que si se está dentro de ROI, y además es un pixel segmentado, entonces se contabiliza el pixel. Al obtener un flanco de bajada de la señal FDR, se calcula el área normalizada. ~~Todo lo antes descrito sincronizado~~ por el reloj de video **Estos componentes están sincronizados** y con un reset asíncrono implementado.

#### 3.11.4 Clasificación de color y tamaño

En esta última etapa, se calcula la clasificación. De parte del clasificador de tamaño por medio de comparadores y en la parte de color por medio de un AD generado. Esta etapa tiene como salida un número de 8 bits compuesto por la señales BCC y BTC quedando compuesto el número de la siguiente manera:

- Bit 0: Bit de Clase de Color (BCC)
- Bit 2-1: Bits de clase de Tamaño (BCT)
- Bit 7-3: Bits constantes a 0, estos se dejan con fines de diseño estandar que se ven en la sección 3.12

### 3.12. Módulos de comunicación y gestión de respuesta

Este bus es denominado ~~comunicación~~ Respuesta del Módulo Clasificador o RMC, ~~este módulo~~ y contiene los siguientes bloques:

✓ **Módulo AD de clasificación de color** Éste módulo tendrá como entradas los 35 números de 15 bits correspondientes a los histogramas normalizados vistos en el módulo de la sección 3.11.3 y como salida un bit que corresponde a la clase de color llamado **Bit de Clase de Color** o BCC, donde su salida es interpretada como:

- 0: Color naranja
- 1: Color verde

sincronización por

Al tratarse de un circuito combinacional, no es necesaria ~~la adición de la señal de reloj~~.

✓ **Módulo clasificador de tamaño** El módulo tiene como entrada ~~únicamente~~ el área calculada por el modulo mostrado en la sección 3.11.3.1, y por salida un número de 2 bits llamado **Bits de Clase de Tamaño** o BCT, el cual es interpretada su salida como:

- 0: Tamaño chico
- 1: Tamaño mediano
- 2: Tamaño grande
- 3: Inválido

se emplea

Al tratarse de una simple comparación ~~la cual es tratada como~~ un circuito combinacional, ~~no es~~ ~~necesaria la adición~~ de la señal de reloj.

que no requiere sincronización mediante

## 3.12 Módulos de comunicación y gestión de respuesta

La capa de comunicación del proyecto, es la encargada de administrar las señales que provee el MCU, sincronizandolas con las internas del FPGA para finalmente regresar una respuesta, la cuál será la clasificación resultante, esto con el fin de realizar una clasificación de la manera más rápida posible. Cabe mencionar, que esta capa está regida por diferentes señales de reloj para diferentes

### 3. Desarrollo

---

etapas. El funcionamiento de cada módulo será explicado en esta sección.

✓ **Módulo de comunicación** Este módulo es el encargado de administrar el protocolo SPI, cuando el MCU transfiere una instrucción de proceso da un pulso al bit **Disparo de clasificación** (DDC) y transfiere el byte de RMC, una vez que el **Disparo de Clasificación Lista** (DCL) proveniente del módulo de secuencia de respuesta, este, transfiere el **Byte de respuesta del SPI** (BRSPI) al MCU.

✓ **Módulo de secuencia de respuesta** Éste módulo es el encargado de administrar la clasificación correctamente. Tiene como entrada las señales Reloj de video, reset, y reloj interno DDC, SSV, HL (de la sección 3.11.3) , y RMC , y como salida la señal de Inicio de clasificación (mostrado en la figura 3.17 como "Inicio") o también llamado INC, BRSPI, y DCL

El reloj interno mencionado, corresponde al del FPGA, siendo este de 150MHz. El funcionamiento del módulo, se muestra en el algoritmo3, el cuál está compuesto por una Máquina de estados finitos. La secuencia está controlada por el reloj interno, y con el, se encarga de que por medio la señal DLC, esperar un nuevo fotograma para ser contemplado en su totalidad, ya que puede ser que el disparo se realice a mediación del fotograma, es por eso que la secuencia espera uno nuevo, para ser procesado, y con el, se genera la señal para el inicio de la clasificación, y la correcta administración de la respuesta al módulo de comunicación.

