INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

Diseño e implementación de un sistema en un solo chip para la navegación y reconocimiento de señales de tránsito en un sistema robótico móvil

ELECTRICAL AND ELECTRONICS ENGINEERING

Design and implementation of a system on chip for navigating and recognizing transit signals on a mobile robotic system

Juan J. Giraldo*, Eliud Estrada*, Diego F. Pineda *, Alexander López* §

**Universidad del Quindío, Programa de Ingeniería Electrónica, Grupo de Investigación GDSPROC.*

§ parrado@uniquindio.edu.co, jjgg1987@hotmail.com, elestrada44@hotmail.com, picadife1@hotmail.com

(Recibido: Diciembre 9 de 2010- Aceptado: Noviembre 19 de 2012)

Resumen

Dado que los accidentes de tránsito son una de las causales más relevantes de muertes a nivel mundial, este artículo pretende sustentar el diseño de un sistema en un solo chip (SoC), que tiene la capacidad de reconocer algunas señales de tránsito, obstáculos y las líneas blancas que delimitan una carretera en un escenario controlado, proporcionando una perspectiva a futuro para la implementación de pilotos automáticos en automóviles reales que ayuden a disminuir los índices de accidentalidad. El SoC opera sobre un FPGA (*Field Programmable Gate Array*) con el propósito que los algoritmos de alta velocidad de procesamiento sean implementados en hardware, y los algoritmos de baja velocidad sean implementados en software. Por lo tanto se desarrollaron diversos módulos hardware para la captura de video, pre-procesamiento de imágenes y control del balance de blancos para la captura de video. También se diseñaron algoritmos en software, tales como navegación, visión artificial estereoscópica y el reconocimiento inteligente de las señales de tránsito, conformando un conjunto de procesos que son administrados por un sistema operativo en tiempo real (RTOS). Los resultados del SoC se obtuvieron a partir de simulaciones de una plataforma robótica navegando en una carretera controlada y diseñada a pequeña escala.

Palabras clave: *FPGA, Procesamiento de imágenes, Redes neuronales artificiales, RTOS, SoC, Visión estereoscópica.*

Abstract

Due to the fact that the traffic accidents are one of the most relevant causes of deaths in the world, this article aims to expose the design of a System on a Chip (SoC) that has the capacity to recognize some transit signals, obstacles and white lines that delimit a road of a controlled scenery, giving a future prospect in order to implement automatic pilots in real automobiles for helping reduce accident rates. The SoC operates on an FPGA (*Field Programmable Gate Array*), so that the high speed processing algorithms are implemented in hardware and the low speed processing algorithms are implemented in software. Therefore different hardware modules were developed for video capturing, pre-processing images and controlling white balance in video capture. Also, software algorithms were designed, such as navigation, stereoscopy artificial vision and the intelligent recognition of the transit signals, shaping a set of processes administrated by a Real Time Operative System (RTOS). The SoC results were obtained through simulations in a robotic platform navigating on a road modeled on a small scale.

Keywords: *Artificial neural networks, FPGA, Image processing, RTOS, SoC, Stereoscopic vision.*

1. Introducción

Los accidentes de tránsito dejan cerca de 1.2 millones de muertes al año en todo el mundo. El exceso de velocidad, la violación a las normas de tránsito, el uso de sustancias psicoactivas y el consumo de licor resultan ser las principales causas de tan altos índices de accidentalidad (Bartley, 2008). Desde hace varios años las empresas ensambladoras de autos han procurado añadir sistemas que protejan la vida del pasajero, pero éstos únicamente reaccionan en el instante de una colisión, más no están en constante análisis del entorno que le permita al auto evitar algún accidente de tránsito. Un ejemplo de estos sistemas es el airbag, que solo se activa en el momento de un choque (Zini, 2004). En los últimos años ha aumentado el interés por incluir sistemas robustos en los automóviles, que logren dar información del entorno, como la cercanía de un carro que amenace con colisionar, el tipo de señalización en las zonas urbanas y/o rurales, la posición espacial en la geografía terrestre y la dirección de las líneas de una autopista que le faciliten la navegación al conductor del automóvil o inclusive sean la guía para que éste se maneje de forma automática (Team Berlin, 2007).

Conforme a lo anteriormente expuesto este trabajo presenta el diseño e implementación de un SoC sobre un FPGA para la navegación de un sistema móvil, que tenga la capacidad de guiarse por las líneas que delimitan la carretera, reconocer algunas señales de tránsito, detectar obstáculos en el camino y además lograr obtener una profundidad a la que se encuentren los obstáculos y las señales de tránsito. Las pruebas del SoC se realizaron en un escenario controlado y diseñado a escala sobre una plataforma robótica semejante a un automóvil que navega en una autopista. El SoC consiste en un complemento hardware/software, donde el hardware hace referencia a los módulos: procesador Nios II, comunicación serial RS 232 y JTAG (*Joint Test Action Group*), captura de estéreo imágenes, control de memoria SDRAM (*Synchronous dynamic random access memory*), PLL (*Phase lock loop*), pre-procesamiento de imágenes de video y balance de blancos automático, desarrollados a través de los lenguajes de descripción de hardware VHDL y Verilog. Por otra parte el software hace referencia a un RTOS encargado de administrar diversas tareas del SoC como la navegación de la plataforma robótica, el reconocimiento inteligente de algunas señales de tránsito a través de una Red Neuronal Artificial (RNA), y la ubicación de los centros de masa en un par de estéreo imágenes para el cálculo de la profundidad a través de métodos de triangulación (Pineda, et al., 2009).

2. Metodología

Esta sección se divide en 4 sub-secciones que describen en detalle cada uno de los módulos en hardware y aplicaciones en software que componen el SoC. La Figura 1 muestra un esquema general del SoC.

Los módulos hardware de captura y pre-procesamiento están diseñados para adquirir y filtrar la imagen de entrada, el módulo de procesamiento está diseñado para implementar sobre él, la aplicación software del SoC, esta aplicación, está basada en un sistema operativo de tiempo real uC/OS II de Micrium, para ejecutar las 4 tareas principales del RTOS.

2.1 Hardware de captura de imágenes

El módulo TRDB-DC2 proporcionado por Terasic (2006), es el hardware de captura de imágenes de video en tiempo real utilizado en el sistema, este módulo hardware está descrito en lenguaje Verilog y tiene como función convertir los datos crudos a la entrada del sistema, en datos de salida en formato de colores (RGB, *Red, Green and Blue*) y escribirlos en memoria (Pineda, et al., 2009).

2.2 Hardware de pre-procesamiento de imágenes

Este módulo permite filtrar las imágenes de entrada por medio de una técnica de transformación lineal y posteriormente realizar una binarización en cada una de sus componentes RGB (Gonzalez & Woods, 2002). Este módulo también implementa un balance de blancos automático que mantiene el nivel de brillo constante en la imagen. La Figura 2 muestra un diagrama de bloques correspondiente al sistema de pre-procesamiento.

