UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Estudio sobre la viabilidad de desarrollar aplicaciones gráficas en sistemas embebidos

Tesis de Máster en Inteligencia Artificial,

Reconocimiento de Formas e Imagen Digital (MIARFID)

Departamento de Sistemas Informáticos y Computación (DSIC)

Universidad Politécnica de Valencia (UPV)

Septiembre 2014

Cindy Johanna González Díaz

Directora:

M. Carmen Juan Lizandra

Estudio sobre la viabilidad de desarrollar aplicaciones gráficas en sistemas embebidos **1**

Autor: Cindy Johanna González Díaz, email: cingondi@upv.es
Directora: M. Carmen Juan Lizandra, email: mcarmen@dsic.upv.es

Agradecimientos

En primer lugar quiero agradecer a mi Madre por su apoyo, cariño, aliento y acompañamiento durante este proceso, sin su ayuda este trabajo no sería posible.

A mi directora de tesis M. Carmen, por darme la oportunidad de trabajar en este proyecto con una beca de colaboración, gracias por sus consejos y dirección.

A Ismael, por su compañía, cariño, consejo y apoyo que me permitieron seguir adelante con mi trabajo hasta el final.

A mis amigos Karina, Javier, Guillermo, Denuis, Irina, Gustavo y Azeddine por su apoyo en los momentos difíciles y por su consejo.

Un agradecimiento especial a mi amiga Lorena, gracias por su compañía, amistad, apoyo, consejo y aliento durante el desarrollo del Máster.

A mis compañeros Juan y Javier, gracias por el acogimiento, acompañamiento, consejo y por permitirme compartir con ustedes buenos momentos.

A toda la gente con la que compartí en Valencia y que hicieron de mi estancia allí algo muy especial.

A mi familia por su apoyo incondicional y consejo aún a la distancia.

Resumen

En los últimos años se ha producido un gran avance en la industria del juguete. Hoy en día gracias a los progresos en la tecnología utilizada los juguetes permiten un aprendizaje interactivo. Muchos de ellos integran aplicaciones visuales gráficas con interacciones en tiempo real. Hecho que facilita el aprendizaje. Hasta el momento en su mayoría, dichas aplicaciones han sido incorporadas a dispositivos móviles, tabletas, o PC, con algunas limitaciones, como la dependencia del modelo y sistema operativo, baja capacidad de procesamiento en algunos dispositivos, la imposibilidad de separar la cámara del dispositivo en el caso de interacción con el mundo real, poca adaptabilidad al espacio y forma del juguete, el coste, y finalmente la complejidad que conlleva la integración con otros elementos que generan movimiento, como motores o actuadores.

En este trabajo de fin de máster se ha realizado un estudio sobre la viabilidad de la construcción de un prototipo hardware que posibilite el desarrollo de aplicaciones gráficas embebidas, y que permita superar las limitaciones mencionadas. Inicialmente se realizó una búsqueda de diferentes dispositivos programables disponibles en el mercado que permitieran realizar esta tarea de una manera óptima. Entre ellos, cabe citar: las FPGAs (Field Programmable Gate Array) que combinan lo mejor de los circuitos integrados de aplicación específica (ASICs) y sistemas basados en procesador, los DSPs (Digital Signal Processing), los procesadores ARM Cortex y las diferentes plataformas para el desarrollo de aplicaciones de adquisición y procesamiento de imagen que combinan el uso de DSPs con los procesadores ARM, como la plataforma OMAP de Texas Instruments. Se optó por implementar una aplicación gráfica sobre la plataforma embebida BeagleBoard xM debido a las características que posee, entre otras, el ser una plataforma de desarrollo especializada en multimedia que se compone de un procesador ARM cortex y un DSP C64P, operada por un núcleo GNU/Linux, que permite acceder a los diferentes periféricos, como teclado, ratón o cámara web.

Se instalaron dos sistemas operativos en la BeagleBoard xM; Linux (Ubuntu) y Android. Además se utilizaron diferentes herramientas (Librerías - Toolkit) para el desarrollo de aplicaciones gráficas, como ARToolKit, OpenCV, entre otras, en el caso de Linux; para Android se trabajó con NyARToolKit, y otros. Por último, se implementó una aplicación con la herramienta de mayores prestaciones para ilustrar el funcionamiento y rendimiento.

Para concluir el trabajo, se calculó el rendimiento de cada sistema operativo y librería, se presentaron resultados comparativos que permitieron plantear recomendaciones finales en cuanto al hardware y software, para el prototipo de pruebas. Finalmente se plantearon las ventajas y desventajas de cada herramienta y se sugirieron algunas diferentes que permitirían mejorar el rendimiento de las aplicaciones gráficas embebidas.

Índice general

Índice de Figuras

Índice de Tablas

Acrónimos

CI	Circuito Integrado
DSP	Procesador Digital de Señal
FPS	Frames por Segundo
FPGA	Field Programmable Gate Array
OCR	Reconocimiento Óptico de Caracteres
RA	Realidad Aumentada
SO	Sistema Operativo
UVC	Controlador de Vídeo USB

Capítulo 1

Introducción

Las aplicaciones gráficas tienen un campo de acción muy amplio, tanto en el entretenimiento como a nivel industrial, llaman especial atención las aplicaciones interactivas que permiten al usuario intervenir en un mundo virtual. Dicha interacción puede ampliarse al mundo real utilizando un dispositivo de adquisición de vídeo, como una cámara, además si es posible modificar una escena real añadiendo objetos virtuales, Realidad Aumentada (RA), se abre un mundo de posibilidades de desarrollo. A través de los años se ha trabajado por mejorar el rendimiento de las aplicaciones gráficas y de los dispositivos hardware necesarios para ejecutarlas. En general se han utilizado ordenadores por su capacidad de procesamiento, y en los últimos años teléfonos móviles y tabletas para hacer dichas aplicaciones portables. Sin embargo para propósitos específicos, como en la industria del juguete, se requiere integrar estas aplicaciones en dispositivos con una mayor funcionalidad, posibilidades de programación, bajo coste y que sean adaptables a cualquier tipo de estructura.

Teniendo en cuenta las razones expuestas, se plantea el presente trabajo de fin de máster con el objetivo de evaluar la viabilidad del desarrollo de aplicaciones gráficas en dispositivos que puedan ser personalizados de acuerdo a requerimientos específicos, se trabajó con los sistemas embebidos debido a que su arquitectura es adecuada para este fin.

1.1. Motivación

En los últimos años se ha requerido de un mayor uso de aplicaciones gráficas en distintos campos, como en la industria juguetera, en el área de robótica, en el campo educativo, entre otros. Dichas aplicaciones permiten la percepción e interacción con el mundo virtual y en algunos casos se complementan con la interacción con el mundo real, donde el usuario puede estar en un entorno real con información generada por el ordenador. Algunas de estas aplicaciones incluyen: los videojuegos, el reconocimiento de formas o colores, OCR y en el área de RA; programación quirúrgica, visualización de imágenes médicas, guía en la fabricación y reparación, planificación de rutas en telerobótica, efectos especiales para propósitos de entretenimiento y plataformas educativas

como APRENDRA[1].

Debido a la necesidad de hacer las aplicaciones gráficas portables se ha optado por utilizar dispositivos móviles o tabletas para su ejecución. Sin embargo se ha visto que un dispositivo personalizado presentaría mayores ventajas, ya que sería posible adaptarlo a las necesidades específicas de cada área. Diferentes características de las aplicaciones gráficas motivan el uso de hardware reconfigurable, una de ellas es el procesamiento intensivo que se requiere ya que objetos sintéticos tienen que ser generados y en el caso de la RA deben ser integrados con vídeo en tiempo real. Por otro lado se sabe que el procesamiento de imágenes y los gráficos por computador son áreas donde se requiere la aceleración del hardware. Las aplicaciones gráficas, con frecuencia, involucran interacciones con el usuario en tiempo real o adaptaciones a las variaciones del entorno, por lo que la velocidad y la flexibilidad de ejecución de los procesadores reconfigurables tiene ventajas sobre los dispositivos con funciones fijas.

Desde hace varios años el procesamiento digital dejó de ser realizado en su mayoría, por ordenadores de uso personal, y pasó a ser tarea de circuitos electrónicos integrados en dispositivos comunes de uso diario, como los automóviles, electrodomésticos, juguetes, entre otros. Dichos circuitos electrónicos integrados, se denominan sistemas embebidos (Grüner, 2012). Con el fin de personalizar estos sistemas se han creado las plataformas de desarrollo, que son sistemas embebidos de propósito general con diversos periféricos y módulos incorporados. En este tipo de plataformas, los desarrolladores pueden concentrarse en probar algoritmos y aplicaciones sin tener que preocuparse por problemas de hardware (Grüner, 2012).

Inicialmente se realizó una búsqueda exhaustiva de diferentes dispositivos programables disponibles en el mercado, entre ellos cabe citar los de tipo FPGA (Field Programmable Gate Array) que combinan lo mejor de los circuitos integrados de aplicación específica (ASICs) y los sistemas basados en procesador, DSPs (Digital Signal Processing) y los procesadores ARM Cortex. Sin embargo, las plataformas de desarrollo mencionadas anteriormente combinan el uso de DSPs con los procesadores ARM, como la plataforma OMAP de Texas Instruments, lo que presenta ventajas sobre los demás dispositivos. Por esta razón para el presente trabajo se seleccionó dicha clase de plataforma. Luego de evaluar diferentes variables como coste y prestaciones se optó por trabajar con la Beagleboard xM, teniendo en cuenta que es una plataforma especializada en multimedia, se compone de un procesador ARM cortex y un DSP C64P, está operada por un núcleo GNU/Linux y permite el acceso a diferentes periféricos, como teclado, ratón o cámara web.

La Beagleboard xM tiene características similares a un ordenador, por lo tanto es posible trabajar con diferentes sistemas operativos instalados en dicha tarjeta. Se ha realizado el estudio utilizando dos sistemas operativos: Linux (Ubuntu) y Android. Cabe destacar, que son de código abierto y por tanto se tiene más posibilidades de personalizar la parte software. En el caso de Android, es el sistema operativo más utilizado en la actualidad, así que es posible encontrar numerosas herramientas de desarrollo y documentación. Por otro lado, se evaluaron diferentes librerías y herramientas para el desarrollo de aplicaciones gráficas, para Linux se realizaron pruebas con OpenCV y ARToolKit y en el caso de

[1] http://www.aprendra.es/

Android se utilizó NyARToolKit. Luego del estudio se han establecido las características que debe tener un sistema embebido, para la ejecución de aplicaciones gráficas, teniendo en cuenta variables como las herramientas disponibles en el mercado, rendimiento, funcionalidad, programación, eficiencia y facilidad de uso.

Finalmente, cabe mencionar que la principal motivación para el desarrollo de esta investigación está basada en los diferentes beneficios que se obtendrían al desarrollar un prototipo hardware que pueda ser personalizado y utilizado en la ejecución de aplicaciones gráficas. Teniendo en cuenta que el mundo académico ha empezado a introducir el uso de aplicaciones gráficas en algunas de sus disciplinas, el desarrollo de iniciativas en la utilización de estas aplicaciones en la educación y su divulgación, puede brindar grandes aportes para la comunidad docente. Por otro lado dichas aplicaciones podrían ser integradas en un juguete y dado que un gran número de empresas de la industria juguetera española se encuentra en Alicante, considerando su cercanía, y los lazos investigadores que unen a los miembros de grupos de investigación de la Universidad Politéncina de Valencia con AIJU (Instituto Tecnológico del Juguete) sito en Ibi (Alicante), se considera que podría ser una aportación muy importante para este sector. Otro sector que podría beneficiarse de estas aplicaciones sería el área de robótica. Por ejemplo, para vehículos de conducción asistida en los que la RA supondría una gran mejora en la seguridad del conductor, ya que sería posible mostrar la trayectoria que debería llevar la carretilla. Finalmente, existen muchas otras aplicaciones industriales de inspección o reconocimiento de productos, donde sería viable incorporar este tipo de dispositivos.

1.2. Objetivos

El principal objetivo del presente trabajo de investigación consiste en realizar un estudio sobre la viabilidad del desarrollo y ejecución de aplicaciones gráficas en sistemas embebidos, con el fin de construir un prototipo hardware que pueda ser personalizado de acuerdo a requerimientos específicos. Para ello se tienen en cuenta dos aspectos principales: el hardware y el software del prototipo.

En cuanto al hardware, se requiere de un prototipo funcional, de un tamaño adecuado, que permita ser fácilmente integrado en un juguete, automóvil, electrodoméstico o en un dispositivo industrial. Además, es fundamental que sea posible adaptar una pantalla táctil para una mejor interacción del usuario, por otro lado es importante que el dispositivo tenga conexión inalámbrica a Internet. Finalmente, deberá contar con una batería recargable, para aumentar la portabilidad del prototipo.