El FPGA trabaja con una señal de reloj de 150 MHz (No es 200 Mhz?.. es pregunta)

**Algoritmo 3** Secuencia de respuesta

---

```

e = 0; //Estado actual
ef = 0; // Estado futuro
si Flanco de subida (Reloj interno) entonces
    switch(e)
        0: //Estado de reinicio
        INC = 0;
        BRSPI = 0;
        DCL = 0;
        ef = 1;
        1: //Estado en reposo
        INC = 0;
        DCL = 0;
        si DDC entonces
            | ef = 2;
        fin
        2: //En espera de un fotograma nuevo
        si SSV entonces
            | ef = 3;
        fin
        3: //En espera de finalizar la clasificación
        INC = 1;
        si HL entonces
            | Leer RMC;
            | ef = 4;
        fin
        4: //Mandar respuesta
        BRSPI = RMC;
        DCL = 1;
        ef = 1;

    fin
    si reset entonces
        | e = 0;
    en otro caso
        | si Flanco de subida (Reloj interno) entonces
            | | e = ef;
        fin
    fin

```

---



Creo que tus conjuntos deben nombrarse de la siguiente forma:

- \*\* Conjunto de datos de pixeles
- \*\* Conjunto de datos de histogramas
- \*\* Conjunto de datos de areas

# 4

Sugiero en adelante nombrar:

Dataset => Conjunto de datos

Train dataset => Conjunto de entrenamiento

Test => Conjunto de prueba

Testeo => Prueba

clips => Video

## Resultados y discusión

Un comentario con respecto a la velocidad:

\*\* Que se puede entender Baja, Media o Alta Velocidad (a lo mejor si fuera posible estimar la velocidad de la banda en m/s, o intentar estimar cuantas naranjas por segundo pasan seria mucho mejor)

Se presentan los resultados de cada etapa de clasificación implementada en el FPGA.

### ✓ Base del data set.

Para la toma del data set de imágenes del cual es la base del data set de pixeles e histogramas, se capturaron 10 clips de video, donde en la tabla 4.1, se denotan las velocidades de banda y el tipo de set al que pertenecen. Los clips fueron elegidos de manera aleatoria por la computadora.

Tabla 4.1. Descripción de velocidades y tipos de set para cada clip de video.

# de clip	Velocidad	Set
1	Baja	Entrenamiento
2	Baja	Entrenamiento
3	Baja	Entrenamiento
4	Media	Entrenamiento
5	Media	Test
6	Media	Test
7	Media	Entrenamiento
8	Alta	Test
9	Alta	Entrenamiento
10	Alta	Entrenamiento

Duración del CLIP

Es posible agregar las duraciones de los clips individuales  
\* No necesariamente un clip de larga duración implicaría que hay una gran cantidad de tomas de naranjas..

Se realizó una prueba generando un modelo empleando solamente el 50% del total del conjunto de entrenamiento

#### 4. Resultados y discusión

Para la prueba del modelo en su totalidad se realizaron X pruebas:

✓ Prueba 1. Se realizó una prueba con el data set completo.

✓ Prueba 2. Se realizó una prueba con el data set de entrenamiento al 50%.

✓ Prueba 3. Se realizó una prueba con el data set de entrenamiento al 30%. Mismo que el ant.

De los clips se obtuvo el data set de imágenes para el entrenamiento y el testeo los cuales se pueden observar en la tabla 4.2. Cabe destacar que el conjunto que componen las capturas de las pruebas 2 y 3, forman parte del conjunto de capturas de la prueba 1 y fueron seleccionadas de manera aleatoria, por lo que puede haber capturas que formen parte del mismo conjunto de capturas de la prueba 2 en el conjunto de capturas de la prueba 3. Para cada prueba, se probó con el mismo data set de Test, por lo que se puede ver que el conjunto de la prueba 2, es similar al conjunto de Test, por otra parte, el conjunto de la prueba 3, tiene menos capturas que el data set de Test.