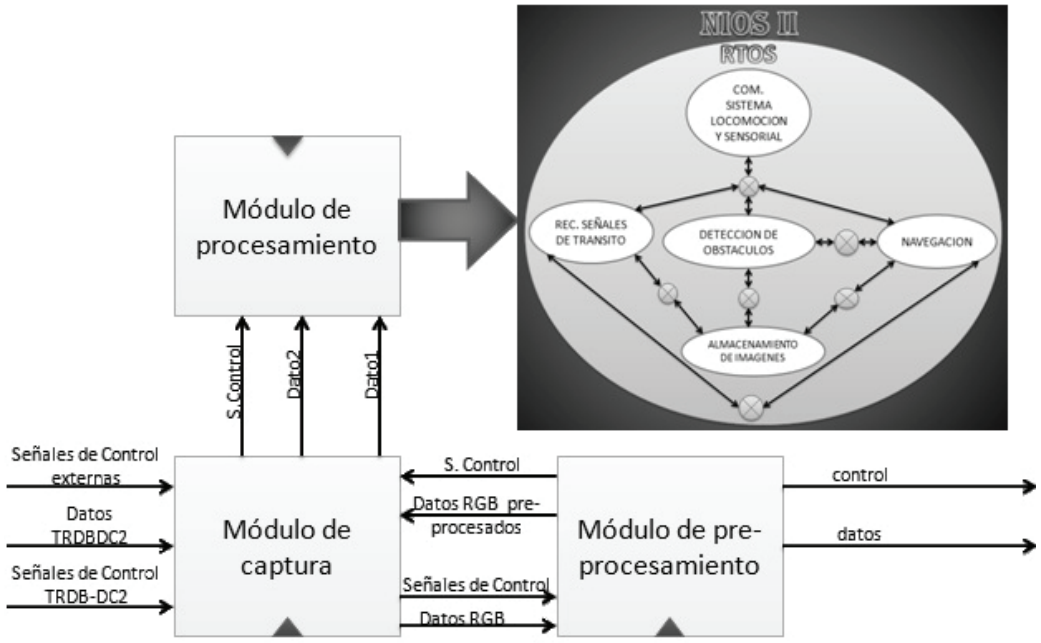

Figura 1. *Diagrama de bloques del SoC desarrollado.*

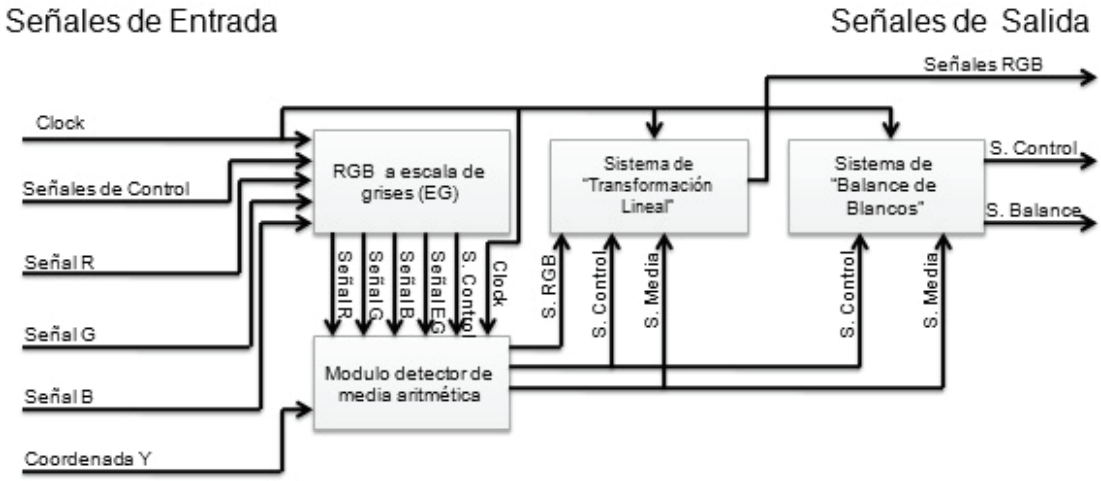

Figura 2. *Sistema de Pre-procesamiento*

2.2.1 Módulo conversor de RGB a escala de grises

Este sistema permite convertir la imagen a color capturada, en una imagen en escala de grises, la Ec. (1) muestra el procedimiento matemático de la conversión (Gonzalez & Woods, 2002).

$$A_{i,j} = 0.3*R_{i,j} + 0{,}59*G_{i,j} + 0{,}11*B_{i,j} \qquad (1)$$
$$0 \le i \le m - 1 \wedge 0 \le j \le n - 1$$

Donde **A** es una matriz de tamaño *n*m* y cada una de sus componentes corresponde al valor equivalente en escala de grises ubicado en la dirección (*i,j*). *R, G* y *B* corresponden a las componentes rojo, verde

$$\frac{77}{2^8} \approx 0.3 \Leftrightarrow 77 >> 8$$

$$\frac{75}{2^7} \approx 0.59 \Leftrightarrow 75 >> 7$$

$$\frac{29}{2^8} \approx 0.11 \Leftrightarrow 29 >> 8$$

y azul del pixel, que son señales de 10 bits. La Ec. (1) se puede implementar utilizando variables en

punto flotante, pero demanda muchos recursos para una implementación en HDL (*Hardware Description Language*), así que el camino más adecuado es encontrar estos valores con desplazamientos aritméticos y multiplicaciones de números enteros que representan cantidades en punto fijo, pues de esta manera se optimizan recursos y procesamiento en el sistema. Por lo tanto, a través de cálculos aritméticos, se encontró que el equivalente en punto fijo de la Ec. (1) corresponde a la Ec. (2).

Donde "$b >> n$" hace referencia a hacer un desplazamiento aritmético de n bits hacia la derecha del número b. Entonces, conforme a la Ec. (2) el módulo conversor de RGB a escala de grises (Pajares & Cruz, 2001) resulta ser un circuito combinacional, donde sólo operan multiplicadores, sumadores y desplazamientos aritméticos.

En la captura de imágenes se utilizaron ambos lentes del módulo TRDB-DC2, al igual que una tarjeta de desarrollo DE2-70, la cual almacena las imágenes en su chip SDRAM, posteriormente envía los datos mediante conexión USB a un computador donde son almacenadas en formato BMP (Pineda, et al., 2009).

2.2.2 Módulo detector de media aritmética

El objetivo de este módulo es entregar un valor de media aritmética para cada cuadro válido de la imagen en escala de grises, para que los datos del siguiente cuadro de la imagen a color, sean filtrados de acuerdo a este valor de media. Como la cámara logra adquirir 15 cuadros/segundo, se puede afirmar que dos cuadros consecutivos tienen características casi iguales, por lo tanto la referencia para realizar la transformación lineal y el balance de blancos a cada cuadro es la media aritmética del cuadro anterior. La media aritmética se calcula usando la Ec. (3).

Se puede observar que la Ec. (3) es un promedio, pues X se obtiene de sumar todos los pixeles de la imagen A y dividir entre el tamaño de la imagen. Se generará un valor de X real pero no entero, por lo tanto, es necesario hacer una aproximación similar a la utilizada en el caso del conversor de RGB a escala de grises por medio de desplazamientos. Como la cantidad de brillo que interesa conocer de la imagen es únicamente la correspondiente a la mitad inferior de la misma, ya que es la zona donde está la información útil de la carretera y la señal de tránsito, se limita $n = 512$ que es el número de filas de media imagen capturada y $m = 1280$ que es el número de columnas de la imagen, pues es donde estaría ubicada la vía de navegación para el sistema móvil. Así que utilizando un conjunto de desplazamientos y multiplicaciones se puede transformar la Ec. (3) en la Ec. (4), teniendo como aproximación que $\frac{1}{5} \cong \frac{51}{2^8}$.

El detector de media aritmética resulta ser un sistema en lazo abierto no combinacional, que posee 3 registros en cascada, así que las señales de salida estarán retrasadas tantos ciclos de reloj como registros en cascada tenga el sistema. Aunque sólo se calcula la media aritmética para la mitad de la imagen, el circuito tarda 1 ciclo de reloj por cada pixel más 3 ciclos de latencia, en total la media aritmética tarda 1280x512+3 ciclos de reloj.