En cuanto al software, es necesario realizar una búsqueda exahustiva de herramientas libres disponibles para la programación de los sistemas embebidos.

Para el desarrollo de aplicaciones gráficas existen múltiples herramientas libres para diferentes sistemas operativos, se deberá realizar una investigación previa, para seleccionar las más adecuadas para ejecutar las pruebas en el prototipo.

Una vez construido el prototipo y realizadas las pruebas, se implementará una aplicación sencilla para evaluar el funcionamiento y rendimiento del prototipo, de esta manera se conocerá si es viable implementarlo en las aplicaciones propuestas.

Finalmente, se deberá hacer un análisis comparativo entre las diferentes herramientas investigadas y ejecutadas en el prototipo, con el fin de señalar las ventajas y desventajas de cada una, para así plantear unas recomendaciones finales.

Con el fin de alcanzar el objetivo principal del presente proyecto, se han planteado los siguientes objetivos específicos:

- Para empezar, se hará una revisión de los sistemas embebidos disponibles en el mercado. Se seleccionará el sistema de mayores prestaciones, teniendo en cuenta las características de procesamiento, rendimiento de gráficos y herramientas de programación. Con el sistema seleccionado se construirá un prototipo de pruebas.

- Realizar un estudio sobre los trabajos previos, relacionados con el desarrollo e implementación de aplicaciones gráficas en sistemas embebidos, de esta manera se conocerán los sistemas más utilizados, el tipo de aplicaciones desarrolladas, el rendimiento alcanzado y las herramientas software disponibles para su programación.

- Construir un prototipo de pruebas con diferentes accesorios, como una pantalla táctil, botones de navegación, conexión inalámbrica (WiFi), cámara integrada y batería recargable, de esta manera se conseguirá un prototipo funcional y portátil.

- Desarrollar un estudio sobre las diferentes herramientas libres de programación disponibles, para el sistema embebido seleccionado. Escoger por lo menos dos para instalar en el prototipo y de esta manera conocer y solventar problemas de instalación y configuración, así como los alcances y limitaciones de cada herramienta.

- Hacer un análisis comparativo entre las dos herramientas de programación seleccionadas en el punto anterior, y plantear las ventajas y desventajas de cada una. Realizar recomendaciones finales.

- Realizar un estudio sobre las herramientas software libres para el desarrollo de aplicaciones gráficas, y seleccionar algunas para ejecutar pruebas de gráficos 2D, 3D y para programar e implementar aplicaciones gráficas de interacción con el mundo real que requieran el uso de una cámara, en el prototipo de pruebas.

- Seleccionar la herramienta software más adecuada de acuerdo a los resultados del punto anterior para implementar una aplicación sencilla en el prototipo, de manera que pueda ser evaluado su rendimiento y funcionamiento.

- Evaluar la viabilidad de utilizar el prototipo construido, en las aplicaciones planteadas en un principio y realizar sugerencias para mejorar el prototipo y para trabajos futuros relacionados.

1.3. Estructura del Documento

El resto del presente trabajo de fin de máster está organizado en los siguientes capítulos:

Capítulo 2: Se plantea el estado del arte sobre los temas relacionados con el trabajo de investigación: conceptos básicos sobre sistemas embebidos, dispositivos hardware, aplicaciones gráficas, herramientas libres para la programación de aplicaciones gráficas y trabajos previos sobre aplicaciones gráficas en sistemas embebidos.

Capítulo 3: Se describe la construcción del prototipo hardware con todos los accesorios que lo componen, como la pantalla táctil, los botones de navegación, la antena WiFi, la cámara, la batería recargable y otros periféricos.

Capítulo 4: Se describen detalladamente las pruebas realizadas con el sistema embebido seleccionado y con las diferentes herramientas software. Se ejecutan varias aplicaciones y se hace un análisis del funcionamiento de la herramienta hardware. Finalmente, se presentan los resultados obtenidos en las diferentes pruebas y un ejemplo sencillo de aplicación.

Capítulo 5: Se concluye con las principales contribuciones del trabajo y se presentan los trabajos futuros de investigación.

Capítulo 2

Estado del Arte

En este capítulo se hace una breve introducción a la historia y conceptos generales sobre los sistemas embebidos. Se definen las posibles aplicaciones gráficas para dichos sistemas, teniendo en cuenta algunos trabajos previos relacionados. Finalmente se describen las herramientas hardware y software más relevantes para la programación de las aplicaciones.

2.1. Introducción a los Sistemas Embebidos

Se han planteado varias definiciones para los sistemas embebidos. Para los propósitos del presente trabajo, un sistema embebido se puede definir, como un producto electrónico que contiene uno o más procesadores y software integrado para ejecutar funciones específicas (Kleidermacher and Kleidermacher, 2012). En el caso de ejecución de aplicaciones gráficas, el sistema debe contar con un dispositivo de salida o inferfaz que represente visualmente la información gráfica virtual o de su entorno en tiempo real, y que además permita la interacción del usuario con la máquina.

Con el fin de entender la funcionalidad de un sistema embebido y sus repercusiones, es importante conocer algo de su historia:

En los años 40 se desarrollaron los primeros ordenadores programables (ENIAC, Z3, etc.) (Peddie, 2013). En 1947, se implementó el primer transistor funcional en los laboratorios Bell (Brinkman et al., 1997). En 1959, Jack S. Kilby creó el primer circuito integrado (CI) funcional (Brock and Laws, 2012). En 1971, Texas Instruments presenta el TMS 1000 e Intel el 4004, los dos candidatos para ser el primer microprocesador de propósito general (Faggin et al., 1996). Luego se desarrollaron los DSP (Digital Signal Processor), que son microprocesadores optimizados para el procesamiento de señales digitales (Parker, 2010). Ya en los años 80, los sistemas embebidos son asequibles para ser dispositivos de consumo y son integrados a cohetes (Wolf and Madsen, 2000). Más tarde, en la misma década se crean los microcontroladores que consisten en un chip sencillo, con un procesador, memoria RAM y controladores (entrada/salida), los microcontroladores tienen menor coste. Pero, son menos flexibles que los microprocesadores de propósito general. El desarrollo de los microcontroladores causa el auge de los sistemas embebidos en los años 80 (Pal, 2012). A mediados de los 80 se crean los FPGA (Field Programmable

Gate Array); chips de Silicio reprogramables para funciones personalizadas, que trabajan a una gran velocidad (Parker, 2010). Más tarde, en 1987 se vendió el primer ordenador basado en un procesador ARM; el Arcon Archimedes. Durante la década de los 90 después de la salida al mercado de los ordenadores portátiles, diferentes compañías se dedicaron a desarrollar CPUs cada vez más potentes. Sin embargo ARM orientó sus investigaciones a la creación de una arquitectura que proporcionara soluciones para que los usuarios fueran capaces de construir sus propios sistemas. Hoy en día estamos en medio de una revolución móvil y los procesadores ARM están integrados en la mayoría de los teléfonos móviles y tabletas. ARM ha creado procesadores gráficos incluso más potentes que algunas tarjetas gráficas para ordenadores de escritorio, utilizando una fuente de alimentación de bajo voltaje (Langbridge, 2014).

En los últimos años el procesamiento digital ha dejado de ser realizado en su mayoría, por ordenadores de uso personal, y pasó a ser tarea de sistemas embebidos en aparatos comunes de uso diario, como los automóviles, electrodomésticos e incluso juguetes; o de uso industrial, como robots y controladores (Grüner, 2012).

Un sistema embebido tiene la función de cumplir un número limitado de procesos o tareas predefinidas. A continuación se mencionan algunas de sus principales características:

- Son sensibles al coste, de acuerdo a sus alcances y componentes.
- La mayoría de sistemas embebidos tienen limitaciones en tiempo real.
- Existe una gran variedad de arquitecturas de CPU para sistemas embebidos, por ejemplo:
  - ARM®
  - MIPS®
  - PowerPCTM
- Utilizan procesadores con diseños específicos para cada aplicación.
- El manejo de energía es crítico en la mayoría de sistemas embebidos.
- Normalmente cuentan con circuitos de depuración construidos internamente.
- Se diseñan teniendo en cuenta la perspectiva de hardware y software.

Normalmente, los sistemas embebidos mencionados pueden ser programados con lenguajes genéricos como C o Java, o específicos como el lenguaje ensamblador o el VHDL. Sin embargo en la actualidad existen gran cantidad de lenguajes y herramientas para realizar desarrollos gráficos o de otro tipo, que no pueden ser utilizadas en dichos sistemas. Es por ello que se crearon las plataformas de desarrollo. Dichas plataformas son sistemas embebidos de propósito general, que combinan la potencia de un ordenador corriente y la capacidad de incorporar periféricos y módulos e instalar diferentes SO con la portabilidad de un dispositivo móvil. En este tipo de plataformas los desarrolladores pueden

concentrarse en hacer pruebas con algoritmos y aplicaciones sin tener que preocuparse por problemas de hardware (Grüner, 2012).

Se realizó una búsqueda exhaustiva de sistemas embebidos disponibles en el mercado actualmente, profundizando en las plataformas de desarrollo, una vez analizadas sus ventajas. A continuación se mencionan los productos más relevantes para el objeto de estudio del presente trabajo.

2.1.1. FPGAs

Las FPGAs son chips de silicio reprogramables utilizados ampliamente en la industria debido a que combinan lo mejor de los circuitos integrados de aplicación específica (ASICs) y sistemas basados en procesador[1]. Las FPGAs ofrecen velocidades temporizadas por hardware y fiabilidad, pero sin requerir altos volúmenes de recursos.

La gran ventaja de las FPGAs sobre los procesadores normales, es que tienen la misma flexibilidad que un software que se ejecuta en un sistema basado en procesador. Pero no está limitado por el número de núcleos de procesamiento disponibles. Por otro, lado los FPGAs procesan las tareas de forma paralela. Así las diferentes operaciones de procesamiento no tienen que competir por los mismos recursos. Cada tarea de procesamiento independiente es asignada a una sección del chip y puede ejecutarse de manera autónoma sin ser afectada por otros bloques de lógica. Como resultado, el rendimiento de una parte de la aplicación no se ve afectado cuando se agregan otros procesos.

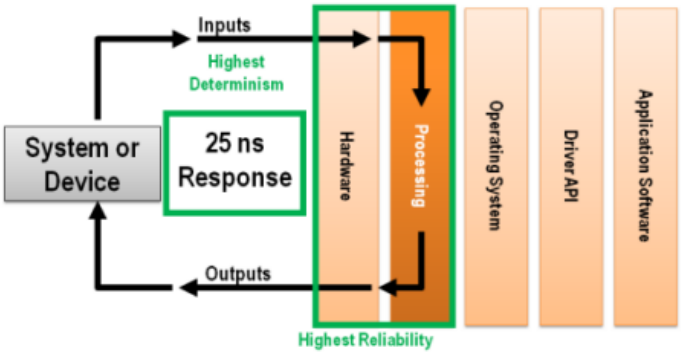

Figura 2.1: Diagrama de funcionamiento de una FPGA.

La principal ventaja de utilizar una FPGA para el desarrollo de aplicaciones gráficas, en lugar de sistemas basados en procesador, es que la lógica de aplicación sería implementada en circuitos de hardware en lugar de ejecutarse aparte de un Sistema Operativo, controladores y software de aplicación. Hecho que mejoraría la velocidad de procesamiento, ejecución, y captura de la imagen en tiempo real.

Sin embargo, se presentarían problemas al utilizar únicamente un FPGA para el procesamiento y la conectividad de E/S en el sistema. Teniendo en cuenta que las FPGAs no tienen el ecosistema controlador y base del código IP que tienen las arquitecturas de los

[1]National Instruments NI. Fpgas a fondo. http://www.ni.com/white-paper/6983/es/

sistemas operativos, para los periféricos, como la WebCam y la pantalla de visualización. Por esta razón se ha optado por desarrollar arquitecturas híbridas en plataformas de desarrollo. Además, sería necesario crear librerías específicas para el desarrollo de aplicaciones gráficas en una FPGA, lo que representa un trabajo exhaustivo y no permite utilizar las herramientas software disponibles para este fin.

2.1.2. Plataforma OMAP

La plataforma OMAP de Texas Instruments es una buena opción para la implementación de aplicaciones gráficas debido que posee una cámara ISP de buena calidad y un potente procesador ARM MPCores, lo que genera una alta capacidad de procesamiento gráfico.

Un ejemplo de la tecnología OMAP son los procesadores OMAPTM 3 que están basados en procesadores ARM Cortex/A8 core y TIs C64x + DSP. Tienen una velocidad de hasta 720MHz. Permiten el desarrollo de una interfaz de usuario avanzada, tienen mejoras en los gráficos, el vídeo y la conectividad para las aplicaciones[1].

Figura 2.2: Plataforma de desarrollo OMAP35X con pantalla LCD.