Tabla 4.2. Distribución de capturas por prueba para el data set de imágenes.

Set	Capturas totales		Prueba 1		Prueba 2		Prueba 3	
	N	V	N	V	N	V	N	V
Entrenamiento	1159	211	1159	211	579	105	347	63
	1370		1370		684		360	
	60.42 %	11 %	60.42 %	11 %	47 %	8.52 %	36.22 %	6.57 %
Test	N	V	N	V	N	V	N	V
	475	73	475	73	475	73	475	73
	548		548		548		548	
	24.76 %	3.8 %	24.76 %	3.8 %	38.55 %	5.92 %	49.58 %	7.62 %
Total	N	V	N	V	N	V	N	V
	1634	284	1634	284	1054	178	822	136
	1918		1918		1232		958	
	85.2 %	14.8 %	85.2 %	14.8 %	85.55 %	14.45 %	85.8 %	14.2 %

En la tabla 4.3, se denota la distribución del data set de pixeles. En todas las pruebas, se puede ver que se muestra un overfitting en el entrenamiento, para los dos tipos de entrenamiento, es por eso que, para comprobar la efectividad del mismo, se simuló por computadora y FPGA el comportamiento de la segmentación, mostrada su efectividad por prueba en la tabla 4.4.

No es 100,000?

Se puede ver, que los data sets de pixeles por prueba, difieren al rededor de 10000 muestras cada uno, por lo que se esperaría un desempeño similar para cada modelo de AD para segmentación.

82 \* Creo que hace falta una figura que explique como concluyes que hay overfitting (acuerdate de tus clases con SAID)

## 4.1. Árbol de Decisión de segmentación

Tabla 4.3. Distribución de data set de pixeles por prueba obtenidos y sus resultados de entrenamiento en WEKA.

Set	Prueba 1		Prueba 2		Prueba 3	
	S	NS	S	NS	S	NS
Entrenamiento	101589	27359	92621	24326	84568	23110
	78.78 %	21.22 %	79.2 %	20.8 %	78.53 %	21.47 %
Muestras Totales	128948		116947		107678	
Presición 10-Fold	99.27 %		99.16 %		99.29 %	
Presición 66 %-34 %	99.21 %		99.14 %		99.35 %	

Poner explicitamente que significa S y NS? (Segmentados/No Segmentados)  
Para cada prueba, en muestras de pixeles no segmentados se muestra una distribucion por debajo del 22 %, esto es debido a las capturas, como se puede ver en la figura 3.11 de un color un poco más verde que el resto, donde el fondo siempre es el mismo para cada captura, por lo que se esperan pixeles bastante iguales en la parte no segmentada y por lo tanto un conjunto de muestras más amplio en la parte segmentada.

## 4.1 Árbol de Decisión de segmentación

En la tabla 4.4 se muestra la tabla de precisiones para el modelo de AD de segmentación, recordando que la prueba es solamente para el conjunto de imágenes que comprenden el set de Test, osease, un universo de pixeles totalmente nuevo para el modelo, para cada prueba. En ella también se muestra el rango de precisiones, es decir, la menor y mayor precisión obtenida, así como el promedio global.

Para la simulación por computadora, se tiene en la prueba 1 un rango de precisión por encima del 90 %, teniendo como resultado global, una precisión casi perfecta. Por otra parte, en las pruebas 2 y 3, también, se tuvo una buena precisión, sin embargo, la prueba 3, tuvo un mejor resultado. Esto puede ser debido al data set de imágenes de donde se tomó.

Para el desempeño en FPGA, en la prueba 1 los rangos de precisión baja notablemente, sin embargo la precisión global se mantiene alta en 95.08 %. Para la prueba 3, se tiene un rendimiento aceptable, con una precisión global por encima del 95 % obtenido en la prueba 1, por otra parte la prueba 2...

#### 4. Resultados y discusión

Mete un poco (un mucho) de ruido, sobre todo por las pruebas 2 y 3 en donde se parecen mucho los mínimos pero en la prueba 1 no es así.