2.2.3 Módulo de transformación lineal

El método de transformación lineal consiste en cambiar los pixeles de las matrices R, G y B de la imagen a color mediante una función lineal de la forma Y= mX + b, este método es frecuentemente usado para aumentar el rango dinámico de niveles de intensidad en imágenes con poco contraste (Gonzalez & Woods, 2002).

$$\mathbf{A}_{i,j} = ((77 * R_{i,j}) >> 8) + ((75 * G_{i,j}) >> 7) + ((29 * B_{i,j}) >> 8) \quad (2)$$
$$0 \leq i \leq m-1 \wedge 0 \leq j \leq n-1$$

$$X = \frac{\sum_{j=0}^{n-1} \sum_{i=0}^{m-1} (A_{i,j})}{n*m} \quad (3)$$

$$X = \frac{\sum_{j=0}^{n-1}\sum_{i=0}^{m-1}(A_{i,j})}{512*1280} = \frac{\sum_{j=0}^{n-1}\sum_{i=0}^{m-1}(A_{i,j})}{2^{17}(2^2+1)} = \frac{\sum_{j=0}^{n-1}\sum_{i=0}^{m-1}(A_{i,j})}{2^{17}} * \frac{1}{5}$$

$$X = \frac{51*\sum_{j=0}^{n-1}\sum_{i=0}^{m-1}(A_{i,j})}{2^{25}} = (51*\sum_{j=0}^{n-1}\sum_{i=0}^{m-1}(A_{i,j})) >> 25 \quad (4)$$

Como la captura de imágenes se realiza de manera frontal en dirección de una carretera como lo muestra la Figura 3b, la característica principal para elegir el tipo de pendiente es la media aritmética, ya que fue hallada a partir de la información del brillo que hay en la mitad inferior de la imagen, o sea conforme al nivel de brillo de la carretera. Quiere decir que la media aritmética hallada en la Ec. (4) es el corte en X de la ecuación punto pendiente, como lo muestra la Figura 3a.

Se puede observar que la Figura 3a consta específicamente de 2 pendientes, una pendiente igual a cero en $Y = 0$ y una pendiente mayor que 1 que inicia en $X = \text{media aritmética}$ hasta el nivel máximo del pixel, por lo tanto eleva los niveles de brillo y de esta manera, la imagen adquiere mayor contraste. El objetivo de aumentar el contraste de la imagen es obtener una frontera amplia entre los niveles bajos y altos de la imagen, para que al binarizar, las características más importantes se resalten. Teniendo como referencia 2 puntos $p1 = (a,0)$ y $p2 = (2^{10}, 2^{10})$ de la pendiente que corta en X, la ecuación punto pendiente que se obtiene y su aproximación en punto fijo del modelo lineal mostrado en la Figura 3a se describe a continuación en la Ec. (5).

La Ec. (5) punto pendiente depende de la media aritmética calculada, en este caso la variable ***a*** es quien la representa y ***x*** es el pixel actual a modificar. La implementación de esta ecuación en punto fijo se compone de multiplicadores, sumadores y desplazamientos. Luego de la transformación lineal que se puede visualizar en la Figura 3c, el proceso de binarización es sencillo, se toma como umbral la media aritmética *a*, donde los valores inferiores sean cero y los superiores sean 1023. Se usa un comparador para binarizar el pixel de salida del sub-módulo de transformación lineal y dos registros de salida que cumplen la función de sincronizar el proceso con el reloj del sistema *clk* (Chu, 2006). La Figura 3d muestra la imagen luego de ser binarizada en las tres componentes RGB, se puede observar que las señales de tránsito y las líneas de la carretera se resaltan al máximo (Pajares & Cruz, 2001).

2.2.4 Hardware de balance de blancos

La luminancia o brillo es una característica muy importante en el procesamiento de imágenes, además de la fuente de luz (sol, luz artificial, entre otros), el brillo depende del nivel de apertura del lente de captura, pues son directamente proporcionales. El objetivo del módulo de balance de blancos es mantener

$$m = \frac{(2^{10})}{(2^{10}) - a} = 2*\frac{(2^9)}{2^9 + 2^9 - a} = 2*\frac{1}{1-(\frac{a}{2^9}-1)}, \text{como}: \sum_{n-0}^{\infty} x^n = \frac{1}{1-x}\ para: |x|<1 \Rightarrow$$

$$2*\frac{1}{1-(\frac{a}{2^9}-1)} = 2*\sum_{n-0}^{\infty}(\frac{a}{2^9}-1)^n \Rightarrow si, n = 2 \Rightarrow m \cong \frac{a}{2^9} + (\frac{a}{2^9}-1)^2 \Rightarrow \quad (5)$$

$$y = m(x-a) = (\frac{a}{2^9} + (\frac{a}{2^9}-1)^2)(x-a)$$

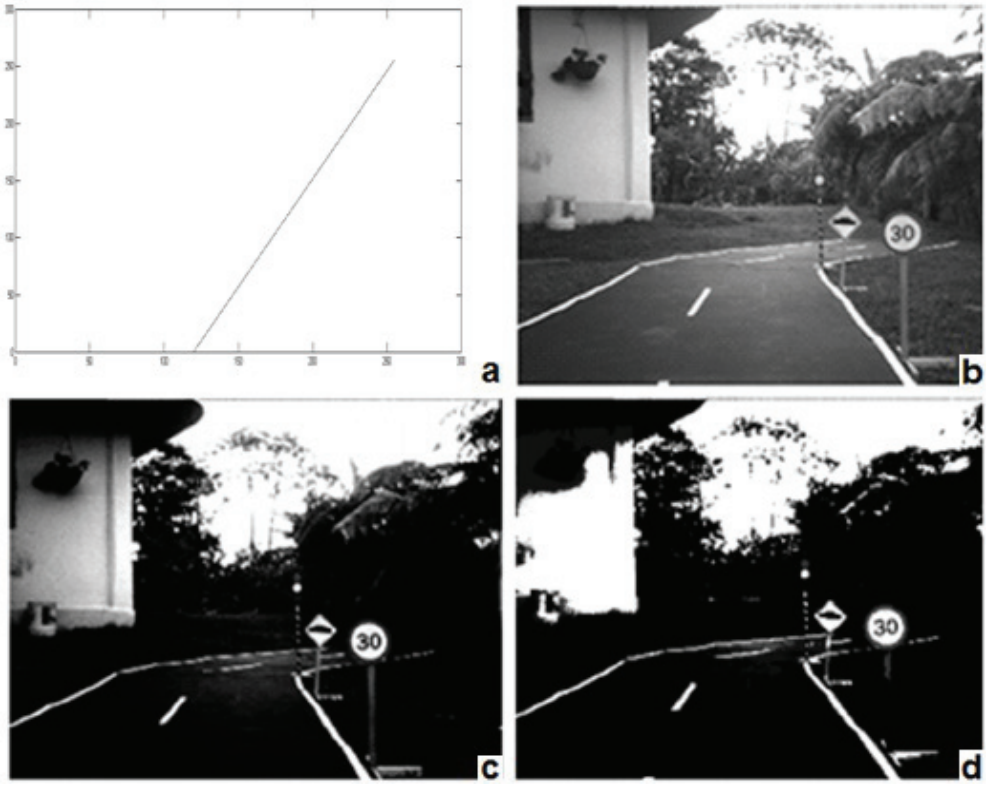

Figura 3. *Pre-procesamiento de la imagen en VHDL, a) pendiente para realizar transformación lineal, b) imagen capturada por el módulo TRDB-DC2, c) imagen luego de someterse a la transformación lineal, d) imagen binarizada en colores.*

constante la cantidad de luz en la imagen, evitando así, el incremento de ruido por el exceso de brillo o por el contrario, la pérdida de información por falta del mismo. El diseño consiste en un controlador difuso (Passino & Yurkovich, 1997; Pineda, et al., 2009), el cual está descrito en una máquina de estados, donde cada estado suma o resta diferentes niveles de apertura del lente del dispositivo TRDB-DC2, dependiendo de la diferencia existente entre una referencia de brillo y la señal de salida.