En la Figura 2.2 se presenta la plataforma de desarrollo OMAP35X. Dicha plataforma está diseñada con una arquitectura modular y extensible compuesta por cuatro módulos. Cuenta con un procesador ARM/Cortex A8, con un DSP de C64x y tiene integrada una pantalla LCD de 7 pulgadas.

El coste sin embargo, es bastante elevado, está en alrededor de 1.300 Euros.

Existen diferentes plataformas de desarrollo de este estilo que son óptimas para el desarrollo de aplicaciones gráficas. Dichas plataformas en su mayoría hacen uso de procesadores ARM combinados con procesadores DSP.

2.1.3. Arduino Due

Las placas de Arduino se conocen principalmente como ordenadores con una sola tarjeta de 8 bits. Son una excelente herramienta en el campo de la electrónica y el desarrollo

[1] http://www.ti.com/

embebido. La familia Arduino viene con un rango completo de módulos, que van desde puertos E/S hasta el almacenamiento de datos en tarjetas SD. Arduino ha sido utilizado en una gran cantidad de proyectos, desde controles para acuarios, hasta robótica para automatizar cortadoras de césped (Oxer and Blemings, 2009).

Aunque una generación entera de Arduino ha utilizado un microcontrolador PIC de 8 bits, Arduindo Due, utiliza un microcontrolador Cortex-M. Cabe mencionar que estas tarjetas no están diseñadas para un procesamiento potente, incluso si la CPU ARM es de 84 MHz. Pero, están diseñadas para proyectos electrónicos basados en E/S. Cuenta con 54 puertos digitales E/S que pueden ser programados como entrada o salida, 12 entradas análogas y 2 salidas análogas.

Las tarjetas de Arduino se pueden encontrar en una gran cantidad de proyectos relacionados con robótica o sensores, simplemente porque el procesador está fuertemente basado en E/S. También son altamente populares porque están basadas en un Cortex-M. Por otro lado, tienen un manejo eficiente de la energía. Es común ver a las tarjetas Arduino trabajar con una batería. El sistema Arduino ha sido usado en robótica móvil o en sistemas de piloto automático para el control remoto de aviones (Langbridge, 2014).

En la Figura 2.3, se observa la tarjeta Arduino Due[1].

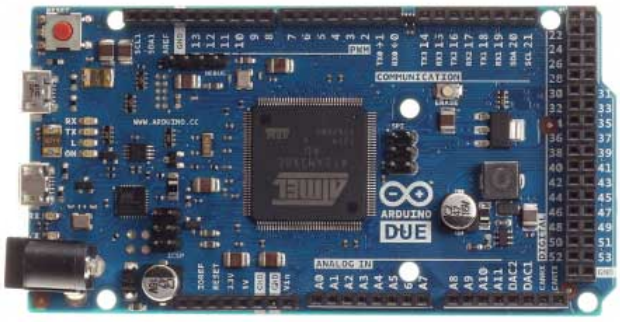

Figura 2.3: Arduino Due.

2.1.4. Raspberry Pi

Es una placa ordenador de muy bajo coste. Tiene el tamaño de una tarjeta de crédito. Desarrollada en el Reino Unido por la fundación Raspberry Pi, con la intención de estimular la enseñanza de programación básica en los colegios[2]

La placa está basada en los procesadores ARM y puede ser utilizada para la mayoría de funciones que realiza un PC, como hojas de cálculo, procesamiento de palabras y juegos. También permite reproducir vídeos en alta definición.

La Raspberry Pi es una plataforma Open Source que permite descargar e instalar un sistema operativo como Linux o Windows y también algunas herramientas de software, como Ubuntu Linux, Mozilla Firefox, Office package Koffice (Análogo a Word/Excel sobre Windows), Python y Vídeo en formato HD. Una de las principales ventajas de esta placa

[1] http://arduino.cc/en/Main/ArduinoBoardDue

[2] The Raspberry PiFoundation. What is a raspberry pi? http://www.raspberrypi.org/

Figura 2.4: Raspberry Pi.

es la relación precio / calidad ya que el coste es de alrededor de unos 40 Euros y permite una gran variedad de aplicaciones de buena calidad.

Cuenta con un procesador ARM11 de 700MHz, con una memoria RAM de 512MB SDRAM. Por otro lado tiene incluida una variante de la API gráfica OpenGL que está diseñada para dispositivos integrados tales como teléfonos móviles, PDAs y consolas de videojuegos, y el vídeo tiene una resolución de 1080p30.

A pesar de las excelentes características de esta placa, las aplicaciones gráficas requieren de una unidad más potente de procesamiento gráfico. Por esta razón se optó por buscar una plataforma con mayores prestaciones en este aspecto.

2.1.5. BeagleBoard

BeagleBoard.org es una organización sin ánimo de lucro creada para proporcionar educación y promocionar el diseño y el uso de hardware y software libre, en computación embebida[1]. Dicha organización se ha dedicado a crear placas de desarrollo de bajo coste, que funcionan como un ordenador sin ventilador y con una única tarjeta, su procesador está basado en los procesadores serie ARM Cortex-A de baja potencia, desarrollados por Texas Instruments. Dichos procesadores poseen toda la capacidad de expansión de las máquinas de escritorio.

Existen diferentes versiones de las placas desarrolladas por la organización, entre ellas cabe citar; la BeagleBone, la BeagleBone Black y la BeagleBoard. La última versión de las más potentes, disponible en el mercado es la BeagleBoard xM, que se describe detalladamente a continuación:

La BeagleBoard xM es una plataforma de desarrollo especializada en multimedia; consta de un procesador ARM Cortex A8 como procesador de propósito general y un segundo procesador DSP C64x, especializado en procesamiento digital de señales.

En la Figura 2.5, se observan las características de la BeagleBoard xM; como se ilustra, la BeagleBoard xM cuenta con cuatro puertos USB para la conexión de diferentes

[1]jkridner.wordpress.com. What is beagleboard-xm? http://beagleboard.org/beagleboard-xm

periféricos, como un teclado, un ratón o una cámara web.

En la versión más potente de la BeagleBoard xM, el procesador tiene una velocidad de 1GHz y una memoria de 512MB de bajo voltaje DDR RAM, la conectividad es soportada por un puerto Ethernet 10/100. Tiene la posibilidad de conectarse a la pantalla de un monitor o televisor a través de un puerto HDMI o de S-Video. Por otro lado tiene un slot para una microSD de hasta 16 GB, donde va contenida toda la programación del sistema.

En cuanto al software, es open source y compatible con Angstrom Linux, Android, Ubuntu, XBMC y Windows CE.

Una de las principales ventajas de esta plataforma de desarrollo es su coste reducido, que es de aproximadamente 150 Euros.

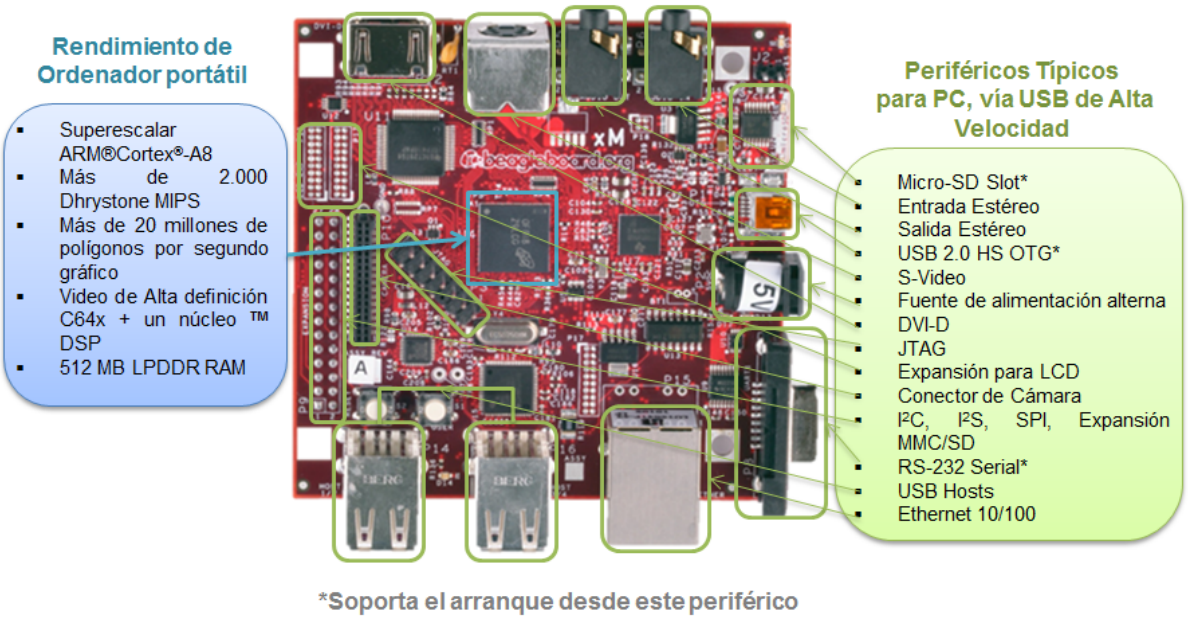

Figura 2.5: BeagleBoard xM.

2.2. Herramientas para la Programación de Sistemas Embebidos

En esta sección se hace una revisión de las herramientas disponibles para la programación de los diferentes sistemas embebidos mencionados en el apartado anterior.

2.2.1. Programación de FPGAs

Tradicionalmente la programación de las FPGAs se ha realizado con lenguajes de bajo nivel, de descripción de hardware (HDLs), como VHDL y Verlog. Sin embargo, el surgimiento de herramientas de diseño de alto nivel gráfico (HLS), como LabVIEW, ha

facilitado la programación de las FPGAs para los usuarios sin experiencia, además, la propiedad intelectual anterior no es requerida. Por lo tanto es posible integrar código VHDL existente con LabVIEW para los diseños con FPGAs. Por otro lado, las herramientas LabVIEW FPGA automatizan el proceso de compilación, de esta manera se facilita el depurar y verificar errores en el código de programación.

2.2.2. Programación de Arduino Due

La tarjeta de Arduino Due debe ser programada con un software específico desarrollado por Arduino. Disponible para su descarga gratuita, en la página web de la corporación. En dicha página se pueden encontrar tutoriales y soporte para el uso del software. El software Arduino se instala en un entorno de Windows, Linux o iOS, ejecutado en un ordenador. Se utilizan los dos puertos USB disponibles en la placa para realizar la programación. El código fuente del software es libre, está disponible y puede ser modificado por el usuario.

2.2.3. Sistemas Operativos Embebidos

Las plataformas que tienen integrado un procesador ARM, como la BeagleBoard y otras mencionadas en el apartado anterior, funcionan bajo un sistema operativo.

El sistema operativo es el cerebro del sistema embebido. Un sistema operativo permite al usuario concentrarse en el desarrollo de la aplicación, dejando de lado detalles de hardware. La configuración de memoria, la conexión a la red, y los periféricos de Entrada/Salida pueden ser manejados directamente por el sistema operativo. Existen diferentes opciones para los sistemas embebidos, cada una con ventajas y desventajas, a continuación se mencionan las disponibles (Langbridge, 2014).

Linux

Linux ha sido utilizado en los sistemas ARM por décadas. Tiene una gran cantidad de usuarios, y la posibilidad de compilar el kernel de origen; que es una gran ventaja para los sistemas embebidos. El kernel se puede configurar de tal manera que se habiliten únicamente las opciones de hardware mínimas y eliminar las que no son necesarias. Por otro lado, si se requiere añadir nuevo hardware, existen múltiples fuentes para obtener los drivers, y también es posible que la comunidad programadora de código abierto haya desarrollado el driver requerido. Para los sistemas embebidos están disponibles, las distribuciones Angstrom y Ubuntu.

VxWorks

Es un sistema operativo en tiempo real, con un kernel multitarea y soporte de un multiprocesador. Es utilizado en una gran cantidad de sistemas críticos para misiones complejas, donde la fiabilidad es una prioridad. VxWorks potencia numerosos proyectos espaciales.

Android

Android es un sistema operativo basado en Linux, diseñado inicialmente por Android Inc. y luego adquirido por Google. La mayoría del desarrollo para Android se hace en

el entorno en tiempo de ejecución de Java. Sin embargo es de código abierto y se puede adquirir de manera gratuita. Cabe mencionar que antes lanzar al mercado la máquina virtual de Java, existían múltiples aplicaciones escritas en C, además del kernel de linux, que aún requieren entornos de programación de bajo nivel.

iOS

Es posible ejecutar iOS en los procesadores ARM, con algunas extensiones de Apple. Sin embargo el sistema operativo es privado, lo que significa que no se puede tener acceso al código fuente, de arranque. Las aplicaciones de Apple iOS están escritas en Objective-C. Pero también se pueden escribir en lenguaje ensamblador, ya sea escribiendo código o añadiendo un fichero ensamblador. Este proceso permite desarrollar aplicaciones altamente optimizadas.