Tabla 4.4. Precisión obtenida por plataforma de segmentación con AD por prueba.

	Precisión por computadora (%)	Precisión por FPGA (%)
Prueba 1	[91.32, 99.87] 99.21	[68.64, 99.83] 95.08
Prueba 2	[87.21, 99.86] 97.5	[87.08, 99.83] 97.52
Prueba 3	[88.53, 99.86] 98.18	[88.74, 99.77] 98.22

## 4.2 Árbol de Decisión de clasificación de color

En la tabla 4.5 se puede ver la cantidad de muestras tomadas para el entrenamiento. En para las tres pruebas, se puede ver que la distribución es casi idéntica al data set de imágenes, aunque con algunas muestras descartadas, debido a que se eliminaron las muestras duplicadas. El rendimiento del entrenamiento para cada prueba y para los dos modos de entrenamiento se alcanza una precisión cercana al 95 %, por lo que no se llega a un overfitting en ningún caso. Cabe destacar que estos valores son interesantes ya que se puede ver que con una menor cantidad de muestras el modelo se comporta muy similar.

Tabla 4.5. Descripción de data set de histogramas por prueba y sus resultados de entrenamiento en WEKA.

Set	Prueba 1		Prueba 2		Prueba 3	
Entrenamiento	N	V	N	V	N	V
	1151	210	578	104	346	63
	84.57 %	15.42 %	84.75 %	15.25 %	84.6 %	15.4 %
Muestras totales	1361		682		409	
Precisión 10-fold	95.73 %		95.30 %		94.62 %	
Precisión 66 %-34 %	96.11 %		93.10 %		96.40 %	

En la tabla 4.6 se muestra la tabla de precisiones obtenida para las pruebas realizadas por computadora y FPGA. Como era de esperar, las precisiones para cada prueba son muy similares a la precisión de entrenamiento de la tabla 4.5, obteniendo precisiones para las dos plataformas también cercanas al 95 %, lo que de manera muy marcada, se denota que con pocas muestras, el modelo de AD para la clasificación de color obtiene un resultado más que aceptable.

Tabla 4.6. Precisión por computadora y FPGA de modelo AD de clasificación de color.

	Prueba 1		Prueba 2		Prueba 3	
	N	V	N	V	N	V
Computadora	96.42 %	89.04 %	94.1 %	89.04 %	97.05 %	82.19 %
	95.43 %		93.43 %		95.07 %	
FPGA	93.38 %	91.89 %	97.26 %	85.14 %	97.26 %	75.68 %
	93.44 %		95.63 %		94.35 %	

## 4.3 Clasificación de tamaño

Para la clasificación de tamaño, primeramente se tomó un **data set** de imágenes, donde se clasificó manualmente cada **imagen**, la distribución se muestra en la tabla 4.7, dejando por tamaño el área aproximada para la distinción de las 4 clases correspondientes, cabe mencionar que no se tuvo ninguna muestra inválida por lo que se agregó posteriormente al **data set**, es por esto que no se reportan las muestras inválidas en la tabla.

Tabla 4.7. Distribución de **data set** de imágenes para clasificación por tamaño.

		Capturas por clase		
Set		Chico	Mediano	Grande
Entrenamiento	388	941	61	
	19.97 %	48.45 %	3.14 %	
	1390			
	71.57 %			
Test	131	394	27	
	6.74 %	20.28 %	1.4 %	
	522			
	26.87 %			
Total	519	1335	88	
	26.72 %	68.74 %	4.53 %	
	1942			

Al realizar el entrenamiento en WEKA para obtener un modelo de comparadores para la clasificación de tamaño se obtuvo en la prueba de 10-fold una precisión de **99.71 %** y por entrenamiento 66-34 se obtuvo una precisión de **99.57 %** respectivamente.