2.3 Hardware de procesamiento de imágenes

Este módulo proporciona las herramientas de hardware necesarias para realizar las tareas de reconocimiento, navegación y comunicación, este módulo se diseñó con la herramienta *SOPC Builder* de Altera Corporation (2009). *SOPC Builder* está basado en un entorno gráfico para crear diseños de *sistemas en chip programables (SOPC)* basados en el procesador RISC de 32 bits Nios II. Adicionalmente se permite incluir interfaces de memoria, periféricos estándar, DSPs, aceleradores hardware y periféricos personalizados. *SOPC builder* construye un módulo que instancia estos componentes y genera el sistema necesario para interconectarlos. La Figura 4 muestra el diagrama de bloques correspondiente al módulo de procesamiento diseñado en *SOPC builder*.

Los componentes utilizados para este sistema fueron el Nios II, que es una herramienta útil para aplicaciones embebidas, desarrollado por Altera. Un módulo Cmos, implementado por el Sr. Clare Hampson (Terasic, 2006a), el cual consiste en un puerto de entrada/salida de datos de 16 bits que permite el acceso a los dos sensores CMOS del dispositivo TRDB-DC2, además proporciona 3 salidas de control; s*tart:* señal de inicio de captura, e*nd*: señal de fin de la captura y *read*: señal de sincronización de lectura de datos con la SDRAM. Las señales *start* y *end* de este módulo fueron bastante útiles cuando se capturaron imágenes para ser almacenadas en formato BMP en diversas dimensiones, pero para el caso particular del prototipo robótico se obtuvieron imágenes de 160x240 con el fin de realizar el procesamiento

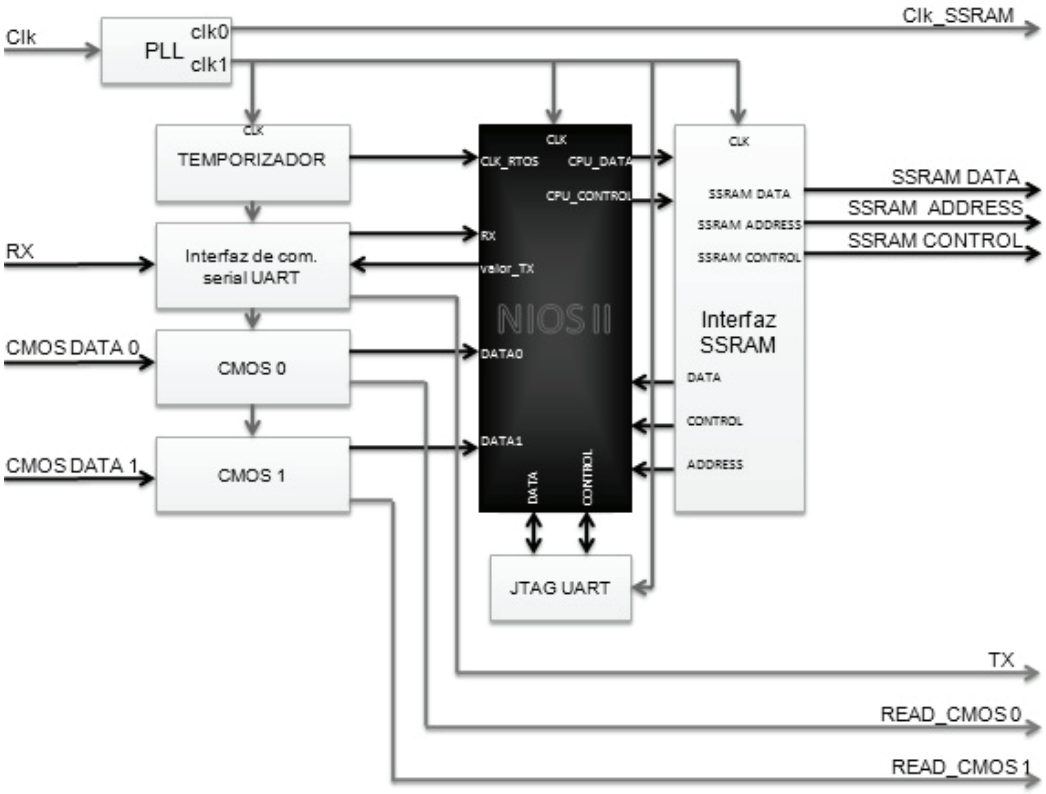

Figura 4. *Módulo de procesamiento de imágenes en tiempo real.*

en tiempo real, dado que las imágenes devengan un gran espacio en memoria y sobre todo tiempo en procesamiento. Posteriormente con estas mismas imágenes se pudo entrenar una RNA para reconocimiento de las señales de tránsito. La señal *read* se utilizó tanto en la captura y almacenamiento de imágenes en formato BMP como en la aplicación en software del SoC, ya que sincroniza automáticamente los datos con el controlador SDRAM haciendo de la lectura de datos una función transparente para el diseñador. Los módulos Interfaz de comunicación serial *UART,* PLL, Temporizador, JTAG *UART* e interfaz para *SSRAM,* fueron útiles en el diseño para establecer comunicación y controlar dispositivos externos.

2.4 RTOS del sistema

Para este trabajo, se utilizó el núcleo de tiempo real uC/OS II suministrado por Micrium (Labrosse, 2002), este elemento proporciona una organización en toda la aplicación, además permite sincronizar los procesos y el acceso a variables compartidas, también ofrece ventajas al tener un planificador expulsivo basado en prioridades, brindando desempeño en tiempo real a todas las tareas de la aplicación. El esquema de la aplicación software del sistema se puede visualizar en el recuadro negro de la Figura 1, con el nombre NIOS II y un subconjunto RTOS que forma parte del SoC. En esta Figura, las elipses blancas describen las tareas de la aplicación y los círculos con la X en medio describen los semáforos que sincronizan el acceso a las variables compartidas. Las tareas implementadas se describen en detalle a continuación:

2.4.1 Comunicación con el sistema de locomoción y sensorial

Esta tarea se encarga de enviar las órdenes desde la aplicación del software hasta el sistema de locomoción. La tarea recibe datos de la tarea *navegación* acerca de la dirección por la cual debe navegar el prototipo mecánico, de la misma manera, la tarea de *reconocimiento de las señales de tránsito y detección de obstáculos*, le entregan datos acerca de la posición y tipo de señal de tránsito o el obstáculo que detecten, en caso que no lo hagan, estas tareas envían un dato

diferente, indicando la **no** existencia de señal u obstáculo (Pineda, et al., 2009).

2.4.2 Almacenamiento de imágenes

Esta tarea es la encargada de realizar copias de las imágenes pre-procesadas en hardware y enviarlas a cada una de las tareas que las utilizan, en total la tarea almacena tres copias del sensor 2 del dispositivo TRDB-DC2 y dos copias del sensor 1 del mismo dispositivo. Esta tarea se encarga de escribir varias copias de cada cuadro capturado por el sistema hardware. Cada una de estas copias tiene asociado un semáforo que sincroniza el acceso exclusivo por cada una de las tareas que las solicita (Pineda, et al., 2009).

