Scientia Et Technica
ISSN: 0122-1701
scientia@utp.edu.co
Universidad Tecnológica de Pereira
Colombia

Guerrero Aguirre, Álvaro; Ramos Giraldo, Paula Jimena
Sistema embebido de bajo costo para visión artificial
Scientia Et Technica, vol. 19, núm. 2, junio-, 2014, pp. 163-173
Universidad Tecnológica de Pereira
Pereira, Colombia

Sistema embebido de bajo costo para visión artificial

Low cost embedded system for machine vision

Álvaro Guerrero Aguirre[1], Paula Jimena Ramos Giraldo[2*]
[1]*Ingeniero Electrónico - Est. Ingeniería Eléctrica, Centro Nacional de Investigaciones en Café - Federación Nacional de Cafeteros de Colombia, Chinchina-Caldas, Colombia.*
[2]*M.Sc - EgnD.C. Ingeniero Electrónico - Est. Ingeniería Eléctrica, Centro Nacional de Investigaciones en Café - Federación Nacional de Cafeteros de Colombia, Chinchina-Caldas, Colombia.*
alvarochester89@gmail.com

***Resumen*— Se desarrolló un sistema embebido[1] para visión artificial utilizando diferentes sistemas de desarrollo presentes en la industria electrónica, ensamblados y puestos a punto para obtener un sistema hardware-software de bajo costo para realizar diferentes aplicaciones con visión artificial. Como resultado, el sistema consta de dos microprocesadores, LPC2106 de 60 MHz y el ATMEGA 2560 de 16 MHz, un menú de navegación y una pantalla de visualización. El sistema tiene la capacidad de analizar 7,14 imágenes por segundo, almacenar 2,6 imágenes por segundo y visualizar una imagen cada 3,4s.**

***Palabras clave*— Visión artificial, sistema embebido, protocolo de comunicación, interfaz usuario-máquina, control de periféricos.**

***Abstract*— An embedded system was developed for computer vision using different current development platforms available in the electronic industry, the platforms was assembled and tuned to obtain a low cost hardware-software system for making different applications using machine vision. As a result, the machine vision embedded system has two microprocessors, LPC2106 at 60 MHz and the ATMEGA 2560 at 16 MHz, a navigation menu and a visualization display. The system has the capability to analyze 7,14 frames per second, to store 2,6 frames per second and to visualize an image every 3,4 seconds.**

***Key Word* — Machine vision, embedded system, communication protocol, user-machine interface, peripheral control.**

I. INTRODUCCIÓN

Actualmente en diferentes secciones de la industria, la medicina, la agricultura y en las actividades cotidianas, se requiere el uso de sistemas que tomen decisiones a partir del procesamiento imágenes adquiridas. Los Sistemas de Visión Artificial (SVA) se han hecho cada vez más comunes en la domótica, seguridad, robótica, entre otros. Este crecimiento ha generado diferentes tecnologías, abaratado los costos y mejorado las capacidades de adquisición de sistemas electrónicos como cámaras y software para el procesamiento, una prueba de ellos son los dispositivos inteligentes.

Con el fin de obtener un sistema dedicado capaz de procesar imágenes y cumplir con funciones de manera programada, se desarrolló un sistema embebido para visión artificial a partir de dos sistemas de desarrollo, CMUCAM3 [2] y el ARDUINO [3], el primero dedicado al procesamiento de las imágenes y el segundo dedicado al control de periféricos, activación de indicadores y manipulación de otros sistemas y sensores.

El desarrollo de un SVA inicia con la selección de la cámara a utilizar considerando las especificaciones de diseño de una aplicación, las cuales dependen directamente del tipo de información a adquirir (longitud de onda), de la distancia entre el objetivo y la lente, el campo de visión de la lente y las condiciones de iluminación. La CMUCAM3 como SVA integra un sensor de imagen de resolución 352x288 píxeles, un sistema óptico y un procesador, permitiendo desarrollar sistemas portátiles independientes del computador con la capacidad de procesar la información a partir de las imágenes adquiridas y realizar control o monitoreo, como por ejemplo, la activación de un motor o una alarma.

A través de la CMUCAM3, es posible realizar control y monitoreo con restricciones de puertos, pues consta de solamente 4 pines de entrada/salida digital o salida PWM, comunicación serial RS232, comunicación serial TTL, es decir, que para el manejo de sistemas que requieran un mayor número de puertos, se requiere realizar una adición de un sistema microprocesado.

Generalmente para el manejo del SVA es necesario una interfaz usuario-máquina, conformado por una pantalla, donde a través de un menú desplegado y un teclado es posible seleccionar las acciones a realizar sobre las imágenes adquiridas y procesadas. Considerando las capacidades del SVA CMUCAM3, se implementó por medio del ARDUINO, un sistema de visualización para la interfaz usuario-máquina y para controlar diferentes opciones del sistema embebido

[1] Un **sistema embebido o dedicado** es un sistema de computación diseñado para realizar una o algunas funciones dedicadas frecuentemente en un sistema de computación en tiempo real.

Fecha de Recepción: 01 de Enero de 2014
Fecha de Aceptación: 24 de Junio de 2014

objeto de este artículo, como configurar, calibrar y controlar el SVA.

II. ANTECEDENTES

A. Generalidades y estado actual de los sistemas embebidos.

Un sistema embebido (SE) es un sistema de computación diseñado para realizar funciones dedicadas en tiempo real, cuyo fin es cubrir necesidades especificas; la mayoría de sus componentes se encuentran incluidos en una misma placa base, generalmente se pueden programar directamente en lenguaje ensamblador o mediante un compilador utilizando lenguaje C o C++, su principal objetivo es reducir costos.

El crecimiento en la industria electrónica se ha generado en parte por el aumento en los niveles de integración, es decir, la cantidad de transistores que se tienen por mm^2 y el costo inversamente relacionado con dicha cantidad. En los años 70's la escala de integración LSI (*Small Scale Integration*) tenía desde 1.000 a 10.000 transistores por circuito integrado con un promedio de 200 transistores por mm^2, en los 80's la escala VLSI (*Very Large Scale Integration*) agrupaba hasta 100.000 transistores por circuito integrado; fue en esta época cuando IBM desarrolló el primer sistema embebido (SE). Los años 90's registraron hasta 270 mil transistores por mm^2 y a comienzos de mileno se contaban con cerca de 4,5 millones de transistores por mm^2. Actualmente se alcanzan niveles de integración de hasta 9 millones de transistores por mm^2 dando origen a microprocesadores más veloces, eficientes y a menor precio [5][11].