El árbol resultante, se describe en la figura 4.1, en donde se agrega la directiva de muestras

#### 4. Resultados y discusión

---

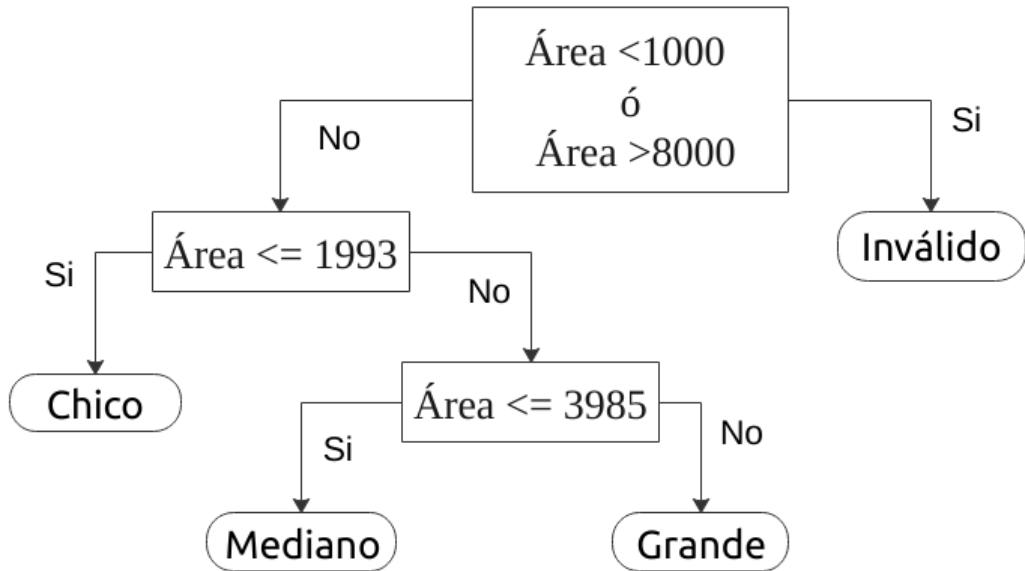


Figura 4.1. AD usado para clasificación de tamaño.

inválidas.

En la tabla 4.8 se muestra el resultado en FPGA de la precisión del árbol mostrado en la figura 4.1, sobre pasando un 90 % de precisión global. Como era de esperarse, la menor precisión fue obtenida en la clase Grande, ya que la distribución es poca en ese tipo de muestras.

Tabla 4.8. Distribución de data set de imágenes para clasificación por tamaño.

Clase	Chico	Mediano	Grande
Precisión en FPGA	86.26 %	93.15 %	77.78 %



Logico porque hay mas naranjas Medianas que grandes y chicas.

A ver si es posible que generes una tabla de confusión, en donde se muestre que clases se obtienen cuando los clasificadores se "equivocan" e intentar hacer un comentario con respecto a esta tabla

	Chicas	Medianas	Grandes
Chicas	90	15	5
Medianas	5	95	30
Grandes	7	6	98

## 4.4 Recursos utilizados

En la tabla 4.9 se muestra la distribución de recursos utilizados por la arquitectura, se utilizaron muy pocos slices como registros y además hay disponible más del 60% de slices como LUTs disponibles. Dados estos datos, se podría construir una doble linea de procesamiento trabajando en paralelo en un mismo dispositivo.

Tabla 4.9. Distribución recursos utilizados del FPGA.

Recurso	Usados	Disponibles	Utilización
Slices-Registros	764	184304	0.41 %
Slices LUT	31222	92152	33.9 %

discusion de los resultados?

\*\* Que se puede deducir de lo que reportas en la tabla

De los registros utilizados, que modulos se estima ocupan mas de estos recursos - cual ocupa menos?

\*\* Es posible optimizar

De los LUTs utilizados, que modulos se estima ocupan la mayor cantidad de LUTs? Cual modulo ocupa menos?