2.4.3 Navegación

Este algoritmo utiliza la línea blanca del lado derecho de la carretera para guiarse, para esto se utiliza sólo un pequeño segmento de la imagen como se observa en la Figura 5a. En el segmento se hallan dos puntos que describen la pendiente de la línea blanca, la combinación de información entre los puntos y la pendiente proporcionan información precisa acerca de la posición del dispositivo móvil dentro de la carretera, en la Figura 5b se muestran los puntos hallados por el algoritmo.

Los pixeles que apuntan las flechas en la Figura 5b, corresponden a los puntos característicos que describen el sentido de la línea blanca, con esta información se diseña un controlador difuso (Passino & Yurkovich, 1998; Pineda, et al., 2009) que envía una señal de salida dependiendo del error existente entre las características detectadas en cada imagen procesada y la referencia propuesta por el diseñador, dicha salida tiene 4 posibles valores que describen la dirección correcta que debe tomar el prototipo móvil para navegar correctamente dentro de la carretera como son : adelante, atrás, izquierda o derecha.

2.4.4 Visión estéreo en la detección de obstáculos y el reconocimiento de las señales de tránsito

Estas dos tareas presentan un procesamiento igual, con respecto a la visión estereoscópica, ambas tareas ubican el centro de masa de los objetos de interés en la imagen, la tarea de ubicación de obstáculos localiza los objetos que obstaculizan la navegación y la tarea de reconocimiento de las señales de tránsito ubica las señales de tránsito. Como el sistema captura dos imágenes donde las cámaras presentan ejes ópticos paralelos con longitud focal de 4,8mm y una distancia inter-ocular de 3,7cm entre sí, es posible obtener la profundidad de los objetos o señales de tránsito calculando la disparidad (Pajares & Cruz, 2001; Sabogal, 2007; Pineda, et al., 2009) que es igual a la diferencia entre los centros de masa de las imágenes de cada lente, la Ec. (6) muestra la función obtenida en la práctica que describe la variable profundidad.

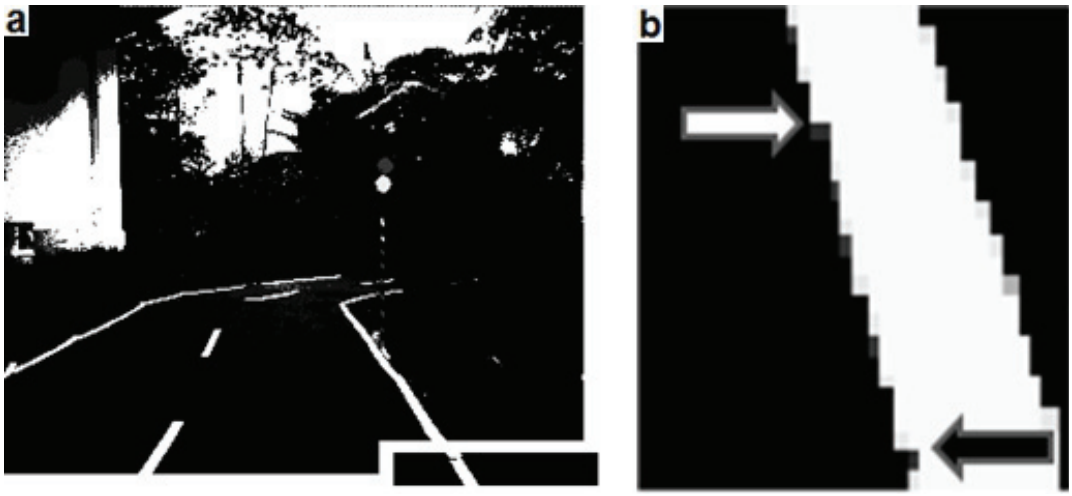

Figura 5. *Puntos para hallar la pendiente de la línea.*

$$disp = xi - xd \Rightarrow prof = \left(\frac{177.6}{disp + 4} \right) *5.8823 - 54.5882 \qquad (6)$$

2.4.5 Detección de obstáculos

La detección de obstáculos es una tarea que se encarga de ubicar el centro de masa de los objetos que obstaculicen la ruta de navegación del prototipo e identifica todo objeto que fue binarizado en el pre-procesamiento en hardware, se debe tener en cuenta que la binarización fue en colores, o sea que la tarea ubica todo objeto que sea rojo, verde, azul, magenta y el resto de combinaciones binarizadas en el plano RGB.

2.4.6 Reconocimiento de las señales de tránsito

Esta tarea se encarga de extraer las características que son únicas de cada señal de tránsito, para realizar una posterior clasificación a través de RNAs (Graupe, 2007). *La extracción de características de las señales de tránsito* consiste en acotar la zona alrededor del centro de masa de la señal y realizar un conteo de pixeles del color rojo y amarillo, ya que son los colores sobresalientes en la mayoría de las señales de tránsito, este conteo se hace en dirección de las filas y en dirección de las columnas, la información del conteo sobre cada fila y columna se almacena en vectores de igual tamaño a la cantidad de filas y columnas de la imagen respectivamente, en la Figura 6 se puede ver una señal de semáforo que indica detenerse (semáforo en rojo) y la zona acotada en la que se realiza el conteo de pixeles. En este caso la información es del color rojo y no existe información del color amarillo (Pineda, et al., 2009).

La información de la cantidad de pixeles en dirección de las filas y en dirección de las columnas genera una "*firma"* que diferencia a cada señal de tránsito de las otras, esta información de la "*firma*" se normaliza dentro de dos vectores patrón de 320 posiciones, cada vector patrón tiene un tamaño constante y hace referencia al color de la señal, o sea un vector patrón rojo para las señales reglamentarias y semáforo en rojo, y un vector patrón amarillo para las señales preventivas y semáforo en amarillo. La Figura 7a y 7b muestran una señal reglamentaria y el vector patrón respectivamente, este vector patrón presenta la información del color rojo en dirección de las columnas en las primeras 100 posiciones, la información del color rojo en dirección de las filas en las siguientes 100 posiciones, la información del color negro al interior de la señal en dirección de las columnas en las siguientes 60 posiciones y las últimas 60 posiciones con la información de ese mismo color negro, pero en dirección de las filas, así que estas últimas 120 posiciones representan la reglamentación de disminuir velocidad a 30. El eje vertical esta normalizado entre un rango de -4 y 4.