Windows CE

Windows CE es un sistema operativo diseñado por Microsoft, especialmente para sistemas embebidos. Está optimizado para sistemas con un mínimo de memoria. Algunas herramientas para el desarrollo de aplicaciones en Windows CE son; Visual Studio, la versión libre de Pascal y Lazarus, Platform Builder que se utiliza para modificar el kernel, entre otros.

2.3. Aplicaciones Gráficas en Sistemas Embebidos

La computación gráfica está relacionada con la generación y síntesis de objetos reales o artificiales a partir de modelos geométricos y físicos (Foley, 1996); abarca una gran variedad de tópicos, como el procesamiento de imágenes, la edición de fotografía, numerosos videojuegos para ordenador y consolas, RA, visión por computador y también efectos especiales para el cine y la televisión. Incluso un procesador de palabras puede ser clasificado en esta categoría (Ferguson, 2013).

La computación gráfica es en gran parte interactiva, el usuario controla el contenido, la estructura y la apariencia de los objetos o de las imágenes, utilizando dispositivos como, el teclado, el ratón o un panel táctil (Foley, 1996).

Teniendo en cuenta el gran campo de acción de los desarrollos relacionados con la computación gráfica, ha surgido la necesidad de crear aplicaciones gráficas portables que sean integradas en objetos como: GPS para automóviles, con el objetivo de visualizar rutas o información aumentada del entorno; en juguetes para visualizar animaciones; en controladores industriales para visualizar información sobre variables de temperatura o presión; en electrodomésticos para interactuar con el sistema; y en diversos dispositivos con aplicaciones visuales específicas en general.

En esta sección se analizan algunas propuestas y trabajos relacionados con la implementación de sistemas gráficos embebidos, desarrollados en los últimos años. Se profundiza en el uso de las plataformas de desarrollo, en especial la BeagleBoard xM, teniendo en cuenta que es una de las más potentes e idónea para el objeto de estudio del presente trabajo.

A principios del año 2009, unos meses después del lanzamiento de la primera tarjeta de BeagleBoard, el laboratorio de Nueva Zelanda, HITLabNZ (EmbeddedAR, 2011), comenzó con la tarea de desarrollar aplicaciones de RA embebidas. Se generó un gran movimiento en el mercado de los Smartphone, con la llegada de nuevas y potentes plataformas como el iPhone, Palm-Pre y Android. El hardware de estos dispositivos se desarrolló en una nueva generación creciente de procesadores embebidos y chips. Por lo tanto, la investigación se centró en desarrollar procesadores embebidos potentes para el procesamiento multimedia, aprovechando las ventajas de los sistemas tradicionales y conservando las características de portabilidad y de duración de la batería de un sistema embebido.

El proyecto realizado, consistió en el desarrollo de una librería de RA para ejecutar aplicaciones en la plataforma BeagleBoard, se basaron en el ARToolKit y la API gráfica de OpenGL. Los resultados obtenidos se presentan en la Figura 2.6.

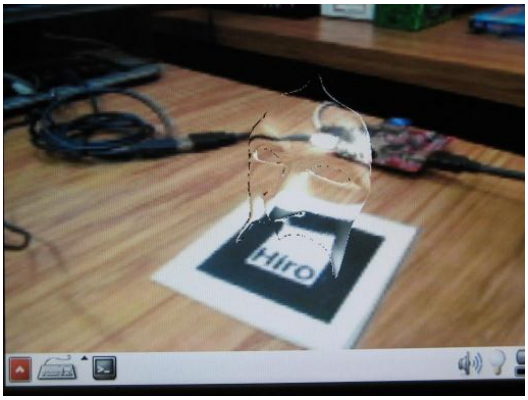

Figura 2.6: RA Embebida.

Uno de los primeros trabajos publicados sobre, aplicaciones gráficas implementadas en una tarjeta de BeagleBoard fue el de Poudel and Shirvaikar (2010), donde se menciona que las aplicaciones de visión por computador en tiempo real, como el vídeo streaming en los móviles, la vigilancia remota y la realidad virtual tienen requerimientos estrictos para un funcionamiento óptimo. Por lo tanto están inmensamente restringidas por recursos limitados de los dispositivos hardware. Es por ello que el uso de algoritmos optimizados es vital para lograr estos requerimientos, especialmente, en las plataformas embebidas. Dicho trabajo de investigación consistió en realizar una evaluación comparativa y optimizar el desempeño de algoritmos comunes para visión por computador en el procesador Intel Pentium, en la BeagleBoard y en una plataforma basada en un procesador OMAP y un DSP. Se utilizó la librería especializada en visión por computador OpenCV. Finalmente se concluyó que para obtener un alto rendimiento en tiempo real, es necesario optimizar los algoritmos utilizando técnicas específicas de programación, también se obtuvo un mejor rendimiento con la plataforma de arquitectura similar a la BeagleBoard xM.

En Aby et al. (2011) se implementa y optimiza un sistema embebido para la detección de rostros. Se hace uso de algoritmos optimizados como Viola Jones. Finalmente se realiza una evaluación comparativa de la aplicación, utilizando un procesador Intel y una BeagleBoard xM. Al igual que en el trabajo anterior, se hace uso de OpenCV, librería

desarrollada por la corporación Intel. Con la BeagleBoard xM se obtuvo un tiempo de ejecución de 0.95 segundos; tiempo solo un poco superior al tiempo de ejecución de un ordenador, que fue de 0.5 segundos. Se concluye que la arquitectura de la BeagleBoard xM (ARM + DSP) es la ideal para ejecutar aplicaciones en tiempo real.

Teniendo en cuenta el impacto del tráfico de vehículos para la estabilidad social y el desarrollo de la comunidad; ya que sin un sistema de control apropiado para la señalización, las posibilidades de congestión de vehículos se incrementan, Al Afif et al. (2011) pensaron en diseñar un sistema automático para el control de las señales de tráfico utilizando un sensor de cámara de vídeo, integrado a una BeagleBoard xM. El sistema utiliza el método de Viola Jones y el entrenamiento Haar para la detección de un objeto vehículo en un frame de vídeo. Luego, para el rastreo del vehículo se utiliza el cálculo de la distancia euclidiana y el filtro de Kalman. La habilidad del filtro de Kalman para predecir la próxima posición del objeto es una característica muy importante del rastreo multiobjetivo. El conteo de los vehículos se realizó utilizando técnicas de lógica difusa. Los algoritmos se implementaron con ayuda de la librería OpenCV. La aplicación implementada en la BeagleBoard permitió desarrollar un sistema ligero y fiable para una futura optimización.

En el trabajo de Lipinski et al. (2012) se integra la nueva generación de los sensores de imagen 3D a los sistemas embebidos. Se desarrolló una plataforma robótica, denominada RoboKinect, de bajo coste para visión 3D. RoboKinect está compuesta por una Kinect de Microsoft, una BeagleBoard y un microcontrolador. Se utiliza código abierto para la programación del sistema. Como valor agregado se implementa una nueva técnica para la detección de objetos 2.5D con el mapa de profundidad de la Kinect. Para la programación de la BeagleBoard se utilizó la distribución de Angstrom y OpenCV para ejecutar el procesamiento de imágenes. De este modo se evitó la necesidad de utilizar el DSP integrado en la BeagleBoard. El RoboKinect es un sistema asequible, con un coste inferior a los 600 dolares que puede ser vendido como dispositivo de enseñanza para las universidades o como juguete para propósitos de vigilancia.

Teoh et al. (2012) indican, que la visión por computador es un campo que incluye métodos para la adquisición, el procesamiento y la interpretación de imágenes. En el mundo embebido, la visión por computador tiene que enfrentarse con la baja potencia de procesamiento y recursos limitados para lograr algoritmos optimizados y un alto rendimiento. En dicho trabajo se implementó un sistema de rastreo de personas utilizando un ordenador de escritorio con una plataforma Intel y a su vez un sistema embebido para optimizar los algoritmos y así obtener un alto rendimiento. Los algoritmos son evaluados en la plataforma de procesador Intel y en la BeagleBoard xM. Utilizaron las librerías de OpenCV para construir los algoritmos de rastreo. Se obtuvieron buenos resultados al utilizar la BeagleBoard xM en cuanto a velocidad de procesamiento además de lograr la potabilidad de la aplicación.

Desde hace algunos años se está dando una convergencia entre la realidad virtual y la electrónica de consumo. Basu et al. (2012) presentaron una interfaz 3D inmersiva, integrada en un dispositivo que se ubica en la cabeza del usuario. El diseño consta de una BeagleBoard xM, que es el sistema de control central del sistema, un iPod touch que se encarga de rastrear la orientación del usuario mediante el uso de un acelerómetro, unas

gafas de vídeo para la visualización, un control de videojuegos con detector de movimiento y orientación, y una batería. El sistema tiene un coste inferior a los 900 dólares, es robusto, portable, liviano y no requiere de cables. El dispositivo tiene un buen rendimiento en el procesamiento. Sin embargo presenta algunos errores de precisión, al orientar el usuario.

La visión artificial sirve de apoyo para los sistemas de navegación, por medio de imágenes de profundidad y del reconocimiento de objetos. En el trabajo de Celis and Molano (2012) se diseñó un sistema de navegación para un robot móvil, utilizando una Kinect de Microsoft, OpenCV y Arduino. Se utilizó la Kinect para la captura de la imagen RGB y la imagen de profundidad. Los resultados obtenidos en la investigación indican que las imágenes de profundidad capturadas por la Kinect requieren de escenarios con iluminación controlada. Este aspecto es compensado con la creación del algoritmo de navegación con procesamiento digital de imágenes. En el trabajo futuro de este proyecto se desea realizar todo el procesamiento de las imágenes con un procesador ARM; se contempla la BeagleBoard xM como una posibilidad.

Majumder and Rathna (2013) desarrollaron una aplicación muy interesante, un dispositivo para el control de un ordenador mediante gestos, utilizando una BeagleBoard xM y una cámara web para la adquisición de la imagen. El objetivo del trabajo consistía en implementar una nueva forma de interacción con el ordenador, reemplazando el ratón y el teclado por el aparato diseñado. El dispositivo tuvo una velocidad máxima de 15 fps con una resolución de la imagen de entrada de 640x480, presentó un buen comportamiento en la ejecución de tareas básicas.

Solak and Bolat (2013) realizaron una aplicación industrial para la detección del color de diferentes productos. El sistema desarrollado consta de una BeagleBoard xM, una cámara USB, una banda transportadora de objetos y una pantalla LCD táctil de 7 pulgadas. Para la programación de la aplicación se utilizó OpenCV. El sistema es capaz de detectar cualquier tonalidad de color y varias formas básicas de objetos. Se obtuvo un excelente rendimiento y una alta velocidad de procesamiento en la aplicación. El sistema diseñado es de bajo coste y de acuerdo a las prestaciones mostradas, es viable implementarlo en procesos de inspección industriales.

De los antecedentes de propuestas y proyectos mencionados, se puede concluir que en su mayoría han tenido un buen desempeño al utilizar alguna tarjeta de BeagleBoard como unidad central de procesamiento. Por otro lado la integración de esta placa ha permitido la portabilidad de diferentes dispositivos con aplicaciones gráficas embebidas de todo tipo, a un bajo coste y con un bajo consumo de energía. La librería más óptima para el procesamiento de imágenes es sin duda OpenCV, que además cuenta con diferentes funciones y herramientas útiles para cualquier tipo de aplicación de visión artificial. El sistema operativo más utilizado para la programación de la BeagleBoard fue la distribución de Linux de Angstrom y Ubuntu. Finalmente, cabe mencionar que se implementaron diferentes algoritmos optimizados para incrementar la velocidad de procesamiento de las aplicaciones, y así no depender totalmente de las características de aceleración gráfica del dispositivo hardware.

2.4. Herramientas para la Programación de Aplicaciones Gráficas Embebidas

Como se menciona en la Sección 2.2.3, los sistemas embebidos basados en procesador ARM funcionan bajo un sistema operativo, en esta sección se hace una revisión de las herramientas libres disponibles y los lenguajes de programación; que posibilitan el desarrollo de aplicaciones gráficas embebidas.

En el área de visión artificial y procesamiento de imágenes, una herramienta fundamental es la librería OpenCV, una de las más utilizadas para programar aplicaciones gráficas en los sistemas embebidos ARM. OpenCV fue lanzada bajo una licencia BSD, por lo tanto es libre tanto en la parte académica como en la comercial. Tiene interfaces para los lenguajes de programación C++, C, Python y Java. OpenCV puede ser utilizada en Linux embebido. Sin embargo en Android embebido tiene algunas limitaciones que se aclararan en capítulos posteriores. OpenCV fue especialmente diseñada para incrementar la eficiencia computacional, con un fuerte enfoque en aplicaciones de tiempo real, un aspecto fundamental, teniendo en cuenta los recursos limitados de los sistemas ARM en comparación con un ordenador. Escrita en C/C++ optimizado, OpenCV puede tomar ventaja del procesamiento con múltiples núcleos. Habilitada con OpenCL, también puede tomar ventaja de la aceleración del hardware del dispositivo ARM donde se ejecute. Reconocida mundialmente, OpenCV tiene más de 47 mil miembros en la comunidad de usuarios y el número de descargas estimadas, exceden los 7 millones. Su uso va desde arte interactivo e inspección de minas, hasta robótica avanzada[1].