\*\* Es posible optimizar estos recursos



# 5

## Conclusiones

La clasificación de fruta asistida por sistemas embebidos es una forma de apoyo para la industria local de alimentos, esto para lograr una aceleración en la clasificación de la misma y lograr metas comerciales como la exportación y ventas en aumento.

un / FPGAs

permite una implementación de alto desempeño

El uso de FPGA para la clasificación por medio de una cámara ~~resulta una forma adecuada de implementación~~ de un clasificador de fruta, ya que permite un menor consumo de energía y paralelismo en la clasificación, ya que si el FPGA es suficientemente grande se pueden implementar múltiples líneas con el mismo dispositivo funcionando en paralelo.

es posible implementar

~~Ayudándose mediante modelos de aprendizaje supervisado, se pueden lograr tareas complejas como la segmentación y la clasificación de color mediante operaciones sencillas realizadas en paralelo.~~

Para ello se propuso una arquitectura de 4 etapas: la primera para extracción de características para segmentación, la segunda para el computo del modelo de segmentación, la tercera para extracción de características para clasificación de tamaño y color, y por último la etapa de clasificación de tamaño y computo del modelo de clasificación de color.

## 5. Conclusiones

---

Para este trabajo, el modelo utilizado es el árbol de decisión ya que por medio de la arquitectura propuesta, ésta puede ser computada por un circuito combinacional, ya que cada rama del árbol se computa en paralelo, obteniendo así, un computo orientado a obtener una respuesta en tiempo real.

representada mediante

La arquitectura propuesta en este proyecto, ~~su función es~~ clasificar por tamaño y color por medio de una segmentación de la fruta, en una escena tomada por una cámara colocada en una maquinaria que transporta fruta a través de la escena. La clasificación por color ~~es resuelta~~ mediante árboles de decisión, los cuales en los resultados muestran que ~~este modelo puede ser incluso~~ entrenado con pocas muestras y obtener resultados por encima del ~~90 %~~ para ambas tareas, obteniendo como resultado una precisión del 97 % para la segmentación, un 94 % para la clasificación de color y un 90 % para la clasificación de tamaño.

de precision

## Trabajos futuros

Se plantean pruebas en campo con la implementación propuesta en el proyecto, usando el mejor modelo obtenido en este trabajo, donde se realizarán pruebas con la maquinaria completa, siendo controlada por el MCU ~~en convivencia~~ con el FPGA, midiendo tiempo de clasificación que se estima rondar a los 16ms por muestra debido al muestreo de la cámara.

Se puede elaborar mas...

\* Que queda pendiente con respecto al arbol de decision?

\* Que se puede hacer con respecto a la inclusion de categorias intermedias de las frutas. Que tan generico es el trabajo para considerar otras clases de citricos o frutos que no necesariamente sean redondos??

EL MCU "convive con el el FPGA, se hacen sus carnes asadas y se toman unas caguamas??

\*\* CHECAR DATOS FALTANTES EN LAS REFERENCIAS. ANEXAR LOS ARTICULOS QUE LE SOLICITE EN EL CORREO ELECTRONICO!!!

## Bibliografía

Juan Perez, Pedro Lopez, Anacleto Menendes and Julio Gallego

- [1] S. Arivazhagan and R. Newlin Shebiah and S. Selva Nidhyanandhan and Lakshmanan Ganesan, "Fruit Recognition using Color and Texture Features," 2010. **Congreso/Revista?** **Paginas?**
- [2] A. Aguirre, M. Alarfaj, E. Li, J. F. Hernández-Sánchez, and S. Thoroddson, "Tomographic particle image velocimetry using smartphones and colored shadows," *Scientific Reports*, vol. 7, p. 3714, 06 2017.
- [3] L.-J. Kau and T.-L. Lee, "An efficient and self-adapted approach to the sharpening of color images," *TheScientificWorldJournal*, vol. 2013, p. 105945, 11 2013.
- [4] G. Ancillo, *Los ciótricos*. Valencia: Universitat de Valeónia, Jardíó Botaònic, 2014.
- [5] Food and A. O. of the United Nations, "Citrus fruit fresh and processed statistical bulletin." [www.fao.org/3/a-i8092e.pdf](http://www.fao.org/3/a-i8092e.pdf), 2016. **Hipervinculos**
- [6] SAGARPA, "National agricultural planning 2016-2030." [https://www.gob.mx/cms/uploads/attachment/file/257073/Potencial-C\\_tricos-parte\\_uno.pdf](https://www.gob.mx/cms/uploads/attachment/file/257073/Potencial-C_tricos-parte_uno.pdf), 2016.
- [7] Seema, A. Kumar, and G. S. Gill, "Automatic fruit grading and classification system using computer vision: A review," in *2015 Second International Conference on Advances in Computing and Communication Engineering*, pp. 598–603, May 2015.
- [8] J. Pablo Mercol, J. Gambini, and J. Miguel Santos, "Automatic classification of oranges using image processing and data mining techniques," *XIV Congreso Argentino de Ciencias de la Computación*, 04 2019.
- [9] Y. Sirisathitkul, N. Thumpen, and W. Puangtong, "Automated chokun orange maturity sorting by color grading," *Walailak Journal of Science and Technology*, vol. 3, 06 2006.