Los altos niveles de integración han permitido construir nuevos sistemas más veloces, con mayor cantidad de puertos, de fácil interacción con otros dispositivos; como los microcontroladores, dispositivos que integran en un solo chip una unidad de procesamiento, una memoria y periféricos de entrada y salida; los microcontroladores se dividen en varias familias, entre ellas los PIC de *Microchip* y los ARM de *ARM Holdings.* Además de los microprocesadores y microcontroladores, en la actualidad están en auge los sistemas de desarrollo o sistemas embebidos, que integran en un microcontrolador, una cantidad de puertos análogos y digitales, y protocolos de comunicación para desarrollar casi cualquier tipo de aplicación que la ingeniería requiera. Los más conocidos hoy en día son: Mbed [9] plataforma de desarrollo con librerías listas para su uso utilizando un compilador que está totalmente bajo un entorno Web, BeagleBoard [10] Sistema de desarrollo basado en un microcontrolador ARM Cortex-A8 con entradas y salidas de audio, video y conexión USB, Raspberry Pi [12] pequeña computadora desarrollada en el reino unido por la fundación Raspberry PI con sistema operativo basado en linux, ARDUINO [3] cosiste en una placa para desarrollo basada en leguaje C, actualmente existen muchas referencias de dicha placa cada una con características y capacidades diferentes.

B. Aplicaciones en visión artificial.

La visión artificial y el procesamiento digital de imágenes normalmente son temas desarrolladas en computadores portátiles o de escritorio, requieren en muchas ocasiones, tarjetas de video, software y hardware especializado. Sin embargo en la actualidad las aplicaciones de visión artificial han migrado a los SE, incorporando sensores de imagen, sistemas ópticos, sistemas de iluminación y procesando la información fuera de un computador para tomar decisiones en ambientes donde se requiere portabilidad, tal como lo muetsra Dongxian He *et al*., [7] en 2007, implementando una estación meteorológica con sensores de temperatura, humedad del suelo, humedad relativa e intensidad de luz y cuatro módulos de cámara CCD, para tener información del entorno de la estación, las imágenes fueron transportadas por la red en archivos http, la información fue procesada en un centro de datos donde un servidor VPN corre las aplicaciones desarrolladas en JAVA para procesar la información obtenida por los sensores.
Otra forma de utilizar los sistemas embebidos para visión artificial, es mediante la adquisición de video, en este caso Jie ZHANG *et al*., [6] utilizan un microprocesador ARM7 S3C44B0X y proponen una técnica para adquisición de video en tiempo real para monitoreo remoto, vigilancia del medio ambiente y muchas otras posibles aplicaciones, el sistema esta conformado por GPS, plataforma de información y canal de transmisión, el sistema también puede adquirir y procesar imágenes.

C. Aplicaciones en servo-visión.

La servo-visión o control visual es un concepto que se viene trabajando en los SE, este consiste en utilizar la visión artificial como retroalimentación para realizar un control sobre un actuador, Zhenyu Ye *et al*. [8] implementaron un sistema de servo-visión en una FPGA aplicado a una impresora de inyección de tinta basada en visión, los autores de este trabajo indican que entre mayor sea la frecuencia de muestreo la visión será una fuente de retroalimentación mucha más rentable.

III. METODOLOGÍA

Se presenta el proceso de desarrollo de un sistema embebido de bajo costo para visión artificial, haciendo un despliegue de las etapas necesarias para conformarlo, partiendo desde la selección de la cámara hasta la implementación de una interfaz usuario-máquina.

Este trabajo esta basado en una manera novedosa y sencilla para conformar un sistema de visión, aprovechando la tecnología que se tiene en la actualidad, utilizando sistemas de desarrollo disponibles, con el fin de reducir el tiempo del desarrollo y optimizar los recursos de hardware e implementación.

Los materiales utilizados fueron:

- Sistema embebido CMUCAM3.
- Sistema embebido ARDUINO.
- Sistema de iluminación basado en LEDs de alto brillo y sistemas difusores de luz. iluminación tipo *full bright field.*
- Display LCD SHIELD del celular Nokia 6100.
- Servomotores.
- Memorias SD.
- Pulsadores.

A continuación se describen las etapas ejecutadas para el desarrollo del sistema embebido VA:

A. Selección de cámara

Para el desarrollo de un SVA se debe tener en cuenta la longitud de onda requerida (visible o fuera del visible), velocidad de muestreo, resolución, espacio de color, características ópticas velocidad de transmisión de datos y protocolo de comunicación.

El SVA CMUCAM3 mostrado en la Figura 1, fue desarrollado en la Universidad de Carnegie Mellon [2], es capaz de capturar hasta 26 cuadros por segundo, con resolución configurable entre 352x288 y 176x144 píxeles, adquisición en los espacios de color RGB, HSV, YCrCb o Monocromatico, conformado por un microprocesador ARM7TDMI de 32 bits y 60MHz [1], un sensor de imagen OV6620 [4], entradas y salidas TTL, comunicación serial.

El sistema tiene una cámara de referencia C3088 con una lente f4,9mm y F2,8 (distancia de la lente al objeto y distancia focal) dicho lente puede ser reemplazado dependiendo de los requerimientos ópticos del sistema, pues se encuentra montado en un soporte de lente estándar M12x5.

Figura 1. Sistema embebido CMUCAM3 con lente, sensor de imagen y procesador

La CMUCAM3 cuenta con una librería llamada CC3 la cual logra por medio de lenguaje C, programar procesamientos de imagen sobre el hardware del SVA. Esta librería identifica y ubica variables (software) en sus respectivos registros (hardware), define el tipo de variables, estructuras, definiciones, y requerimientos de memoria para cada una de las funciones programadas.

Se realizó una exploración de la librería CC3 con el fin de conocer las variables, tipos de variables, estructuras, funciones utilizadas en el control de todos los periféricos de la CMUCAM3 y el funcionamiento de todas las rutinas para el procesamiento de imágenes, desde la adquisición, almacenamiento en memoria interna (FIFO) y externa (SD), obtención del histograma, seguimiento del color, diferenciación y ventanas virtuales.

Para utilizar la librería, primero se escribe el código en lenguaje C utilizando un editor como DEV C, este código se guarda con extensión .c en una carpeta dentro de la ruta de la librería acompañado de un archivo *Makefile* que debe contener el nombre del archivo a compilar, después se procede a compilar el código por medio de un emulador de Linux en Windows llamado Cygwin, utilizando el compilador GCC, el cual se encarga de tomar el código fuente y llevarlo al lenguaje maquina (.hex) del procesador de la CMUCAM3. Posteriormente se conecta la CMUCAM3 por el puerto serial y se ejecuta la aplicación llamada LPC2000 Flash Utility V2.0.1 de PHILIPS para descargar el código compilado en la memoria Flash de la CMUCAM3.

B. Procesamiento de las imágenes adquiridas.

Con el fin de explorar las capacidades de procesamiento y determinar los tiempos de proceso para diferentes algoritmos, se desarrollaron diferentes programas, entre ellos medición de la intensidad del canal rojo de una imagen a partir de su histograma, manipulación de las características de la cámara como tiempo de exposición, balance de blancos y espacio de color de las imágenes adquiridas, adicionalmente se realizó un algoritmo de prueba para controlar el puerto SD para almacenar imágenes y archivos de texto en una memoria y se realizaron programas de verificación del control de periféricos como LEDs indicadores y servo motores.