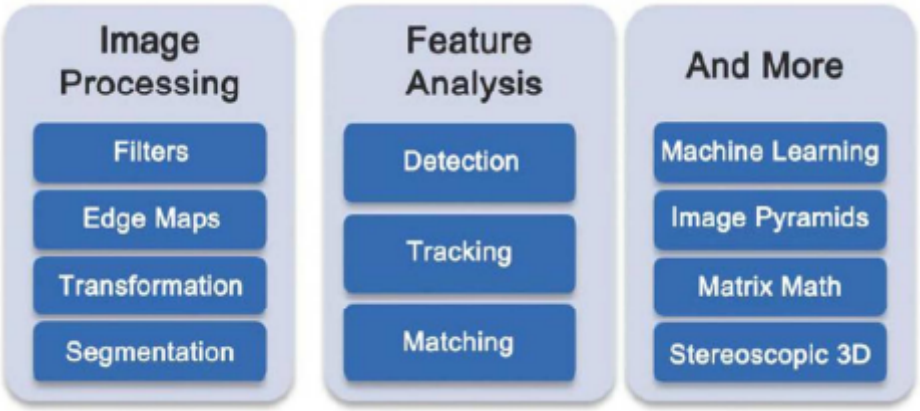

Figura 2.7: Características de la librería OpenCV.

En la Figura 2.7 se presentan las principales características de la librería OpenCV. Como se observa, tiene múltiples funciones para el área de visión artificial, procesamiento de imágenes y análisis de características.

[1]Opencv (open source computer vision). http://opencv.org/

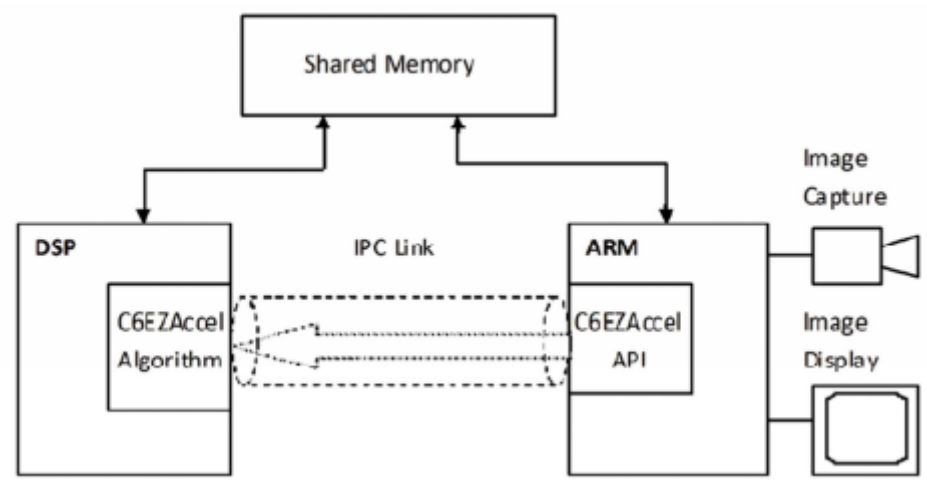

Figura 2.8: Aplicación DSP+ARM similar a la arquitectura de la plataforma BeagleBoard.

En la Figura 2.8 se presenta un sistema embebido, diseñado por Texas Instruments utilizando la librería OpenCV, para aplicaciones de captura y procesamiento de vídeo (Coombs and Prabhu, 2011). En conclusión se puede decir que la librería OpenCV ha sido ampliamente utilizada en aplicaciones de visión por computador y que está siendo implementada en dispositivos embebidos, con buenos resultados en plataformas de arquitectura DSP + ARM, permitiendo un alto desempeño, ya que la configuración de la memoria y la adquisición de vídeo se comparte entre el procesador DSP y el ARM. Estas plataformas permiten la integración del bajo consumo de energía con un framework de fácil interacción para el usuario.

En ocasiones las aplicaciones gráficas requieren de análisis numérico con matrices y vectores o el desarrollo de funciones matemáticas, en el caso del cálculo de áreas o para ubicar objetos 2D o 3D en coordenadas determinadas. Para ello existe la herramienta NumPy, una extensión de Python que le agrega soporte, constituyendo una librería de funciones matemáticas de alto nivel. Al utilizar NumPy, la funcionalidad de Python es comparable a la de Matlab ya que permite al usuario escribir programas de una alta eficiencia y velocidad, basados en operaciones con vectores y matrices. Otra herramienta relacionada con Python, es Pygame, que consiste en un conjunto de módulos diseñados para escribir juegos. Pygame permite crear juegos con todas las funciones y programas multimedia en el lenguaje Python. Pygame es altamente portable y funciona en casi todas las plataformas y sistemas operativos. Por último, es importante mencionar que el lenguaje Python puede utilizarse sin problemas en entornos embebidos basados en ARM.

Para las aplicaciones gráficas de RA, es posible utilizar la librería ARToolKit (Kato and Billinghurst, 1999) en los sistemas embebidos; en su versión para Linux. En los siguientes apartados se profundizará en este aspecto en particular. ARToolKit es una librería especializada en aplicaciones de RA. Utiliza el seguimiento de vídeo para calcular en tiempo real la posición de la cámara y la orientación relativa a la posición de los marcadores físicos. Al conocer la posición real de la cámara, se ubica una cámara virtual en

el mismo punto y se sobreponen modelos 3D sobre el marcador real. De esta manera se realiza el seguimiento del punto de vista y la interacción del objeto virtual.

Otra librería que posibilita el desarrollo de aplicaciones de RA embebidas es ARUco (Munoz-Salinas, 2012), librería desarrollada por un grupo de investigadores de la Universidad de Córdoba, España, basada en OpenCV. Algunas de sus características principales son; la detección de marcadores con una línea de código en C++, la detección de tableros para RA. Es posible integrarla con OpenGL y OGRE, es rápida y multiplataforma. Debido a que se basa en OpenCV, contiene diversos ejemplos que permiten ejecutar aplicaciones de RA en poco tiempo y tiene licencia BSD, por lo tanto es libre y modificable.

Las librerías de RA previamente mencionadas, se ejecutan únicamente bajo Linux, para Android embebido está disponible NyARToolKit(NyARTK, 2012), un SDK de código abierto para el desarrollo de aplicaciones de RA basadas en el reconocimiento de marcadores. Es un framework multiplataforma disponible para Android, Java, C#, AS3, C++ y Processing, que puede ser utilizado en sistemas embebidos basados en ARM. Utiliza marcadores del tipo ARToolKit, y dispone de soporte para diferentes formatos 3D (.mqo, .md2, .obj), mediante el uso de la librería min3D (Serrano, 2012). Las aplicaciones de RA para Android, se pueden crear, modificar y compilar en la interfaz de desarrollo Android Studio, integrada en eclipse, que cuenta con un emulador de dispositivos móviles y las diferentes versiones de Android.

Capítulo 3

Construcción y configuración del prototipo de pruebas

Tras un estudio inicial de las herramientas hardware y software disponibles para el desarrollo de aplicaciones gráficas embebidas y teniendo en cuenta la información presentada en el Capítulo anterior, se decide seleccionar el sistema embebido con mayores prestaciones, con el objetivo de construir un prototipo de pruebas.

Haciendo un balance entre funcionalidad, potencia de procesamiento, precio y la posibilidad del uso de herramientas software libres y genéricas, se opta por trabajar con una BeagleBoard xM.

En esta sección se describe el proceso para la construcción y configuración del prototipo de pruebas, utilizando la BeagleBoard xM, y se realiza una descripción detallada de sus características y configuración.

3.1. Hardware del Prototipo

Como se mencionó en la sección 2.1.5, la BeagleBoard xM es un sistema embebido de bajo coste, que tiene un rendimiento comparable al de un ordenador portátil. El procesador ARM Cortex - A8 que tiene integrado, es el corazón del sistema y por lo tanto el encargado de gestionar todas las tareas relacionadas con el hardware (periféricos). El prototipo construido se compone de varios elementos hardware, descritos en esta sección.

3.1.1. Dispositivo de visualización

Teniendo en cuenta que el principal objetivo el presente trabajo es ejecutar aplicaciones gráficas, el aspecto más importante para la construcción del prototipo, es el dispositivo de visualización.

La BeagleBoard xM cuenta con un conector para interfaz de vídeo digital DVI-D, uno de s-video y uno HDMI. Por lo tanto es posible conectar cualquier monitor o televisor corriente, o un LCD de pantalla plana y alta definición. Sin embargo existen otros accesorios que permiten personalizar el sistema de visualización, como la BeagleBoard xM Expansion V2, que es una pantalla LCD de 7 pulgadas, diseñada y construida por la

compañía ChipSee, para agregarle algunas funcionalidades a la BeagleBoard xM. Dicha pantalla es táctil, con resolución de 800*480 píxeles y tiene integrado un módulo WiFi, botones de navegación y un acelerómetro digital. En la Figura 3.1 se presenta una imagen de la pantalla y sus diferentes componentes.

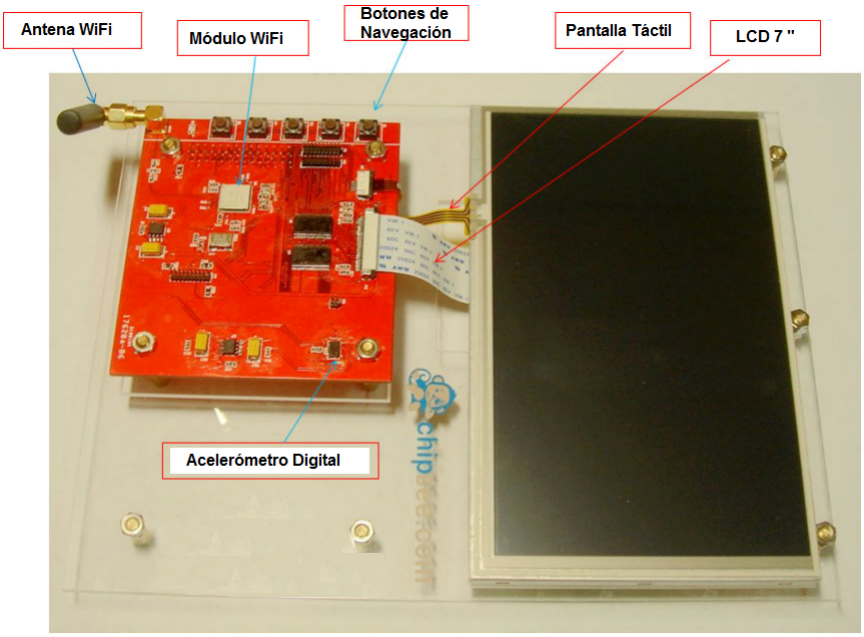

Figura 3.1: BeagleBoard xM Expansion.

Las características de esta pantalla, la hacen ideal para integrarla a la BeagleBoard xM, ya que permite interactuar con el sistema de manera táctil o con botones de navegación. Por otro lado, el acelerómetro es un valor agregado para la ejecución de videojuegos u otras aplicaciones gráficas que requieran el movimiento del dispositivo. Además, su diseño agrega portabilidad al sistema de visualización y permite su integración a juguetes, robots o controles industriales. Finalmente, la antena WiFi agrega propiedades de conectividad, con lo que se consigue un prototipo totalmente inalámbrico. En la Figura 3.2 se muestra la pantalla táctil integrada a la BeagleBoard xM (ChipSee Info and Tech Co., 2011).

Figura 3.2: Integración de la Pantalla Táctil con la BeagleBoard xM.

3.1.2. Tarjeta Micro SD

Para ejecutar cualquier sistema operativo en la BeagleBoard xM es necesario crear una imagen del software en una tarjeta micro SD e insertarla en el slot correspondiente. Se recomienda utilizar una tarjeta uSD Sandisk Clase 10 con mínimo 4 GB de espacio, otro tipo de marcas como Kingston pueden funcionar de manera incorrecta. Para el prototipo se utilizó una micro SD de 16 GB (Figura 3.3).

Figura 3.3: Tarjeta uSD Sandisk Clase 10 de 16 GB.

3.1.3. Fuente de Alimentación

La BeagleBoard se alimenta con 5 Voltios DC. Se puede utilizar un adaptador de 5V/2A como se hizo en el presente trabajo, o existe un módulo creado por la compañía Liquidware que contiene una batería de ion de litio que proporciona los 5V directamente a la BeagleBoard. Dicho dispositivo puede montarse en la parte trasera de la tarjeta, mediante separadores, manteniendo un fácil acceso a los puertos de expansión. La batería tiene un circuito de carga especializado, que aumenta el rendimiento de las aplicaciones, en especial las de mayor duración. El módulo suministra alimentación a través de un conector

de dos pines, igual al del adaptador, o a través de cables conectores macho duales. Para la recarga de la batería se puede utilizar un puerto mini USB y para la carga acelerada un puerto USB (Liquidware.Corp, 2014).