## BIBLIOGRAFÍA

---

- [10] A. Martínez-Usó, F. Pla, and P. García-Sevilla, "Multispectral image segmentation for fruit quality estimation," in *Proceedings of the 2005 Conference on Artificial Intelligence Research and Development*, (Amsterdam, The Netherlands, The Netherlands), pp. 51–58, IOS Press, 2005.
- [11] G. Feng and C. Qixin, "Study on color image processing based intelligent fruit sorting system," in *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788)*, vol. 6, pp. 4802–4805 Vol.6, 2004.
- [12] D. Unay and B. Gosselin, "Stem and calyx recognition on 'jonagold' apples by pattern recognition," *Journal of Food Engineering*, vol. 78, no. 2, pp. 597 – 605, 2007.
- [13] Y. A. Ohali, "Computer vision based date fruit grading system: Design and implementation," *Journal of King Saud University - Computer and Information Sciences*, vol. 23, no. 1, pp. 29 – 36, 2011.
- [14] S. Khoje, D. S. K. Bodhe, and A. Adsul, "Automated skin defect identification system for fruit grading based on discrete curvelet transform," 2013. **Congreso/Revista?**  
**Paginas?**
- [15] H. M. Zawbaa, M. Hazman, M. Abbass, and A. E. Hassanien, "Automatic fruit classification using random forest algorithm," in *2014 14th International Conference on Hybrid Intelligent Systems*, pp. 164–168, Dec 2014.
- [16] Woo Chaw Seng and S. H. Mirisae, "A new method for fruits recognition system," in *2009 International Conference on Electrical Engineering and Informatics*, vol. 01, pp. 130–134, Aug 2009.
- [17] A. Rocha, D. C. Hauagge, J. Wainer, and S. Goldenstein, "Automatic fruit and vegetable classification from images," *Computers and Electronics in Agriculture*, vol. 70, no. 1, pp. 96 – 104, 2010.

- [18] M. S. Hossain, M. Al-Hammadi, and G. Muhammad, "Automatic fruit classification using deep learning for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, pp. 1027–1034, Feb 2019.
- [19] M. A. Nuño-Maganda, Y. Hernández-Mier, C. Torres-Huitzil, and J. Jiménez-Arteaga, "Fpga-based real-time citrus classification system," in *2014 IEEE 5th Latin American Symposium on Circuits and Systems*, pp. 1–4, Feb 2014.
- [20] F. Saqib, A. Dutta, J. Plusquellic, P. Ortiz, and M. S. Pattichis, "Pipelined decision tree classification accelerator implementation in fpga (dt-caif)," *IEEE Transactions on Computers*, vol. 64, pp. 280–285, Jan 2015. Mayusculas
- [21] D. Wilking and T. Röfer, "Realtime object recognition using decision tree learning," in *RoboCup 2004: Robot Soccer World Cup VIII* (D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, eds.), (Berlin, Heidelberg), pp. 556–563, Springer Berlin Heidelberg, 2005.
- [22] Q. Li and A. Bermak, "A low-power hardware-friendly binary decision tree classifier for gas identification," *Journal of Low Power Electronics and Applications*, vol. 1, pp. 45–58, 12 2011.
- [23] E. Stattner, P. Hunel, N. Vidot, and M. Collard, "Acoustic scheme to count bird songs with wireless sensor networks," in *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1–3, June 2011. Minusculas
- [24] Y. SIRISATHITKUL, N. THUMPEN, and W. PUANGTONG, "Automated chokun orange maturity sorting by color grading," *Walailak Journal of Science and Technology (WJST)*, vol. 3, no. 2, pp. 195–205, 2011. Nombre de Revista?
- [25] G. Feng and C. Qixin, "Study on color image processing based intelligent fruit sorting system," vol. 6, pp. 4802 – 4805 Vol.6, 07 2004. Nombre de Revista?
- [26] G. Kay and G. de Jager, "A versatile colour system capable of fruit sorting and accurate object classification," in *Proceedings of the 1992 South African Symposium on Communications and Signal Processing*, pp. 145–148, Sep. 1992.