C. Interfaz Usuario-Maquina.

Para que el usuario tenga control sobre el sistema de visión se requiere una interfaz usuario-maquina que sirva para controlar y configurar el sistema, dicha interfaz está basada en un menú de navegación que cuenta con pulsadores y una pantalla para observar lo que ocurre en frente del sensor de imagen y generar de esta forma independencia con el computador.

La interfaz fue implementada en una tarjeta ARDUINO MEGA 2560 [3], como sistema embebido maestro el cual controla la CMUCAM3, una pantalla de visualización, un menú de navegación y otros periféricos. El ARDUINO y el display LCD a color son mostrados en la Figura 2. La tarjeta ARDUINO cuenta con un microprocesador Atmega 2560 de 8 bits y 16MHz, tiene comunicación serial, maneja entradas y salidas digitales, salidas PWM y se programa mediante una plataforma basada en lenguaje C (*Processing*). El Display (Color LCD Shield) de ARDUINO es un dispositivo que

viene montado en una tarjeta para conectarlo directamente a la placa de ARDUINO, controlado y configurado por la librería *ColorLCDShield.h*, tiene una resolución de 132x132 pixeles y puede representar imágenes en el espacio de color RGB utilizando 8 bits por cada pixel o 12 bits por cada pixel dependiendo de la configuración que el usuario elija.

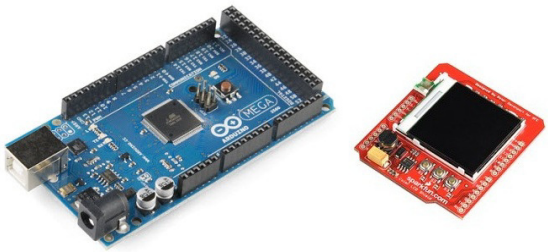

Figura 2. Tarjeta ARDUINO y Display LCD.

D. Protocolo de comunicación.

Para comunicar ambos sistemas (CMUCAM3 y ARDUINO) se desarrolló un protocolo de comunicación que permite controlar desde la interfaz implementada en ARDUINO, las funciones y periféricos de la CMUCAM3, dicha comunicación es de tipo serial RS232 a 115.200 bits por segundo.

IV. RESULTADOS

En esta sección se presentan los resultados de los algoritmos implementados en ambos sistemas embebidos, partiendo de las pruebas exploratorias y pasando por el procesamiento de imágenes, control de periféricos hasta la interfaz de usuario-máquina.

A. Levantamiento de la librería CC3.

Se identificaron variables, tipos de variables, funciones y estructuras, a continuación se mencionan las funciones más importantes:

- *cc3_uart_init* función para configurar el puerto serial, numero de puerto, velocidad, cantidad de bits y modo.
- *cc3_camera_init* inicializa el hardware de la cámara y reinicia los parámetros por defecto de la imagen;
- *cc3_filesystem_init* inicializa los dispositivos del sistema como el puerto SD;
- *cc3_timer_wait_ms* genera un retardo en la ejecución del código;
- *cc3_timer_get_current_ms* devuelve el tiempo en milisegundos que lleva el dispositivo desde que se enciende;
- *cc3_pixbuf_frame_set_roi* función para establecer una región de interés donde sus variables de entrada son los puntos extremos de la región;
- *cc3_camera_set_raw_register* permite configurar los registros directamente sobre el sensor OV6620;
- *cc3_camera_set_resolution, cc3_camera_set_colorspace, cc3_camera_set_brightness, cc3_camera_set_contrast* configuran las características de la imagen;
- *cc3_led_set_state, cc3_gpio_set_mode, cc3_gpio_set_value, cc3_gpio_set_servo_position* permiten controlar los periféricos de la CMUCAM3.

En la librería hay más funciones para el procesamiento de imágenes que no son nombradas en este articulo.

B. Activación de un conjunto de LEDs a partir de la detección de intensidades de rojo (histograma) de una imagen en tiempo real.

Se implementó un algoritmo que se encarga de expresar el nivel de rojo de una imagen adquirida por el SVA, utilizando los LEDs de la CMUCAM3.

El diagrama mostrado en la Figura 3 describe el funcionamiento del algoritmo, el cual consiste en adquirir una imagen y hallar su histograma[2] utilizando la función *cmucam2_get_histogram* de librería CC3; la función tiene como parámetros: el canal y la cantidad de bins[3], que por defecto es 28; cuando se halla el histograma del canal rojo, el programa devuelve 28 valores (bins), estos valores representan la cantidad de píxeles en cada rango de intensidades. La representación del color en la cámara es de 8 bits se tienen 256 intensidades diferentes sin embargo la cámara solo toma el rango (16 – 240), es decir, 224 intensidades, divididas en 28 bins, cada bin representa la cantidad de píxeles que hay en 8 intensidades diferentes. Se determina la posición del bin que tiene el número mayor y esta posición se ubica en un nivel de acuerdo a la clasificación mostrada en la segunda columna de la Tabla 1, para visualizarlo, se da la orden de encendido o apagado de los LEDs de la CMUCAM3 de acuerdo al nivel encontrado, observando la Tabla 1 en las columnas 3, 4 y 5 se tiene el estado de cada LED según el nivel de rojo hallado, teniendo en cuenta que el nivel 0 (cero) es la mínima cantidad de rojo en una imagen, y el nivel 7 es la máxima cantidad de rojo en una imagen.

[2] **Histograma**. Representación gráfica de una variable en forma de barras, donde cada barra representa la frecuencia de los valores representados.
3 **Bins**. Cantidad de divisiones del histograma que en este caso es mínimo 1 y máximo 2255.

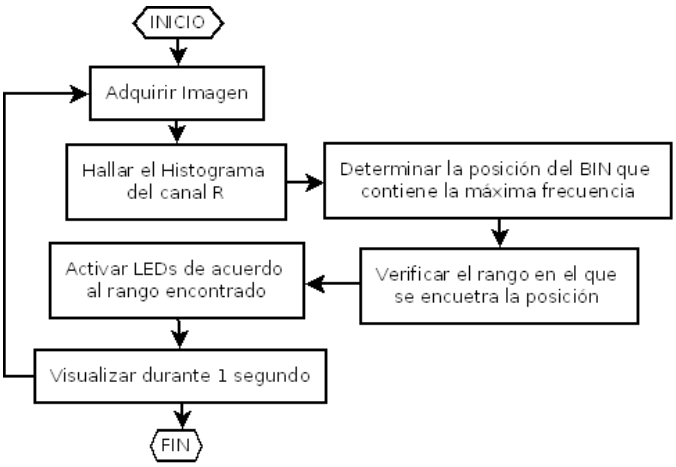

Figura 3. Algoritmo para detectar rojo a partir del histograma.