El uso de este módulo aumentaría la portabilidad del prototipo.

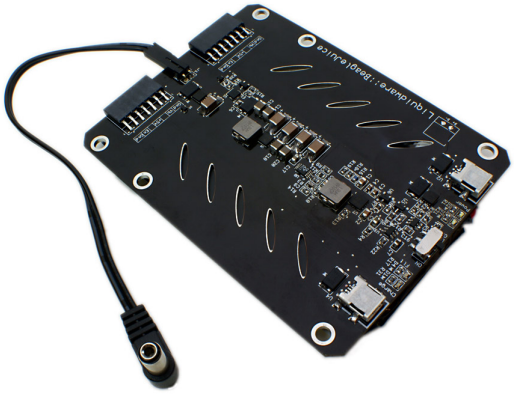

Figura 3.4: Batería Recargable para BeagleBoard de 5V.

3.1.4. Cámara

La selección de la cámara es un aspecto fundamental para las aplicaciones gráficas en tiempo real, por ejemplo las de visión por computador y RA. Como se observa en la Figura 2.5, la BeagleBoard xM tiene un conector para módulos de cámaras. La compañía Leopard Imaging Inc, tiene disponibles un gran número de módulos para ser conectados. Dichas cámaras se acoplan directamente al procesador digital de señales y de esta manera la velocidad de adquisición de vídeo se incrementa. En la Figura 3.5 se observa la cámara LI5M03 integrada a la BeagleBoard xM. Dicha cámara tiene una resolución de 5 Mega Píxeles, un formato de salida en RGB y soporte para 720p, 60fps, a un precio de 60 Euros.

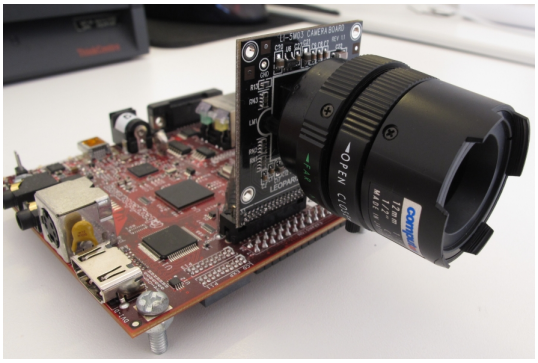

Figura 3.5: Cámara LI integrada a la BeagleBoard xM.

Otra opción, de menor coste, es utilizar una cámara web corriente. Para la construcción del prototipo de pruebas se utilizó la Webcam Logitech c170, que tiene una resolución en

la captura de vídeo de 1024 x 768 píxeles y tecnología Fluid Crystal™ que permite más fluidez, nitidez y detalles en las imágenes en movimiento. La cámara se conecta a través de un puerto USB, como se ilustra en la Figura 3.6.

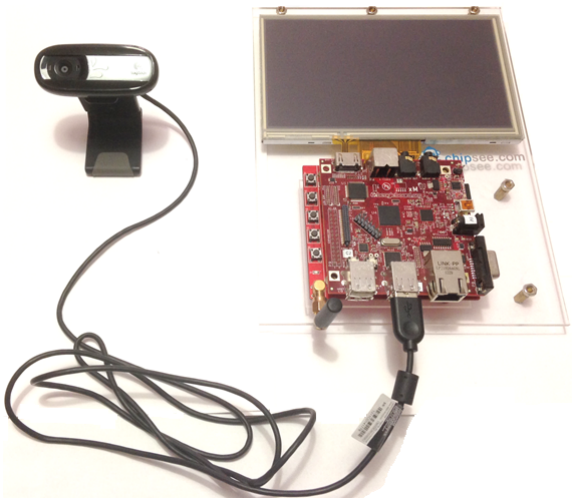

Figura 3.6: Cámara web Logitech y BeagleBoard xM.

3.1.5. Otros Periféricos

Otros periféricos, como el teclado y el ratón, se conectan a la BeagleBoard xM a través de puertos USB. Además se comunica con cualquier ordenador utilizando el puerto serial que tiene disponible. Por otro lado tiene la opción de añadir audio, con altavoces y micrófono. La conexión a internet puede realizarse a través de un cable Ethernet y finalmente tiene la posibilidad de comunicarse con cualquier dispositivo con entrada a un puerto USB 2.0 de alta velocidad o cable OTG.

3.2. Software del Prototipo

Como se mencionó en la sección 2.2.3, el cerebro de la BeagleBoard es el sistema operativo. También se observó que están disponibles diversos sistemas operativos embebidos. Entre dichos sistemas se seleccionaron Linux y Android para realizar pruebas en el prototipo, ya que son los más utilizados en el desarrollo de aplicaciones gráficas y además permiten el uso de múltiples herramientas de software libre.

3.2.1. Instalación y Configuración de Ubuntu

Para los sistemas basados en ARM, Linux ofrece varias distribuciones, en el caso del prototipo, se utilizó Ubuntu, debido a que es una de las distribuciones más estables y a su facilidad de uso.

Ubuntu ha sido portado extraoficialmente a múltiples arquitecturas. A partir de la versión 9.04 se empezó a ofrecer soporte extraoficial para los procesadores ARM.

Cannonical (Empresa creadora de Ubuntu) proporciona ayuda técnica y actualizaciones de seguridad por 18 meses a las versiones de corta duración y por 5 años a las versiones LTS o de larga duración. Por lo tanto las únicas versiones con soporte al día de hoy son: Ubuntu 12.04 LTS, Ubuntu 13.10 y la versión actual Ubuntu 14.04 LTS.

Para configurar el dispositivo de visualización BeagleBoard xM Expansion V2 (Ver Sección 3.1.1), es necesario contar con una versión de Ubuntu con el driver para pantalla táctil y otras características, configuradas desde el Kernel de Linux. La última versión disponible en Chipsee (Empresa creadora de la BeagleBoard xM Expansion V2) es la Ubuntu 11.04 (Ver Figura 3.1), una versión que como se indicó anteriormente, ya no tiene soporte de Cannonical. En un principio se intentó utilizar esta versión. Pero se presentaron dificultades enormes para la instalación de los paquetes ya que no se encontraban en los repositorios actuales y se debían buscar en repositorios antiguos. Hecho que generaba errores de dependencias y pérdida de tiempo en la instalación de los programas. Se optó entonces por utilizar la versión 12.04 LTS (Quantal) para ARM, la última versión disponible para la BeagleBoard xM al inicio del presente proyecto. Cabe mencionar que al día de hoy ya está disponible para su descarga la versión 14.04 LTS.

Figura 3.7: Ubuntu 11.04 portado en la BeagleBoard xM (Versión Táctil).

A continuación se describen detalladamente los pasos realizados para la instalación

de Ubuntu 12.04 LTS en la BeagleBoard xM. Antes de empezar, se debe contar con un ordenador o máquina virtual con Linux instalado, se recomienda la versión Ubuntu 10.04 o superior.

1. **Descarga del Software:** El primer paso consiste en buscar y descargar la versión requerida de Ubuntu, para ello, se utiliza el siguiente enlace:

   ```
   Link: http://rcn-ee.net/deb/rootfs
   ```

   Una vez encontrada la versión, se descarga desde la ventana de comandos el paquete .tar correspondiente.

   ```
   wget  http://rcn-ee.net/deb/rootfs/quantal/ubuntu-12.10-consolearmhf-
   2013-06-14.tar.xz
   ```

2. **Crear Disco de Arranque:** Como se menciona en la Sección 3.1.2, es necesario crear un disco de arranque para la instalación del sistema operativo. Se utilizó una micro SD con las características requeridas para crear la imagen de Ubuntu. Primero se descomprime el paquete .tar previamente descargado, en la carpeta de preferencia. Luego se localiza en dicha carpeta un script con el nombre **setup_sdcard.sh**, por lo general este fichero está disponible en todas las versiones de Ubuntu y es bastante útil para agilizar el proceso. Dicho script se encarga de realizar automáticamente las particiones necesarias para crear la imagen del sistema operativo. Antes de su ejecución, se identifica la unidad donde está alojada la micro SD.

   ```
   df -h
   ```

   En este caso la micro SD está localizada en /dev/sdb1. Sin embargo es importante utilizar el nombre /deb/sdb para incluirla ubicación completa de la unidad externa.

   ```
   Filesystem      Size  Used Avail Use% Mounted on
   /dev/sda1        12G  6.9G  3.8G  65% /
   udev            978M  4.0K  978M   1% /dev
   tmpfs           394M  800K  394M   1% /run
   none            5.0M     0  5.0M   0% /run/lock
   none            985M  208K  985M   1% /run/shm
   /dev/sdb1       3.7G  121M  3.4G   4% /media/beagleboardxm
   ```

   Figura 3.8: Unidades externas en Linux.

   Finalmente se ejecuta el script como se indica.

   ```
   sudo ./setup_sdcard.sh --mmc /dev/sdb --uboot beagle_xm
   ```

El proceso tarda entre 5 y 15 minutos, dependiendo de la velocidad de procesamiento del ordenador. Una vez completo el proceso, se genera una ventana de confirmación, que muestra que todo está correcto. Se retira la micro SD del ordenador y luego se inserta en la BeagleBoard xM.

Es importante tener en cuenta que la BeagleBoard xM solo se debe encender con la micro SD conectada, de lo contrario pueden generarse problemas de corto circuito.

3. **Configuración Inicial:** Luego de conectar la BeagleBoard xM a la fuente de alimentación, se entra al entorno de Ubuntu no gráfico, con la siguiente información:

```
Username:ubuntu
Password:temppwd
```

Dicha información viene por defecto en el SO. Lo primero que se debe hacer en el sistema, es ejecutar todas las actualizaciones disponibles.

```
sudo apt-get update
sudo apt-get upgrade
```

Luego de actualizar el sistema, se configura la conexión Ethernet.

```
sudo ifconfig -a
sudo dhclient eth0
sudo ifconfig -a
```

Se habilitan todos los periféricos, para el correcto funcionamiento del teclado, ratón, cámara web y pantalla LCD.

```
cd ~
cd boot/uboot
nano uEnv.txt
```

Se debe modificar el fichero uEnv.txtborrando la almohadilla # de la línea de código buddy=spidev.

```
#SPI: enable for userspace spi access on expansion header
#buddy=spidev
```

4. **Interfaz Gráfica:** Es necesario descargar e instalar una interfaz gráfica para facilitar la interacción del usuario con la BeagleBoard xM. Para ello se busca un GUI compatible con la versión de Ubuntu instalada.

```
sudo apt-get install gdm xfce4 network-manager
```

Se utiliza el siguient script para configurar la interfaz gráfica de manera automática.

```
/bin/bash /boot/uboot/tools/ubuntu/minimal_lxde_desktop.sh
```

Debido a que se tiene como dispositivo de salida una pantalla LCD de 7 pulgadas con una resolución de 800x480 píxeles, se deben asignar estos parámetros para una correcta visualización del GUI. Este proceso se realiza como se indica:

Listando las distintas resoluciones disponibles

```
xrandr -q
```

Estableciendo la resolución de la pantalla LCD

```
xrandr -s 800x480
```

Al ajustar la resolución se observa que los dpi difieren de la resolución de la pantalla. Hecho que afecta el diseño de las ventanas, su tamaño, entre otros aspectos. Por lo tanto es necesario modificar este factor de manera manual hasta lograr un buen ajuste:

```
xrandr -dpi 96 -s 800x480
```

Se debe seleccionar algún valor ente 96, 72 y 135.

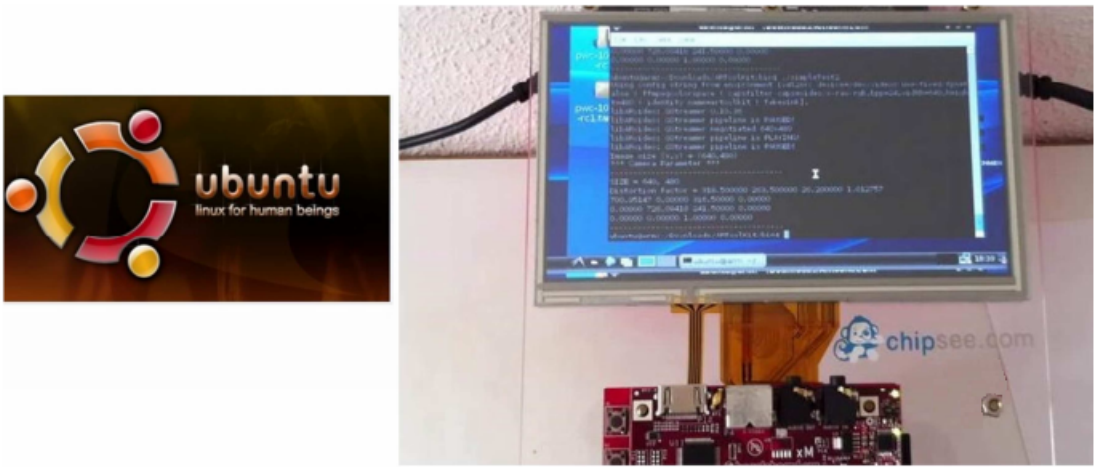

Figura 3.9: Ubuntu Quantal portado en la BeagleBoard xM.