Las señales preventivas se caracterizan por ser amarillas y por contener en su interior información de color negro acerca de la prevención, así que el vector patrón que se llena en este caso es el vector patrón amarillo, como se muestra en la Figura 7d, sus primeras 100 posiciones contienen información del color amarillo en dirección de las columnas, las siguientes 100 posiciones con información del color amarillo en dirección de las filas, y al igual que la señal reglamentaria sus últimas 120 posiciones contienen la información del color

Figura 6. *Zona de acotamiento para extraer características de la señal de semáforo.*

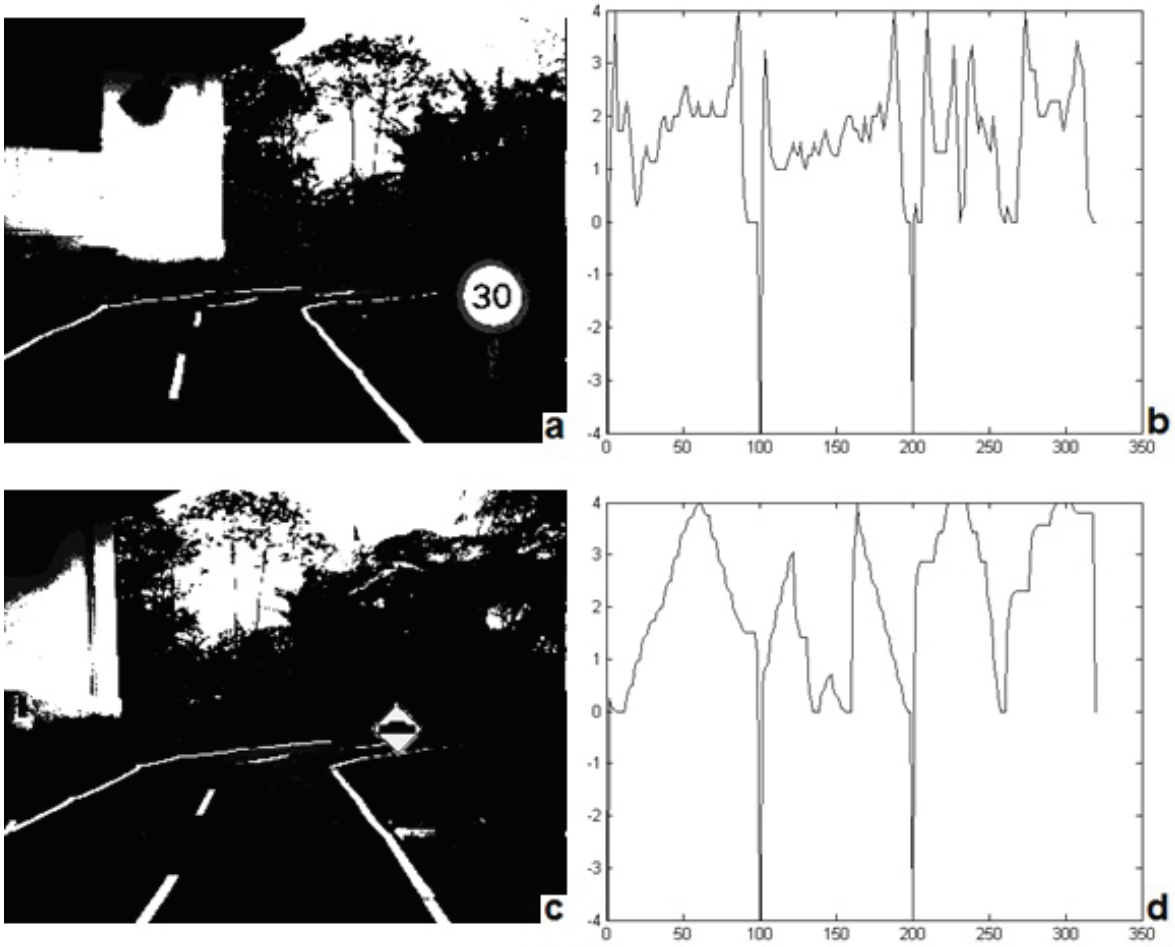

Figura 7. *a) Señal reglamentaria y b) su vector patrón obtenido. c) Señal preventiva y d) su vector patrón obtenido.*

negro que representa la señal preventiva, para el caso de la Figura 7c un resalto. Es de aclarar que cuando la señal presenta información de un solo color, como por ejemplo la señal preventiva, entonces el vector patrón amarillo será el único que se llene y el vector patrón rojo estará vacío, pero si fuese el caso de un semáforo en rojo y amarillo entonces ambos vectores patrón tendrían información.

La clasificación de las señales de tránsito se realiza gracias a una RNA. Luego de capturar con la cámara del sistema alrededor de 100 fotografías de las señales en el escenario a diversas distancias, se realizó un entrenamiento en Matlab de dos RNAs con una capa oculta y una capa de salida, la capa oculta está conformada por 5 neuronas y una función de activación *tansig* (tangente sigmoidal hiperbólica), la capa de salida está conformada por 2 neuronas y una función de activación *purelin* (la salida es igual al valor de entrada) a la salida, el patrón de entrada tiene 320 características y la salida es de 2 posiciones (Pineda, et al., 2009) como se visualiza en la Figura 8, el tipo de entrenamiento usado fue el *backpropagation* (Graupe, 2007).

Una red propaga el vector patrón rojo y la otra propaga el vector patrón amarillo. Luego del entrenamiento, fue necesario convertir la información de las redes en Matlab a un archivo en lenguaje C, se usó una herramienta útil desarrollada como tesis de pregrado (Quintero, 2004), del programa de ingeniería electrónica de la Universidad del Quindío, esta herramienta es una librería para Matlab y lenguaje C llamada *"jannl"* que permite convertir a lenguaje C descripciones de RNAs hechas en Matlab y permitiendo incluir cada RNA en el Programa principal del SoC.

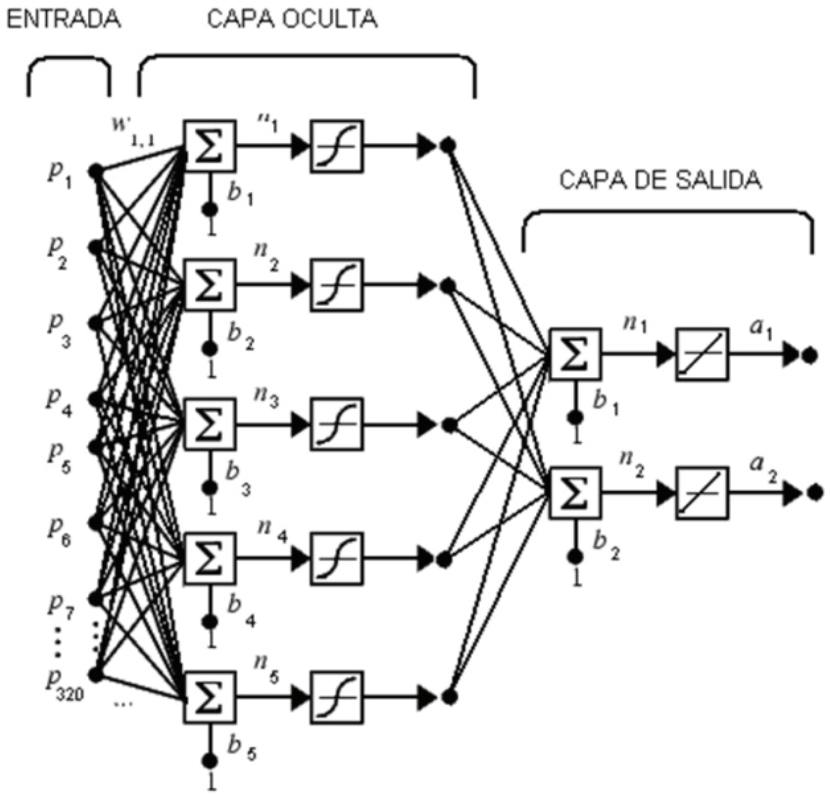

Figura 8. *RNA implementada en el sistema con una capa oculta y una de salida.*

3. Resultados y discusión

Los resultados del SoC fueron obtenidos a partir de pruebas sobre una plataforma robótica que navega sobre una carretera modelo, basada en una autopista de doble carril unidireccional, con longitud de 2.2 m y su ancho de 1.06 m, las líneas de los extremos de la vía son continuas y las del centro son segmentadas, todas estas con un ancho de 2.25 cm, así como se muestran en las Figuras 3, 5, 6 y 7. El prototipo robótico de prueba está conformado por una plataforma mecánica que se asemeja a un auto de fórmula 1, en la cual se encuentran contenidos una tarjeta DE2-70 basada en FPGA y un sistema embebido basado en microcontrolador, dedicado al manejo de los motores actuadores y a recibir órdenes del SoC por medio de una conexión serial asíncrona. El módulo TRDB-DC2 es ubicado en la parte trasera del robot para la captura constante de imágenes. La fuente de potencia para el funcionamiento general de todo el sistema la conforman baterías con celdas de litio para dar completa autonomía al sistema, éste se puede visualizar en la Figura 9.