Niveles	Rango	LED 1	LED 2	LED 3
0	(6 - 7)	OFF	OFF	OFF
1	(8 - 9)	OFF	OFF	ON
2	(10 - 11)	OFF	ON	OFF
3	(12 - 13)	OFF	ON	ON
4	(14 - 15)	ON	OFF	OFF
5	(16 - 17)	ON	OFF	ON
6	(18 - 19)	ON	ON	OFF
7	(20 - 28)	ON	ON	ON

Tabla 1. Estado de los LEDs según el nivel de rojo en una imagen.

En la Figura 4 y la
Figura 5 se presentan dos histogramas, uno para una imagen que no tiene componente de rojo en la cual se puede observar que las intensidades más altas están a la izquierda del eje X (valores bajos de intensidad de rojo), y otro para una imagen que es completamente roja en donde las intensidades más altas se encuentran a la derecha del eje X (valores más altos de intensidad de rojo), las gráficas fueron obtenidas en Microsoft Excel® utilizando los valores entregados por el algoritmo, en donde el eje X corresponde a los 28 bins que entrega la función del histograma y el eje Y a su correspondiente intensidad.

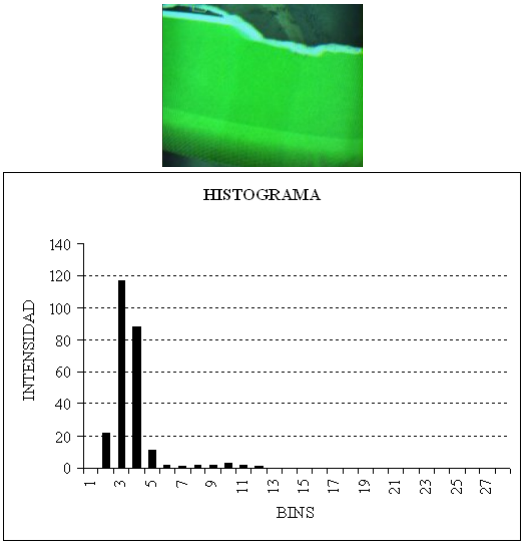

Figura 4. Histograma imagen que no tiene componente de rojo.

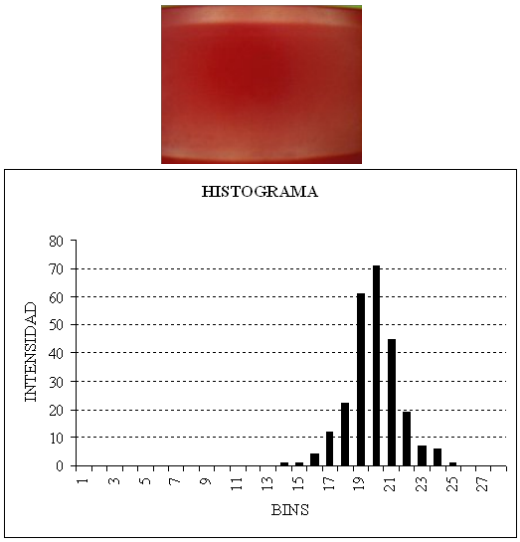

Figura 5. Histograma imagen que tiene componente de rojo.

B. Modificación de la configuración del sensor de imagen.

Como se ha mencionado antes, la CMUCAM3 a través de la librería CC3 tiene la posibilidad de configurar las características de la imagen, cambiando los valores de configuración directamente en los registros del sensor, en la Tabla 2 se relacionan el número de registro para el brillo, contraste y resolución, y el valor ingresado para lograr la configuración deseada.

En la librería existen funciones que permiten realizar estas y otras configuraciones sin necesidad de ingresar directamente a los registros del sensor. Se implementó en el sistema VA un algoritmo donde el usuario tiene la posibilidad de escoger a conveniencia las características de la imagen a adquirir ingresando los correspondientes valores a través de la interfaz usuario-maquina.

Característica	Registro	Valor
Brillo	0x06	(0 - 255)
contraste	0x05	(0 - 255)
Resolución	0x14	(0x20) baja (0x00) alta

Tabla 2. Registros y valores de configuración del sensor OV6620.

C. Almacenamiento de imágenes en una memoria SD.

En el desarrollo de un SVA es necesario almacenar las imágenes que adquiere el sistema, con el fin de caracterizar algún evento especifico; para ello la CMUCAM3 cuenta con un puerto de conexión para una memoria SD y las funciones en la librería que controlan dicho puerto para guardar imágenes en diferentes formatos como el JPEG. Se probó el algoritmo que se encarga de almacenar dichas imágenes una vez es pulsado el botón externo de la CMUCAM3, y se realizaron algunas modificaciones para tener la capacidad de almacenar secuencias de imágenes con solo una pulsación del botón y así lograr la posibilidad de caracterizar eventos, el algoritmo puede almacenar imágenes, en alta y en baja resolución, además se puede modificar para almacenar imágenes procesadas.

D. Sistema de Iluminación.

El sistema de iluminación implementado, está conformado por un arreglo de LEDs de alto brillo de 80mW cada uno alimentados a 12V, una lamina difusora para homogenizar la luminosidad en la escena y unos soportes en acrílico de 3mm para ensamblar el sistema de iluminación a la CMUCAM3 en la Figura 6 se observa el montaje de dicho sistema de iluminación.

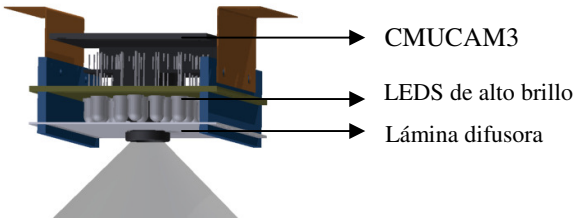

Figura 6. Sistema de iluminación full bright field.

E. Interfaz Usuario-Maquina.

La etapa inicial para desarrollar la interfaz de usuario fue tener comunicación entre los dos sistemas principales (CMUCAM3 y ARDUINO). Estos sistemas tienen comunicación serial, sin embargo, la CMUCAM3 maneja un protocolo RS232[4] mientras que el ARDUINO maneja un protocolo TTL de 0 a 5V, entonces fue necesario implementar un circuito (ver Figura 7) para acondicionar el protocolo TTL del ARDUINO al protocolo RS232 de la CMUCAM3 a una velocidad de 115.200 bits por segundo mediante un MAX232; esta comunicación es bidireccional, es decir, que se puede enviar o recibir datos al mismo tiempo, ventaja que será de gran utilidad para el control del SVA, ya que permite independizar los sistemas embebidos del computador y tener un sistema portátil que alimentado por baterías puede ser llevado a cualquier lugar.

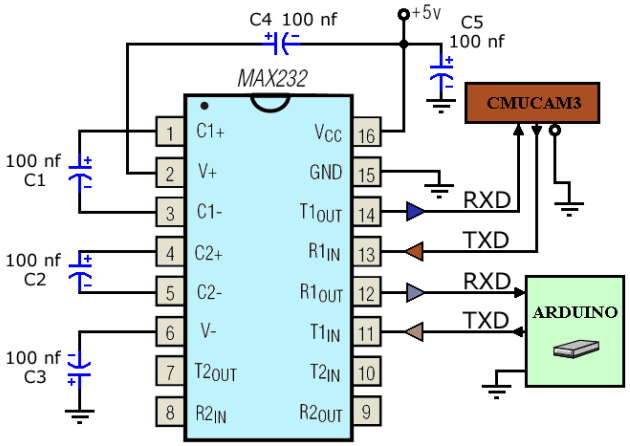

Figura 7. Comunicación CMUCAM3 y ARDUINO.