En la Figura 3.9 se observa la interfaz gráfica utilizada y el entorno de la distribución de Ubuntu instalado en la plataforma BeagleBoard xM. La interfaz permite utilizar dos escritorios e interactuar con la placa de manera sencilla, mediante la barra de inicio típica, ventanas y carpetas.

El procedimiento descrito, se construyó con base en la información disponible en (Nelson, 2014) donde se profundiza en la personalización de Ubuntu Quantal para los procesadores ARM.

3.2.2. Instalación y Configuración de Android

Como se menciona en la Sección 2.2.3, Android es un sistema operativo basado en el kernel de Linux especialmente diseñado para dispositivos móviles con pantalla táctil. Es de código abierto y por lo tanto configurable y adaptable a requerimientos específicos. La comunidad de desarrolladores de Google, rowboat, se ha centrado en habilitar Android para los dispositivos de Texas Instruments, entre ellos la BeagleBoard xM. En el siguiente enlace se pueden encontrar una gran variedad de herramientas útiles para el desarrollo de proyectos con Android y la familia BeagleBoard:

`https://code.google.com/p/rowboat/downloads/list`

Sin embargo para la BeagleBoard xM Expansion V2 (Ver Sección 3.1.1) es necesario contar con una versión de Android modificada para las características del dispositivo, como la pantalla táctil, la antena WiFi, los botones de navegación y el acelerómetro digital. Para ello, Chipsee (Empresa creadora de la BeagleBoard xM Expansion V2) lanzó la versión modificada de Android GingerBread 2.3. En un principio se instaló esta versión. Pero su funcionamiento no era óptimo; contaba con pocas herramientas de configuración y la velocidad en la ejecución de tareas y aplicaciones era muy baja. Luego, a mediados del 2013, Chipsee lanzó la versión personalizada de Android Ice Cream Sandwich (ICS - 4.0.3), la última versión desarrollada hasta el día de hoy. Se decidió entonces por trabajar con esta versión.

A continuación se describen detalladamente los pasos realizados para la instalación de Android Ice Cream Sandwich (ICS - 4.0.3) en la BeagleBoard xM. Antes de empezar, se debe contar con un ordenador o máquina virtual con Linux instalado. Se recomienda especialmente trabajar con la versión Ubuntu 10.04 de 32 Bits, ya que se realizaron modificaciones en el kernel de Linux para Android y con otras versiones se presentaron inconvenientes para su compilación.

1. **Descarga del Software:** El primer paso consiste en descargar la versión de Android (ICS - 4.0.3) desarrollada por Chipsee, el paquete se localiza en el siguiente enlace:

   `http://www.chipsee.com/media/pdf_folder/ICS-Chipsee-Beagle-Release-2013-0407.rar`

   Dicho paquete contiene una imagen previamente compilada de Android, el código fuente del kernel y de Android, algunas aplicaciones de ejemplo (.apk) y el manual de instrucciones para su instalación.

2. **Configuración de la Cámara Web:** Aunque la versión de Android descargada es muy completa, pasa por alto algunos detalles de hardware. Uno de ellos y fundamental para el prototipo de pruebas, es la configuración de la Cámara Web.

   Como se indicó en la Sección 3.1.4, para el prototipo se utilizó una cámara web conectada por puerto USB a la BeagleBoard xM. En general, ninguna versión de Android tiene habilitada por defecto esta opción ya que los dispositivos móviles cuentan con cámaras integradas de otro tipo y no requieren el uso de cámaras externas.

   El proceso para habilitar la cámara web, se divide en dos partes:

   - Modificar y compilar nuevamente el código fuente del kernel y de Android.
   - Modificar las aplicaciones desarrolladas en Android para adquirir el vídeo a través del puerto USB. Este proceso en particular se explica detalladamente en el siguiente Capítulo.

3. **Modificar y Compilar el Kernel de Android:** El kernel es el núcleo del sistema operativo Android. Es el encargado de administrar todos los dispositivos hardware del sistema y sus funciones. Por lo tanto para agregar la cámara web es necesario hacer algunas modificaciones de hardware en el kernel.

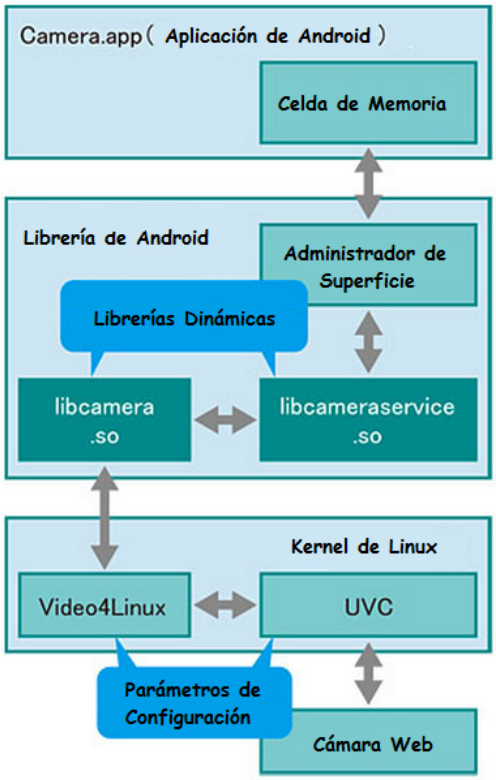

Figura 3.10: Esquema de la Cámara Web en Android.

Antes de empezar el procedimiento, es importante conocer el esquema de ejecución de una cámara conectada por puerto USB en Android.

Como se ilustra en la Figura 3.10, dicho esquema se compone de:

- La aplicación para la cámara (.apk), programada en Java para Android.
- Las librerías para la cámara de Android, entre las que se encuentran las librerías dinámicas libcamera.so y libcameraservice.so, que se encargan de actualizar el sistema de archivos raíz.
- El kernel de linux, donde se debe habilitar la API de captura de vídeo Video4Linux y el estándar para dispositivos USB, UVC (USB Video Class).

Para modificar el kernel se siguen unos pasos previos que se explican en el siguiente punto, luego se abre la carpeta del código fuente, descargado en el punto 1 y se ubica el paquete del kernel, se descomprime y se escribe la instrucción **menuconfig**, luego de ejecutar una instrucción para limpiar las variables.

```
> cd Source
> tar zxvf kernel-chipsee.tar.gz
> cd kernel-chipsee/
> make CROSS_COMPILE=arm-eabi- distclean
> make ARCH=arm menuconfig
```

Al ejecutar dicha instrucción, se abre la ventana de configuración del kernel. Una vez allí se habilita el vídeo (UVC), que se encuentra en los dispositivos V4L (Video4Linux), luego se guardan cambios y se cierra la ventana.

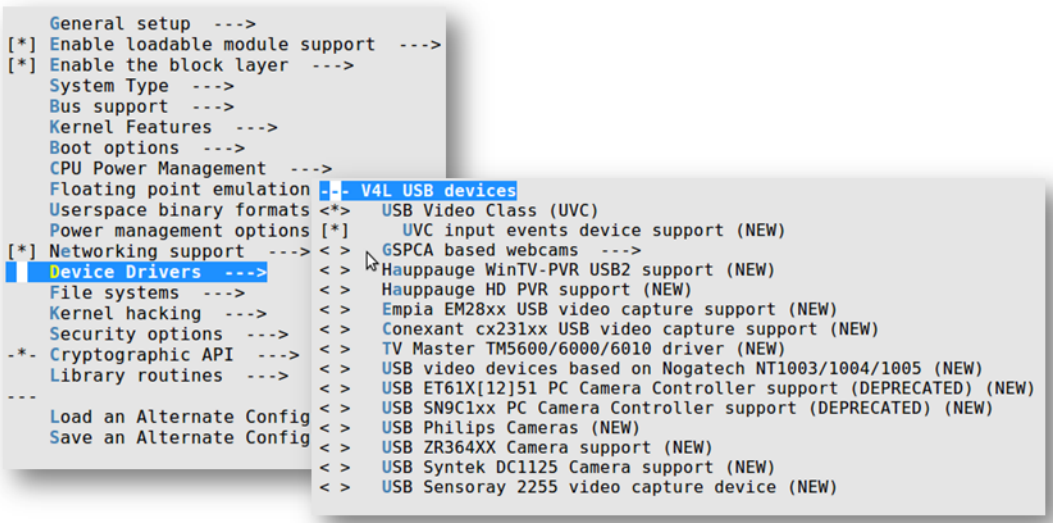

Figura 3.11: Configuración del Kernel.

Por último, se compila el kernel con la siguiente línea de código:

```
> make ARCH=arm CROSS_COMPILE=arm-eabi- uImage -j<N>
```

El valor de **N**, debe ser el doble del número de procesadores del ordenador, por ejemplo para una máquina de doble núcleo, como en este caso, se utilizó -j4.

Si la compilación culminó con éxito, al finalizar se genera el archivo uImage en la ruta; arch/arm/boot/.

4. **Modificar y Compilar Android:** Para el correcto funcionamiento de la cámara USB, también es necesario realizar algunas modificaciones en el código fuente de Android, dichas modificaciones y el proceso de compilación, se describe a continuación:

   - En primer lugar, se descarga el paquete de código fuente original desarrollado por Texas Instruments.

     ```
     http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android
     _DevKit/TI_Android_ICS_4_0_3_DevKit_3_0_0/exports/TI-Android-ICS
     -4.0.3_AM37x_3.0.0.bin
     ```

   - Se requiere la instalación de algunos paquetes para configurar el anfitrión de la compilación de Android.

     Instalar la versión sun-6 de Java

     Se escriben dos líneas de código en el fichero sources.list para descargar la versión de Java requerida, luego se descarga e instala.

     ```
     >sudo gedit /etc/apt/sources.list
     >deb http://ppa.launchpad.net/sun-java-community-team/sun->java6/
     ubuntu maverick main
     >deb-src http://ppa.launchpad.net/sun-java-community-team/>sun-
     java6/ubuntu maverick main
     >sudo add-apt-repository ppa:sun-java-community-team/sun-java6
     >sudo apt-get update
     >sudo apt-get install sun-java6-jdk
     ```

     Una vez instalado Java, es necesario seleccionar la versión a utilizar.

     ```
     sudo update-alternatives --config java
     ```

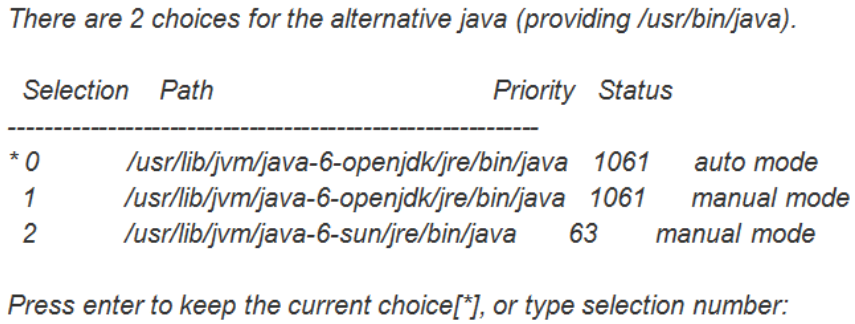

Figura 3.12: Selección de la Versión de Java.

La opción 2 es la versión adecuada para la compilación de Android.

Instalar los paquetes necesarios

```
>sudo add-apt-repository "deb http://archive.canonical.com/
lucid partner"
>sudo apt-get install git-core gnupg sun-java6-jdk flex bison
curl gperf libsdl-dev libesd0-dev libwxgtk2.6-dev
build-essential zip libncurses5-dev zlib1g-dev minicom
tftpd uboot-mkimage expect
```

Se extrae el paquete pre-compilado, se acepta la licencia y se instala.

```
>mkdir $HOME/rowboat-android
>cd $HOME/rowboat-android
>chmod a+x TI-Android-ICS-4.0.3_AM37x_3.0.0.bin
>./TI-Android-ICS-4.0.3_AM37x_3.0.0.bin
```

- Una vez instalado el paquete se ingresa a la carpeta **rowboat-android** y se buscan los siguientes ficheros para hacer las modificaciones respectivas:

BoardConfig.mk

```
# Habilita la cámara USB
20 #USE_CAMERA_STUB := true
21 BOARD_USB_CAMERA := true
```

device.mk

```
# Direcciona el fichero de permisos de escritura
20 device/ti/xxx/ueventd.xxx.rc:root/ueventd.xxx.rc \
79 #Camera
80 PRODUCT_PACKAGES += \
81         camera.omap3 \
82          Camera
```

init.rc

```
# Modifica el fichero de inicio del sistema
38 symlink /mnt/sdcard /sdcard
39
40 # mount sdcard third partition on /part-3
41   mkdir /part-3
42   mount vfat /dev/block/mmcblk0p3 /part-3/
```

ueventd.xxx.rc [1]

```
# Modifica el nodo del video USB, para la BeagleBoard es el 9
1 /dev/video9         0666  root       root
```

Finalmente se entra al directorio hardware/ti/xxx/camera y se modifican los ficheros de configuración de la cámara.