## BIBLIOGRAFÍA

---

- [27] X. Wang, C. Wu, and M. Hirafuji, "Visible light image-based method for sugar content classification of citrus," *PLOS ONE*, vol. 11, p. e0147419, Jan. 2016.
- [28] M. Hervas, R. Alsina-Pagès, F. Alías, and M. Salvador, "An fpga-based wasnt for remote real-time monitoring of endangered species: A case study on the birdsong recognition of botaurus stellaris," *Sensors*, vol. 17, p. 1731, 06 2017.
- [29] Q. Li and A. Bermak, "A low-power hardware-friendly binary decision tree classifier for gas identification," *Journal of Low Power Electronics and Applications*, vol. 1, pp. 45–58, 12 2011.
- [30] J. Gill, P. S. Sandhu, and T. Singh, "A review of automatic fruit classification using soft computing techniques," 2014. Revista o Congreso?  
Paginas?
- [31] H. M. Zawbaa, M. Hazman, M. Abbass, and A. E. Hassanien, "Automatic fruit classification using random forest algorithm," in *2014 14th International Conference on Hybrid Intelligent Systems*, pp. 164–168, Dec 2014.
- [32] H. Afrisal, M. Faris, G. P., L. Grezelda, I. Soesanti, and M. F., "Portable smart sorting and grading machine for fruits using computer vision," pp. 71–75, 11 2013. Revista o Congreso?
- [33] M. Makky and P. Soni, "Development of an automatic grading machine for oil palm fresh fruits bunches (ffbs) based on machine vision," *Computers and Electronics in Agriculture*, vol. 93, p. 129–139, 04 2013. Le sugiero quitar el mes como dato de publicacion de un libro, dejar solo año!
- [34] A. García Serrano, *INTELIGENCIA ARTIFICIAL. Fundamentos, práctica y aplicaciones*. 07 2012. Editorial? (Alfapato?)
- [35] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, apr 2017.
- [36] A. Criollo, L. Suárez Zambrano, and O. Oña, *Machine learning: Importance, advances, techniques and applications*. 04 2018. Editorial?

- [37] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.
- [38] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.
- [39] S. Salzberg, "C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993," *Machine Learning*, vol. 16, pp. 235–240, 1994.
- [40] J.-Y. Dufour, *Intelligent Video Surveillance Systems*. Wiley-ISTE, 2012.
- [41] A. Koschan and M. A. Abidi, *Digital Color Image Processing*. New York, NY, USA: Wiley-Interscience, 2008.
- [42] R. Bajaj and S. Fahmy, "Mapping for maximum performance on FPGA DSP blocks," vol. 35, pp. 1–1, 01 2015. Journal??
- [43] K. Jack, *Video Demystified: A Handbook for the Digital Engineer, 5th Edition*. USA: Newnes, 5th ed., 2007.
- [44] D. Bailey, *Design for Embedded Image Processing on FPGAs*. ~~01~~ 2011.
- [45] A. Kaehler, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, ~~jan~~ 2017.