La interfaz usuario-máquina está basada en un menú de navegación en el cual se inicia el sistema, se realiza un saludo y se muestra un mensaje de "ayuda" con las indicaciones básicas para navegar en el menú, la interfaz cuenta con varias funciones propias del sistema de visión artificial en el cual el usuario tiene la posibilidad de configurar la CMUCAM3, realizar procesamiento de imágenes y control de periféricos, dicho menú fue implementado en la tarjeta ARDUINO MEGA, utilizando tres pulsadores para navegar (ARRIBA, ENTER y ABAJO), y la pantalla LCD a color como sistema de visualización, en la Figura 8 se encuentra el diagrama de flujo del menú implementado en ARDUINO.

El display COLOR LCD SHIELD de ARDUINO tiene una librería llamada "*ColorLCDShield*" donde hay funciones para definir el contraste de la pantalla, pintar un píxel de un color determinado, dibujar círculos, rectángulos y líneas, escribir caracteres o cadena de caracteres en pantalla en una posición determinada.

A continuación se realiza la descripción detallada de las funciones que conforman el menú del sistema de visión artificial, para cada una fue necesario diseñar un protocolo de comunicación entre la CMUCAM3 y el ARDUINO haciendo uso de las librerías CC3 de la CMUCAM3 y ColorLCDShield del ARDUINO.

[4] Protocolo especial para comunicación serial que opera entre -12V y 12V.

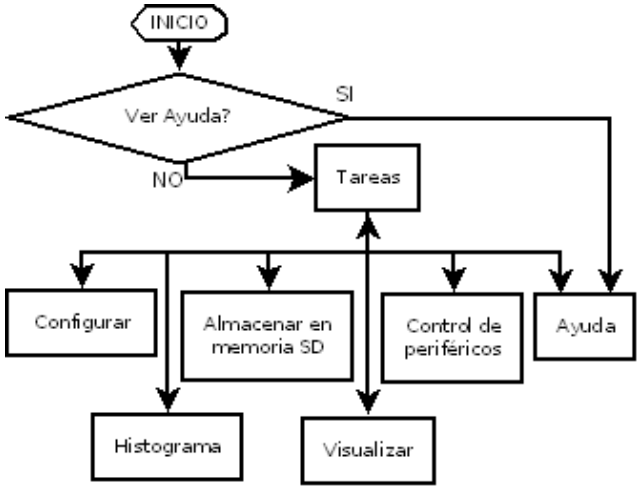

Figura 8. Menú de navegación del sistema.

1) Función Configurar.
Para realizar la configuración de la cámara se diseñó el siguiente protocolo de comunicación:

Cuando el usuario selecciona la función *Configurar*, el ARDUINO envía un carácter "C" a la CMUCAM3, esta lo interpreta, ingresa a la sección de configuración y responde al ARDUINO con un "ACK", cuando el ARDUINO recibe la respuesta pide al usuario la característica a modificar y el valor a establecer, el ARDUINO envía un carácter a la CMUCAM3 dependiendo de los datos ingresados así, para el brillo se envía una "B", Contraste una "T", Resolución una "R" y para Espacio de color una "E", una vez recibido el carácter por la CMUCAM3, esta responde con un "ACK" e ingresa al registro seleccionado, cuando el ARDUINO recibe respuesta de la CMUCAM3, envía el valor o parámetro ingresado por el usuario, la CMUCAM3 lo recibe y lo almacena en el registro de configuración definido previamente, en el diagrama de la Figura 9 se muestra el protocolo implementado.

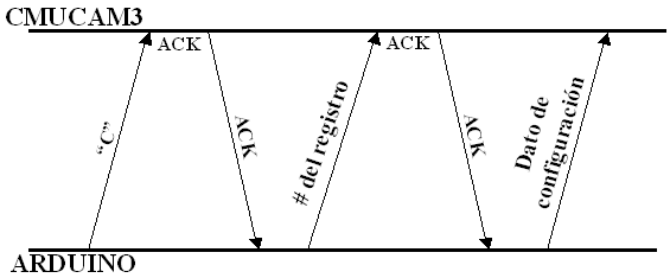

Figura 9. Protocolo de comunicación para configurar la CMUCAM3.

Se aclara que solo se puede configurar un registro a la vez, cuado se ingresa el valor de configuración, ambos sistemas se devuelven al estado de espera donde el usuario selecciona que función desea ejecutar.

2) Función Histograma.

Cuando el usuario selecciona la función histograma, el ARDUINO envía el carácter "H" a la CMUCAM3, esta lo interpreta, envía un "ACK" y entra a ejecutar el algoritmo que determina la intensidad de rojo y activa los LEDS de la CMUCAM3 (ver Figura 3), el ARDUINO recibe la respuesta de la CMUCAM3 y queda a la espera de que el usuario determine el momento de finalización de ejecución del algoritmo, cuando el usuario presiona el botón ENTER para finalizar, el ARDUINO genera una interrupción para que la CMUCAM3 termine y regrese al estado de espera, en la Figura 10 se observa gráficamente el protocolo anterior.

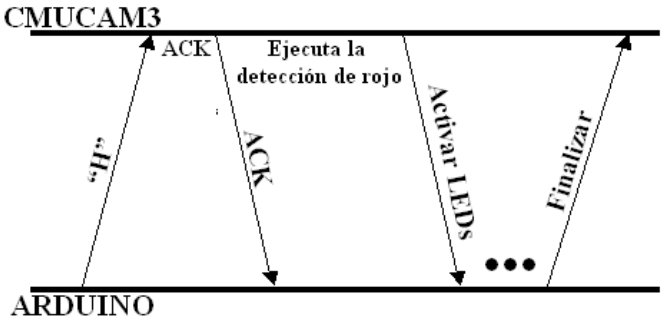

Figura 10. Protocolo de comunicación para hallar el histograma.

3) Función almacenar en memoria SD.

Cuando se selecciona esta función, el ARDUINO envía el carácter "A" a la CMUCAM3, esta lo recibe, responde con un "ACK" para informar que recibió la información y ejecutar el algoritmo para almacenar imágenes en la memoria SD. El ARDUINO queda a la espera de que la CMUCAM3 envíe un "ACK" si la imagen se adquirió y almacenó correctamente en la memoria SD, o un "NCK", si ocurrió algún error durante el proceso de almacenamiento o no se tenia conectada la memoria en el puerto, cuando esto ocurre el ARDUINO muestra un mensaje en pantalla diciendo al usuario: "*Revise la memoria y vuelva a intentarlo*", en la Figura 11 se observa el protocolo para almacenar una imagen.