El hardware del sistema se sintetizó sobre un FPGA Cyclone II, y se utilizó menos del 10% de los recursos lógicos de éste. Al sintetizar en hardware las etapas de *captura* y *pre-procesamiento*, se minimizó el costo computacional y se optimizó el funcionamiento del sistema en general, ya que el hardware diseñado usa bloques fragmentados que explotan el paralelismo y presentan baja latencia (Chu, 2006), además estas etapas son diseñadas para el funcionamiento específico del SoC y de los periféricos que en él se conecten. La herramienta SOPC Builder de Altera utilizada para generar la etapa de *procesamiento*, permite al diseñador implementar todas las herramientas necesarias para el funcionamiento de las aplicaciones en software y proporciona las interfaces necesarias para interactuar con los periféricos necesarios en el SoC.

Se realizó un análisis de planificabilidad, para decidir las prioridades y tiempos de ejecución de las tareas del RTOS que conforma el SoC diseñado y se midieron los tiempos de ejecución de cada una de las tareas. Es de aclarar que la

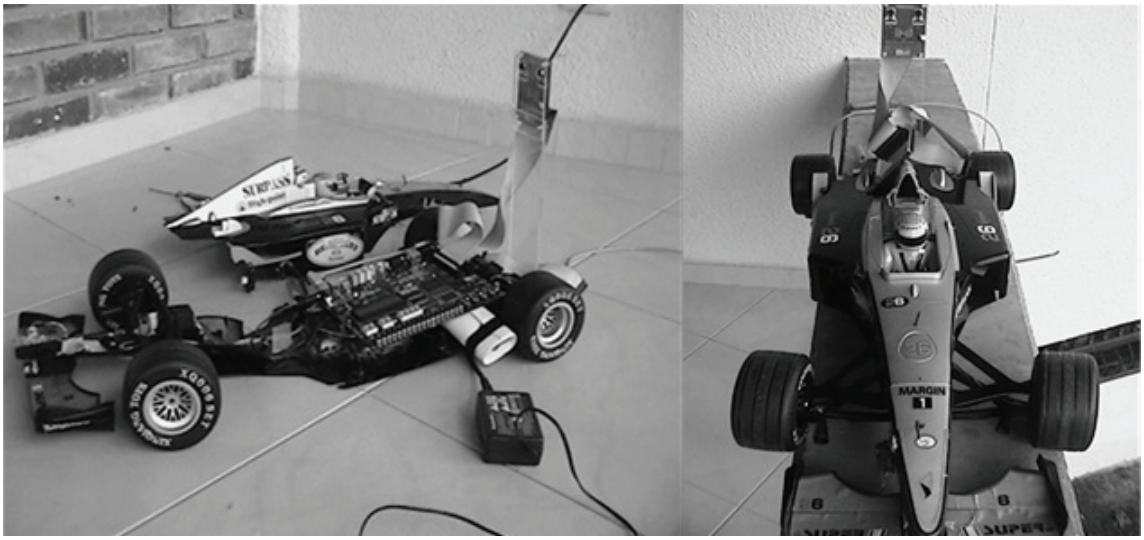

Figura 9. *Prototipo Robótico basado en FPGA y microcontrolador.*

frecuencia máxima con la que se configuró el modulo TRDB-DC2 para la captura de imágenes fue de 15 cuadros/segundo, a continuación se muestran los resultados de las tareas del RTOS:

Tarea de almacenamiento de imágenes en memoria de datos: esta tarea es imprescindible para el sistema, pues en ella se realiza el almacenamiento y copia de las imágenes necesarias para cada tarea del RTOS, el tamaño de las imágenes de copia en este caso es de 160x240, ya que el procesamiento de imágenes en software implica mucho tiempo de computo, así que se sacrifica significativamente la resolución para mejorar tiempo de procesamiento. Esta tarea limita el sistema de navegación a una frecuencia de trabajo máxima de 5.917 cuadros/segundo. Su tamaño en pila fue de 4096 bytes, el tiempo de cómputo igual a 169 ms, con periodo de ejecución de 500 ms y primer (1er) prioridad del RTOS.

Tarea de comunicación serial con el sistema de locomoción y sensorial: esta tarea se encarga de mantener la comunicación con un microcontrolador dedicado al movimiento del prototipo Robótico, es una tarea muy veloz y útil para enviar y recibir las palabras de control. Su tamaño en la pila fue de 2048 bytes, el tiempo de cómputo igual a 1 ms, con periodo de ejecución de 50 ms y segunda (2da) prioridad del RTOS.

Tarea de toma de decisiones para la navegación: el diseño del algoritmo de esta tarea minimizó el costo computacional, el algoritmo fue más eficiente, ya que sólo se recorren los pixeles donde la probabilidad de encontrar la línea correcta es más alta y en caso de existir una curva, la cantidad pequeña de pixeles toma solo una porción de la curva, aproximándola a una recta, de esta manera el resultado de la pendiente es más exacto que considerar la curva completa. La implementación de este algoritmo junto con la comunicación serial y el almacenamiento de la imagen pueden hacer que el prototipo Robótico logre hasta una velocidad de 0.724 m/s. Su tamaño en pila fue de 2048 bytes, el tiempo de cómputo igual a 6 ms, con periodo de ejecución de 200 ms y tercer (3er) prioridad del RTOS.

Tarea de detección de obstáculo: esta tarea le permiten al prototipo robótico decidir cómo actuar ante la existencia de un obstáculo o una señal de tránsito. A pesar de que el hardware pre-procesó imágenes de 1024x1280, es de aclarar que las imágenes utilizadas para calcular las profundidades de los objetos y las señales de tránsito, tenían un tamaño de 160x240 como se describió anteriormente en el titulo "Hardware de Procesamiento de imágenes", esto debido a que los algoritmos para el cálculo de la profundidad en estero imágenes implican un alto costo computacional (Pajares & Cruz, 2001). Mientras más pequeña la imagen menor

precisión existe en el cálculo de la profundidad pero mejora la velocidad de procesamiento del RTOS. La máxima distancia posible de medir es 140cm, ya que para distancias mayores a ésta las imágenes capturadas por cada lente del dispositivo TRDB-DC2 resultan ser casi las mismas, imposibilitando el cálculo de la profundidad debido a una ausencia de disparidad (Sabogal, 2007). El sistema de visión estéreo tiene la capacidad de ubicar objetos de distintos colores binarizados, con la característica que si existen varios objetos en la carretera con un mismo color, el sistema da la prioridad al obstáculo más cercano de ser ubicado. El algoritmo es capaz de localizar un objeto que tenga dimensiones mayores a $1.5cm^3$ a una distancia mínima de 10cm del prototipo robótico, la aplicabilidad de este resultado en la realidad depende de un sistema de procesamiento de alta velocidad que permita realizar el cálculo de la profundidad en imágenes con una resolución de al menos 1024x1280 pixeles. El tamaño en pila de esta tarea fue de 4096 bytes, con tiempo de cómputo entre 100 y 600 ms, periodo de ejecución entre 300 y 800 ms y cuarta (4ta) prioridad del RTOS.