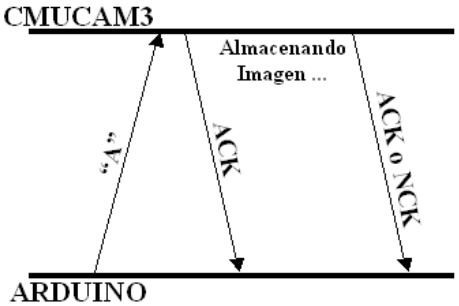

Figura 11. Protocolo de comunicación para almacenar una imagen.

4) Función Visualizar.
Las características de las imágenes que se pueden visualizar en el DISPLAY de ARDUINO son diferentes a las imágenes

adquiridas por la CMUCAM3, en el display la resolución es de 132x132 píxeles y la representación del color se realiza de dos formas, una utilizando 8 bits por píxel así, tres para el canal rojo, tres para el verde y dos para el azul como se observa en la Figura 13, y la otra es utilizando 12 bits por píxel, cuatro por cada canal como se observa en la Figura 14. Para que la información recibida por el ARUINO sea la correcta, es necesario realizar un acondicionamiento de la imagen antes de enviarla al display, dicho acondicionamiento consiste en configurar la CMUCAM3 para que adquiera la imagen en baja resolución, tome una región de interés de la imagen de tamaño 132x132, ver Figura 12.

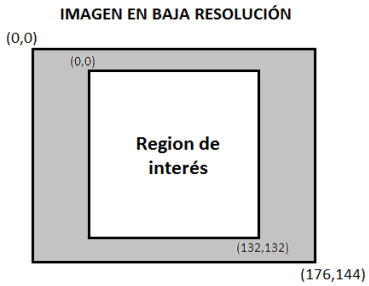

Figura 12. Región de interés para visualizar en el LCD a color.

Luego se realiza el cambio de la representación del color en 8 bits o 12 bits según sea el caso, a continuación se describirá cada uno de ellos:

Representación de la imagen en 8 bits: Teniendo en cuenta que la imagen en la CMUCAM3 es de 24 bits por píxel, se crea la necesidad de modificar la información generada por la CMUCAM3 para que el ARDUINO la interprete. La modificación consiste en realizar una normalización del valor de cada canal cambiando el valor entre 0 y 255 por un valor entre 0 y 1, luego expandir los valores del canal rojo y verde a una representación de tres bits, multiplicando el valor normalizado por 8, el valor del canal azul se expande a una representación de 2 bits multiplicando el valor por 4.

En el valor del canal rojo se realiza un corrimiento de 5 posiciones a la izquierda, multiplicando 5 veces por 2, en el valor del canal verde se realiza un corrimiento de 2 posiciones a la izquierda, multiplicando 2 veces por 2, el valor del canal azul no se modifica, por último se concatenan los tres valores por medio de la operación lógica "OR", quedando la representación de los tres componentes en un solo byte listo para ser enviado al ARDUINO, en el diagrama de la Figura 13 se puede observar el proceso en detalle.

Representación de la imagen en 12 bits: El cambio de representación de 24 bits por píxel a 12 bits por píxel consiste en normalizar el valor de cada canal de (0-255) a (0-1), luego expandir cada valor normalizado a un valor de cuatro bits, multiplicando por 16 posteriormente se realiza un corrimiento a la izquierda de cuatro posiciones en el canal rojo multiplicando 4 veces por 2, el valor obtenido se concatena con el valor del canal verde por medio de una operación OR, quedado así un byte conformado por el canal rojo y verde; en el canal azul se realiza el mismo corrimiento a la izquierda de cuatro posiciones y se envían los dos bytes que conforman el valor del píxel en 12 bits, en el diagrama de la Figura 14 se observa el proceso para cambiar la representación de la imagen a 12 bits.

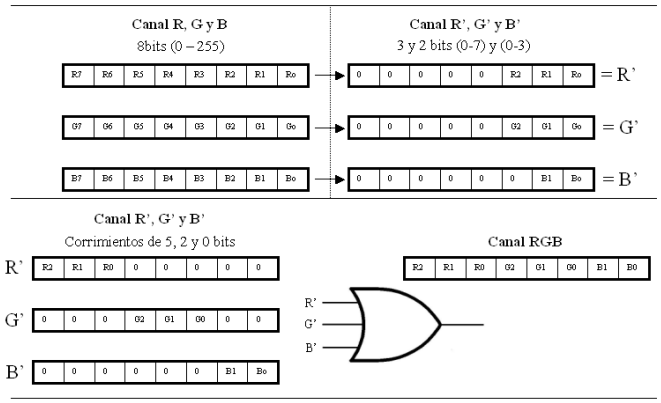

Figura 13. Proceso para cambiar la representación a 8 bits.

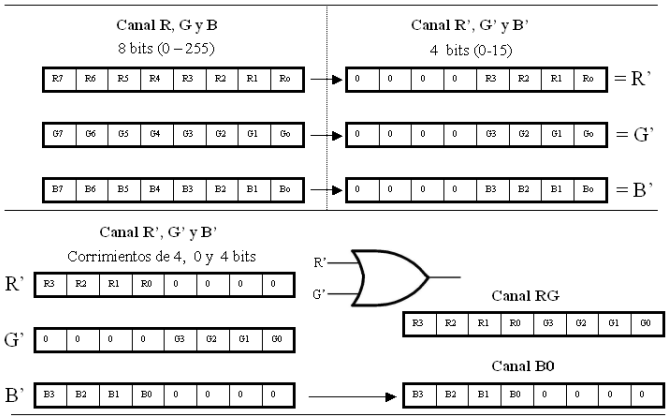

Figura 14. Proceso para cambiar la representación a 12 bits.

Para finalmente visualizar las imágenes en la pantalla LCD de ARDUINO se diseñó e implementó un protocolo de visualización, el cual varia dependiendo de la representación que se utilice, dicho protocolo fue implementado con el fin de no tener pérdidas de información durante la transmisión de la imagen al ARDUINO, ya que el búfer de entrada de datos es de 128 bytes, y no tiene espacio suficiente para recibir todos los datos.

Protocolo de comunicación para transmitir imagen en 8 bits: Cuando el usuario selecciona la función visualizar imagen en 8 bits el ARDUINO envía el carácter "8" a la CMUCAM3 y esta debe responder con un "ACK" para que el ARDUINO se prepare para recibir datos, por su parte la CMUCAM3 debe adquirir una imagen, extraer la ventana de interés de 132x132 y proceder a ejecutar la rutina mostrada en la Figura 13, por cada píxel de la ventana, la imagen es enviada al ARDUINO píxel por píxel en tramas de 128 BYTES, cuando se envían 128 datos, la CMUCAM3 espera un "ACK" del ARDUINO,

esta pausa se realiza con el fin de no perder información por submuestreo. El ARDUINO recibe los datos y utiliza la función "*setpixel*" de la librería *ColorLCDShield* para escribir dicho valor en pantalla, teniendo en cuenta la coordenada (X, Y) del píxel recibido y sabiendo que se inicia en (0,0) y se envía fila por fila de forma secuencial hasta llegar a la (132,132). En la Figura 15 se puede observar el protocolo para transmitir una imagen en 8 bits desde la CMUCAM3 al ARDUINO, y en la Figura 16 se puede observar una imagen con representación de 8 bits por píxel en la pantalla LCD.

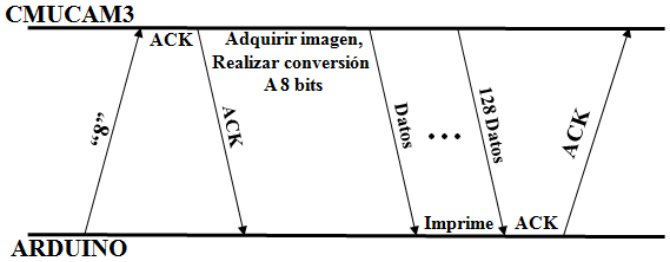

Figura 15. Protocolo de transmisión imagen de 8 bis.

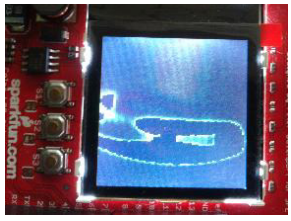

Figura 16. Imagen de 8 bits visualizada en la pantalla LCD.

Protocolo de comunicación para transmitir imagen en 12 bits: Cuando se selecciona la función visualizar imagen en 12 bits, el ARDUINO envía el carácter "9" a la CMUCAM3, esta debe responder con un "ACK" para que el ARDUINO entre en una rutina de espera de datos, por su parte la CMUCAM3 debe adquirir una imagen, extraer una de ventana de 132x132 píxeles y ejecutar la rutina descrita en la Figura 14, en este caso se tienen 2 BYTES por píxel, se envían al ARDUINO primero el que contiene los canales R y G, luego se envía el canal B, el ARDUINO recibe ambos valores, los concatena en una sola variable, multiplicando el primero por 256 y sumando el segundo valor, luego de concatenado, utiliza la función "*setpixel*" de la librería *ColorLCDShield* para escribir el valor de 12 bits en la pantalla LCD, teniendo en cuenta la coordenada (X,Y) del píxel sabiendo que se inicia en la (0,0) y se envía fila por fila de forma secuencial hasta llegar a la (132,132), en la Figura 17 se observa el protocolo de transmisión de imagen en 12 bits. En la Figura 18 se puede observar una imagen de 12 bits visualizada en la pantalla LCD de ARDUINO, esta imagen tarde un tiempo de 6.642 ms para ser enviada en su totalidad al ARDUINO, tarda aproximadamente el doble de tiempo que la de 8 bits porque esta utiliza dos BYTES por píxel para representar el color mientras que en la de 8 bits solo se necesita un BYTE.

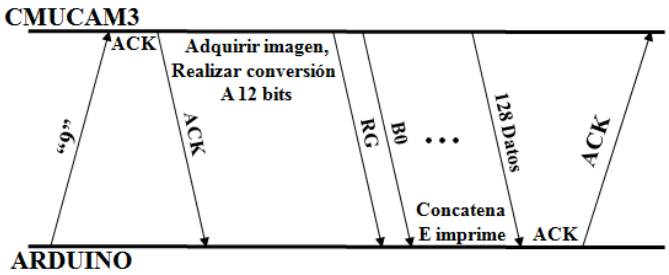

Figura 17. Protocolo de transmisión imagen 12 bits.

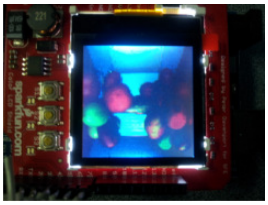

Figura 18. Imagen de 12 bits visualizada en la pantalla LCD.

5) Función control de periféricos.

A partir del análisis de las imágenes adquiridas por la CMUCAM3 y enviadas al ARDUINO se pueden determinar acciones de control sobre diferentes periféricos como motores, servomotores o LEDs; para este desarrollo, se realiza la activación de dos matrices de LEDs (rojos y verdes), utilizando las salidas digitales de la placa ARDUINO por medio de un algoritmo que recibe una imagen, la divide en nueve zonas diferentes y enciende el LED que representa el color que predomina en la zona, en caso de que no se encienda ninguno significa que en la zona no predomina ninguno de estos dos colores, en la Figura 19 se puede observar la imagen dividida en zonas y la matriz de LEDS que la representa.

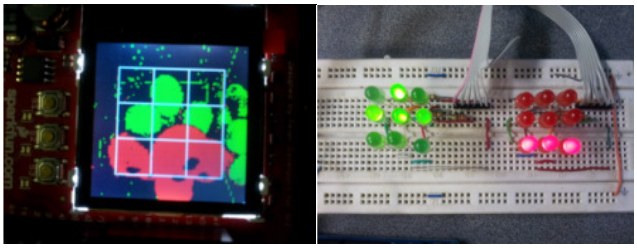

Figura 19. Imagen dividida en zonas y matrices de LEDS.

6) Función Ayuda.

Se muestra en la pantalla LCD un pequeño resumen de cómo utilizar el sistema embebido VA.

G. Resumen de tiempos requeridos por el sistema para ejecutar cada función

Se realizó un registro del tiempo requerido por cada algoritmo para ejecutarse, en la Tabla 3 se puede observar cada algoritmo y el tiempo requerido para cuando es una imagen de alta o baja resolución.

Algoritmo	Alta Resolución	Baja Resolución
Configurar	1 ms	
Histograma	--	140 ms
Almacenar	1.183,5 ms	385,8 ms
Visualizar 8 bits	3.354 ms	
Visualizar 12 bits	6.649 ms	
Control Periféricos	3.648 ms	

Tabla 3. Tiempo empleado por el sistema VA para ejecutar cada algoritmo.

H. Especificaciones finales del sistema embebido.

El sistema embebido de bajo costo para visión artificial desarrollado, está basado en dos microcontroladores, el LPC2106 de 60 MHz y el ATMEGA 2560 de 16 MHz; tiene 3 pulsadores para navegar en el menú del sistema, un puerto para conexión de memoria SD, 4 LEDs de visualización en la CMUCAM3 programables desde la CMUCAM3, dos matrices de LEDs y una pantalla LCD para visualización. Además cuenta con 25 entradas o salidas digitales de propósito general, dos puertos de comunicación serial TTL, un puerto de comunicación I2C y 16 entradas o salidas análogas, 4 pines que pueden ser entrada/salida digital o salida PWM para controlar servomotores en la Figura 20 se puede ver un diagrama esquemático del sistema desarrollado.