Tarea de reconocimiento de señales de tránsito: Las RNAs utilizadas en el sistema se entrenaron con un total de 40 fotografías cada una y con otras 20 se realizó el test de acierto, cada red logró entrenarse con las siguientes características:

Número de épocas igual a 1150 para la RNA color rojo y 712 para la RNA color amarillo. Su parámetro *Goal* igual a 0.0001 y el porcentaje de acierto obtenido en entrenamiento por ambas RNAs fue de100%. Esta tarea tuvo un tamaño en pila de 4096 bytes, con tiempo de cómputo entre 120 y 1050 ms, periodo de ejecución entre 320 y 1250 ms y quinta (5ta) prioridad del RTOS. Las RNAs logran reconocer 6 diferentes clases de señales; señal reglamentaria, señal preventiva, semáforo en rojo, semáforo en amarillo, semáforo en rojo y amarillo, y la presencia de nada. Las RNAs logran clasificar de manera correcta las 5 señales de tránsito, con una distancia máxima de 150cm.

Por otra parte, la memoria utilizada en variables globales por el SoC fue de 240000 bytes. La velocidad promedio del prototipo Robótico incluyendo todos los sistemas hardware/ software del SoC fue de 0.23 m/s. El hardware de pre-procesamiento y el módulo de balance de blancos alcanzan una velocidad de 15 cuadros/ segundo. El algoritmo de navegación en el SoC puede actualizarse a una velocidad de 12.5 cuadros/segundo si se opera individualmente. En el mejor de los casos el SoC puede alcanzar una velocidad de procesamiento de 5 cuadros/ segundo. El módulo detector de media aritmética tarda 1280*512+3 ciclos de reloj, como el reloj de sincronización de los lentes del módulo TRDB-DC2 es de 25Mhz, entonces el tiempo efectuado para el cálculo de media aritmética es igual a 26.21ms.

En la práctica el sistema se comportó de manera satisfactoria, cumpliendo con los tiempos diseñados en el análisis de planificabilidad, cada uno de los tiempos de ejecución y las prioridades se definieron según la teoría de sistemas operativos en tiempo real y las características funcionales del sistema. Los tiempos de cómputo y ejecución de las tareas *detección de obstáculos* y *reconocimiento de señales de tránsito*, son algoritmos cuyos tiempos varían dependiendo de los objetos presentes en el entorno y la distancia a la que estos se encuentren (señales de tránsito u obstáculos), además estas tareas se consideran tareas esporádicas ya que sólo se ejecutan en su totalidad si existe un objeto dentro de la escena, por lo tanto los tiempos de ejecución de las tareas varían proporcionalmente al tiempo de cómputo.

4. Conclusiones

La implementación de un SoC, diseñado para guiar sobre una carretera un prototipo robótico móvil, clasificar señales de tránsito y detectar la distancia a la que se encuentran, hace un gran aporte a la investigación a nivel nacional, siendo uno de los primeros trabajos que logra implementar este tipo de tecnologías, dando una perspectiva a futuro de posibles implementaciones sobre automóviles reales.

Implementar un RTOS dentro de un SoC permite

hacer del software un sistema modular donde cada tarea es independiente, logrando controlar variables y recursos compartidos que eviten la corrupción en los datos con el uso de funciones de sincronización.

A pesar de la limitación en la velocidad de respuesta de los lentes del módulo TRDB-DC2, la poca resolución de las imágenes de copia para el RTOS y la baja frecuencia de procesamiento de la tarjeta DE2-70; Se logró implementar un SoC con características relevantes que logran guiar un dispositivo móvil en una carretera controlada, lo cual indica que una plataforma con mayores cualidades de respuesta permitirá optimizar las diversas tareas del RTOS y obtener mejores resultados no sólo en un entorno controlado, sino también en un ambiente real.

La tarea para la clasificación de señales de tránsito del RTOS es un diseño que permite reconocer no sólo las señales establecidas en el trabajo sino que también puede llegar a reconocer cualquier señal de tránsito preventiva o reglamentaria, pues cada señal de tránsito genera una "firma" única que la identifica.

La implementación de módulos hardware para el pre-procesamiento de imágenes, es una alternativa eficaz para el tratamiento de imágenes, ya que al usar bloques fragmentados que explotan el paralelismo y presentan una baja latencia, se logra el ahorro de cómputo que potencializa el desempeño de los sistemas de procesamiento.

5. Agradecimientos

Los más sinceros agradecimientos a Jesús el hermoso Dios, a COLCIENCIAS por el apoyo a la investigación a través del programa Jóvenes Investigadores e Innovadores "Virginia Gutiérrez de Pineda" año 2009, a la Universidad del Quindío por avalar jóvenes en la investigación, al Programa de Ingeniería Electrónica y al grupo de investigación GDSPROC de la Universidad del Quindío por acoger este tipo de proyectos que generan nuevos conocimientos e innovación.

6. Referencias Bibliográficas

Altera Corporation. (2009). *Nios II processors reference Handbook.* USA.

Bartley, G.P. ed. (2008). *Traffic Accidents: Causes and Outcomes.* New York: Nova Science Publishers, Inc.

Chu, P.P. (2006). *RTL HARDWARE DESIGN USING VHDL: Coding for efficiency, Portability, and Scalability.* New Jersey: John Wiley & Sons, Inc.

Gonzalez, R.C., & Woods, R.E. (2002). *Digital Image processing*, 2nd ed. New Jersey: Prentice Hall, Inc.

Graupe, D. (2007). *Principles of Artificial Neural Networks, Volume 6: Advanced Series on Circuits and Systems.* 2nd ed. 5 Toh Tuck Link, Singapore 596224: World Scientific Publishing Co. Pte. Ltd.

Labrosse, J.J. (2002). *MicroC/OS-II The Real-Time Kernel.* 2nd ed. Lawrence: CMP Books.

Pajares, G., & Cruz, J.M. (2001). *Visión por computador,* Madrid: RA-MA, S.A.

Passino, K.M., & Yurkovich, S. (1998). *Fuzzy Control,* Menlo Park: Addison Wesley Longman, Inc.

Pineda, D.F., Giraldo, J.J., & Estrada, E. (2009). *Piloto Automático Implementado en un Robot Móvil.* Tesis de Pregrado, Programa de Ingeniería Electrónica, Facultad de Ingeniería, Universidad del Quindío, Armenia, Colombia.

Quintero, J.J. (2004). *Herramienta para redes neuronales en tiempo real,* Tesis de Pregrado, Programa de Ingeniería Electrónica, Facultad de Ingeniería, Universidad del Quindío, Armenia, Colombia.

Sabogal, I.D. (2007). *Desarrollo de Software y Especificación de Arquitectura de Hardware para un Módulo de Visión Estereoscópica,* Tesis de Pregrado, Programa de Ingeniería Electrónica, Facultad de Ingeniería, Universidad del Quindío, Armenia, Colombia.

Team Berlin. (2007). *Spirit of berlin: An Autonomous Car for the DARPA Urban Challenge Hardware and Software Architecture.* Berlin: Freie Universität Berlin.

Terasic. (2006a). *Motion Detection Demo using DE2 with Camera Module.* http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=68&No=50&PartNo=3

Terasic. (2006b). *TRDB-DC2 user guide.* Taiwán.

Zini, G. (2004). *Estudio de innovaciones factibles en el diseño de la seguridad de impacto de un automóvil.* Tesis de Pregrado, Facultad de Ingeniería, Universidad de Buenos Aires. Buenos Aires, Argentina.