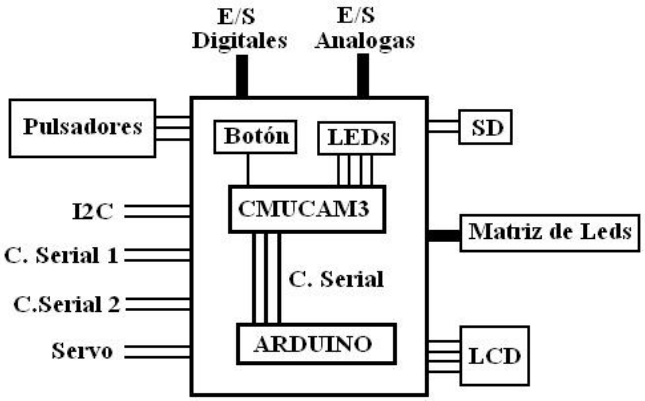

Figura 20. Diagrama esquemático del sistema VA.

El sistema tiene la capacidad de analizar la intensidad de rojo de una imagen a partir del histograma a 7,14 imágenes por segundo. Puede almacenar en una memoria SD a una velocidad de 2,6 imágenes por segundo en baja resolución y a 0,84 imágenes por segundo en alta resolución. Se visualizan imágenes de la CMUCAM3 cada 3,4s imágenes de 8 bits y cada 6,7s imágenes de 12 bits.

F. CONCLUSIONES

El trabajo mostrado en este artículo sirve como guía a la comunidad científica y académica para el desarrollo de nuevas aplicaciones automáticas, a un menor costo y en un menor tiempo, comparado con el desarrollo completo de hardware-software de una aplicación específica.

La CMUCAM3 tiene un alto potencial de procesamiento de imágenes, sin embargo no cuenta con las capacidades para manipular un gran número de periféricos, para solucionar este inconveniente de la tecnología, se configuró un ARDUINO MEGA que tiene los recursos necesarios para el manejo de dispositivos. Uniendo las bondades de cada uno de los sistemas fue posible desarrollar un SVA con capacidad de procesamiento de imágenes y manipulación de dispositivos.

Los programas implementados tanto en la CMUCAM3 como en el ARDUINO fueron desarrollados bajo lenguaje de código abierto, por lo anterior, se crea la posibilidad de que el usuario realice modificaciones internas al sistema, dando versatilidad y un gran campo de aplicación al SVA, ya que puede ser adaptado a diversas aplicaciones para las cuales en principio no está diseñado.

El protocolo de comunicación entre la CMUCAM3 y el ARDUINO puede ser ampliado, y depende directamente de los requerimientos adicionales que tenga el diseño, como la disposición de PLCs, control de sistemas AC, instrumentación de sensores, conexión a redes de datos, entre otras.

La capacidad de la CMUCAM3 es de 26 cuadros por segundo, sin embargo, el sistema embebido desarrollado disminuye esta capacidad en al menos el 70% para análisis de histogramas, en el 90% para almacenamiento y requiere mas de 3s para visualización.

Tener información píxel a píxel facilita el diseño de algoritmos basados en el conteo, de esta manera en la medida que la imagen esta siendo adquirida internamente por la cámara, esta siendo procesada al mismo tiempo. Para procesamiento y transformaciones en el espacio de la frecuencia es necesario utilizar el sistema FIFO, instalado en la CMUCAM3 y requiere mayor tiempo para la ejecución, el cual no fue medido en este trabajo.

Observando los tiempos requeridos para la operación de las funciones dentro el sistema embebido desarrollado, se generan prioridades sobre las funciones, para este caso la prioridad mas alta la tienen los procesos de detección de información sobre imágenes, prioridad media para la configuración de la cámara y prioridad baja para el almacenamiento y visualización.

A partir de la información adquirida por la CMUCAM3 es posible tomar decisiones frente a diferentes eventos, esta característica y las posibilidades de manipular sistemas electrónicos y mecánicos de manera independiente de un computador, generan aplicabilidad de este tipo de sistemas en industrias insipientes de tecnología que requieren soluciones a bajo costo, entre ellas se encuentra la industria agrícola, donde sería posible con sistemas como el descrito en este artículo, desarrollar sistemas para agricultura de precisión, control en procesos de cosecha y poscosecha.

AGRADECIMIENTOS

Los autores expresan sus agradecimientos al Departamento Nacional de Ciencia, Tecnología e Innovación – Colciencias, por la cofinanciación del proyecto titulado "Desarrollo de una herramienta portátil con visión artificial para la cosecha selectiva de café", sobre el cual se desarrollaron las actividades descritas en este artículo. De igual forma agradece a la Universidad Nacional de Colombia por la participación en este proceso a través del desarrollo de prácticas universitarias conducentes a trabajos de grado de uno de sus estudiantes.

REFERENCIAS

[1] ARM (Advanced Risc Machines). Hoja de datos del microprocesador ARM7TDMI. Agosto de 1995. [En línea]. Disponible en: http://web.eecs.umich.edu/~panalyzer/pdfs/ARM_doc.pdf

[2] Carnegie Mellon University, hoja de datos de la CMUCAM3. [En línea]. Disponible en: http://www.superrobotica.com/download/cmucam3/CMUcam3_datasheet.pdf

[3] Editor ATMEL, hoja de datos del ARDUINO MEGA 2560, [En línea]. Disponible en: http://www.atmel.com/Images/doc2549.pdf

[4] OMNIVISION TECHNOLOGIES, Inc. Hoja de datos del sensor OV6620. [En línea]. Disponible en: http://www.cs.cmu.edu/~cmucam/Downloads/ov6620DSLF.PDF.

[5] IC Insights, The MacClean Report 2014 (Resumen). [En línea]. Disponible en: http://www.icinsights.com/services/mcclean-report/report-contents/

[6] Jie ZHANG, Aicheng LI, Jianlong LI, Qi YANG, Chengcheng GANG. Research of Real-time Image Acquisition System Based on ARM 7 for Agricultural Environmental Monitoring. 978-1-4244-9171-1/11. 2011 IEEE. Nanjing, China.

[7] Dongxian He, Youlu Bai, Yingzhe Wang, Hua Wu. A Crop Field Remote Monitoring System Based on Web-server-embedded Technology and CDMA Service. Proceedings of the 2007 International Symposium on Applications and the Internet Workshops. 0-7695-2757-4/07. China 2007.

[8] Zhenyu Ye, Yifan He, Roel Pieters, Bart Mesman, Henk Corporaal, Pieter Jonker. An Embedded Vision System for High Frame Rate Visual Servoing. 978-1-4577-1707-9/11 IEEE. Holanda 2011.

[9] Mbed Microcontrollers. [En línea]. Disponible en: http://mbed.org/handbook/mbed-Microcontrollers

[10] BeagleBoard. [En línea]. Disponible en: http://beagleboard.org/

[11] TELEOBJETIVO, Así han revolucionado los microprocesadores. Blog sobre ciencia tecnología y fotografía. [En línea]. Disponible en: http://www.teleobjetivo.org/blog/asi-han-evolucionado-los-microprocesadores.html

[12] Raspberry pi. [En línea]. Disponible en: http://www.raspberrypi.org/