Android.mk

```
41 ifeq ($(BOARD_USB_CAMERA), true)
42 LOCAL_CFLAGS += -D_USE_USB_CAMERA_
43 endif
```

CameraHardware.cpp

```
81 #ifdef _USE_USB_CAMERA_
82 mCamera->Open(VIDEO_DEVICE_0);  /*USB camera use this node*/
83
84 #else
98 #endif
```

- Configurar el PATH para las herramientas arm-eabi- que están disponibles en el directorio prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin

```
>export PATH=$HOME/rowboat-android/prebuilt/linux-x86/toolchain
/arm-eabi-4.4.3/bin:$PATH
```

- Generar los archivos básicos del sistema para Android.

```
>cd $HOME/rowboat-android
> make TARGET_PRODUCT=omap3evm OMAPES=5.x -j<N>
# N es el número de procesadores del ordenador
```

[1] El nodo del vídeo USB para la BeagleBoard es 9 de acuerdo al manual de desarrollo de Texas Instruments para Android ICS, disponible en: http://processors.wiki.ti.com/index.php/TI-Android-ICS-PortingGuide

Este proceso generará los archivos del sistema básicos para Android en los directorios:

```
out/target/product/omap3evm/root/
out/target/product/omap3evm/system/
```

- Al modificar los ficheros del código fuente, se debe compilar nuevamente el sistema de arranque de Android (Bootloader).

  [**x-loader**]

```
> cd Source
> tar zxvf x-load-omap3.tar.gz
> cd x-loader-omap3/
> make CROSS_COMPILE=arm-eabi- distclean
> make ARCH=arm CROSS_COMPILE=arm-eabi- omap3beagle_config
> make ARCH=arm CROSS_COMPILE=arm-eabi-
> ./signGP ./x-load.bin
```

  Se genera el fichero MLO.

  [**u-boot**]

```
> cd Source
> tar zxvf u-boot-omap3.tar.gz
> cd u-boot-omap3/
> make CROSS_COMPILE=arm-eabi- distclean
> make ARCH=arm CROSS_COMPILE=arm-eabi- omap3_beagle_config
> make ARCH=arm CROSS_COMPILE=arm-eabi-
```

  Se genera el fichero u-boot.bin.

- Los siguientes ficheros deberán ser reemplazados para el correcto funcionamiento de todas las funciones de la BeagleBoard xM, como el G-sensor, teclas GPIO, entre otros.

```
/system/lib/hw/sensors.omap3evm.so   [Binary]
/system/usr/keylayout/Generic.kl     [ASCII]
/system/build.prop                   [ASCII]
/calibrate                           [ASCII]
/init.rc                             [ASCII]
/init.chipsee.sh                     [ASCII]
```

5. **Crear Disco de Arranque:** En este punto se describe el proceso de creación del disco de arranque de Android, utilizando los binarios generados en los pasos anteriores. Los parámetros de arranque predeterminados están ubicados en el script boot del directorio -/RowboatTools/mk-bootscr/boot.scr. Para modificar dichos parámetros se debe generar nuevamente el script boot.scr, como se indica:

```
cd ~/RowboatTools/mk-bootscr
```

Editar los parámetros y generar el script.

```
./mkbootscr
```

Los argumentos de arranque para la BeagleBoard xM son los siguientes:

```
setenv bootargs 'console=ttyO2,115200n8 androidboot.console=ttyO2
mem=256M root=/dev/mmcblk0p2 rw rootfstype=ext3 rootdelay=1
init=/init \ ip=off omap_vout.vid1_static_vrfb_alloc=y vram=8M
omapfb.vram=0:8M omapdss.def_disp=dvi omapfb.mode=dvi:1024x768-16'
```

Código fuente para la tarjeta micro SD

Se crea un directorio y se copian los binarios compilados con anteriormente:

```
>mkdir ~/image_folder
>cp $HOME/rowboat-android/kernel/arch/arm/boot/uImage ~/image_folder
>cp $HOME/rowboat-android/u-boot/u-boot.bin ~/image_folder
>cp $HOME/rowboat-android/x-loader/MLO ~/image_folder
>cp ~/RowboatTools/mk-bootscr/boot.scr ~/image_folder
>cp $HOME/rowboat-android/out/target/product/omap3evm/rootfs.tar.bz2
~/image_folder
>cp ~/RowboatTools/mk-mmc/mkmmc-android.sh ~/image_folder
```

Preparar la tarjeta Micro SD

Se requiere un formato específico en la tarjeta micro SD, para que funcione correctamente con al archivo boot de Android. Calcular el tamaño de la tarjeta con el siguiente procedimiento:

```
sudo fdisk -l  /dev/<nombre_de_la_tarjeta>
```

Se obtiene la siguiente información.

```
Disk /dev/mmcblk0: 1990 MB, 1990197248 bytes
```

Se divide el tamaño de la tarjeta en bytes entre 255 cabeceras, 63 sectores y 512 bytes y finalmente se redondea el número obtenido al entero más cercano, de esta manera calculará el número de cilindros de la tarjeta:

```
1990197248/255/63/512 = 241.96 = 241 cylinders
```

241 cilindros es un tamaño adecuado para las particiones que requiere Android.

Crear la imagen de Android:

Se conecta la tarjeta micro SD al ordenador con GNU/Linux, y se entra al directorio creado en el apartado anterior.

En el directorio se encuentra el script **mkmmc-android.sh**. Dicho script se encarga de hacer las particiones necesarias y de crear el disco de arranque. Luego se debe localizar la unidad donde se encuentra la tarjeta micro SD, como se indica en la en Sección 2, en el apartado **Crear Disco de Arranque**, y seguir el mismo procedimiento. Sin embargo, previo a ello, es necesario cambiar la línea 27 del script **mkmmc-android.sh**, pues tiene un error de configuración:

```
SIZE=‘fdisk -l $DRIVE | grep $DRIVE | awk ’{print $5}’‘
```

6. **Iniciar Android en la BeagleBoard xM:** Cuando finaliza el proceso anterior, se inserta la micro SD en la BeagleBoard xM y se enciende. Una vez ha cargado Android, en los ajustes, se configura para que permanezca encendida.

Figura 3.13: Android ICS 4.0.3 portado en la BeagleBoard xM.

Los botones de usuario están definidos como; Menu, Principal, Volumen +, Volumen - y Regreso. El driver WiFi no tiene soporte para el protocolo WEP, únicamente soporta WPA/WPA2 PSK. En la Figura 3.13 se observa el sistema operativo Android ICS 4.0.3 portado en la BeagleBoard xM.

Capítulo 4

Pruebas y Desarrollos Realizados

Al finalizar la configuración de cada sistema operativo, se decidió hacer una selección de herramientas software libres, para el desarrollo de diferentes aplicaciones gráficas de prueba. Para esta elección se tuvo en cuenta la revisión bibliográfica realizada en el Capítulo 2. Se observó que en el área de RA hay muy pocos desarrollos para sistemas embebidos ARM y que la librería más utilizada de todas las publicaciones investigadas es OpenCV. Además de estar especialmente diseñada para arquitecturas DSP+ARM como se menciona en la Sección 2.4. Por lo tanto las pruebas y desarrollos realizados, se basaron en diferentes librerías de RA y en OpenCV.

En esta sección se detallan las pruebas ejecutadas con cada sistema operativo, describiendo los alcances y limitaciones de cada uno.

4.1. Aplicaciones gráficas en Ubuntu Embebido

Luego de hacer una búsqueda de herramientas libres para el desarrollo de aplicaciones gráficas en Linux, y teniendo en cuenta las utilizadas en trabajos previos, se seleccionaron las presentadas en los siguientes apartados, para Ubuntu.

4.1.1. ARToolKit

Se utilizó la última versión disponible de la librería ARToolKit (Ver Sección 2.4), para la distribución de Ubuntu (Quantal). Cabe mencionar, que las últimas versiones son multiplataforma.

La funcionalidad de cada versión es la misma. Pero su desempeño puede variar dependiendo de la configuración del hardware. En el caso de la BeagleBoard, depende de la velocidad de procesamiento del sistema y de la cámara.

Instalación y Configuración

Las herramientas requeridas para la instalación de ARToolKit son: OpenGL, GLUT y las librerías para aplicaciones audiovisuales. En este caso se trabajó con GStreamer.

Se instalan entonces, las dependencias requeridas.

```
> sudo apt-get install freeglut3-dev libgstreamer0.10-dev
libgstreamer-plugins-base0.10-dev libxi-dev libxmu-headers
libxmu-dev libjpeg62-dev libglib2.0-dev libgtk2.0-dev
```

Se extrae y se compila ARToolKit.

```
> wget "http://sourceforge.net/projects/artoolkit/files/artoolkit/
2.72.1/ARToolKit-2.72.1.tgz/download" -O ARToolKit-2.72.1.tgz
> tar xzvpf ARToolKit-2.72.1.tgz
> cd ARToolKit
> ./Configure
> make
```

A continuación se configura la herramienta para adquirir el vídeo a través de una cámara USB. La opción 5 (Framework GStreamer).

```
"1: Video4Linux" para capturadoras de video (mediante V4L
versión 1).
"2: Video4Linux+JPEG Decompression (EyeToy)" para cámaras Play
Station EyeToy (mediante V4L versión 1).
"3: Digital Video Camcoder throught IEEE 1394 (DV Format)" para
cámaras firewire.
"4: Digital Video Camera throught IEEE 1394 (VGA NONCOMPRESSED
Image Format)" para cámaras firewire.
"5: GStreamer Media Framework" para webcams USB.
```

Para crear los Makefiles[1] de la instalación, es necesario crear un fichero de configuración para el pkg-config[2]. De esta manera se definen las rutas de las nuevas librerías y sus ficheros de cabecera. Una vez realizado y guardado el fichero con los parámetros correspondientes, se crean los Makefiles.

```
> pkg-config --cflags AR
-I/usr/local/include/AR
> pkg-config --libs AR
-L/usr/local/lib -lARgsub -lARgsub_lite -lARgsubUtil -lARMulti -
lARvideo -lAR
```

Es importante configurar la ruta principal, en el fichero **/ .bashrc**.

```
> export LD_LIBRARY_PATH=/usr/local/lib
```

[1]Fichero de texto que se utiliza en Linux para llevar la gestión de la compilación de programas.

[2]Programa que provee una interfaz unificada para llamar a bibliotecas instaladas, cuando se está compilando un programa a partir del código fuente.

Se crea una variable para configurar el nodo de la captura del vídeo y su formato. De acuerdo a la resolución de la pantalla utilizada (Ver Figura 2.7), se configuran los parámetros de altura y ancho (height - width). El nodo del vídeo en este caso es el 0 (/dev/video0).

```
> export ARTOOLKIT_CONFIG="v4l2src device=/dev/video0
use-fixedfps=false ! ffmpegcolorspace ! capsfilter
caps=video/x-rawrgb,bpp=24,width=640,height=480 ! identity
name=artoolkit ! fakesink"
```

Se realizaron las pruebas correspondientes y se confirmó que la cámara utilizada (ver Figura 3.6) soporta el módulo gráfico de ARToolKit con OpenGL, usando el siguiente comando:

```
> gst-launch-0.10 v4l2src ! xvimagesink
```

Ejecución de Ejemplos

Una vez configurados y verificados todos los parámetros de la cámara y la librería, se ejecutaron diferentes ejemplos disponibles en ARToolKit, con el fin de verificar su funcionamiento y rendimiento en la BeagleBoard.

Figura 4.1: Test de gráficos de ARToolKit en la BeagleBoard xM.

- **GraphicsTestd:** El ejemplo realiza la prueba de los gráficos con un objeto simple. En la Figura 4.1 se observa una tetera girando a una tasa de aproximadamente 10 fps.

- **SimpleTest:** Para validar algunos de los ejemplos disponibles en la librería, se utilizó el marcador **Hiro**[3], presentado en la Figura 4.2. En la Figura 4.3 se observa la ejecución de SimpleTest, con un objeto básico de 6 caras (cubo). Para este caso los fps disminuyen considerablemente a 0.5.

Figura 4.2: Marcador Hiro, para validar la aplicación de RA.

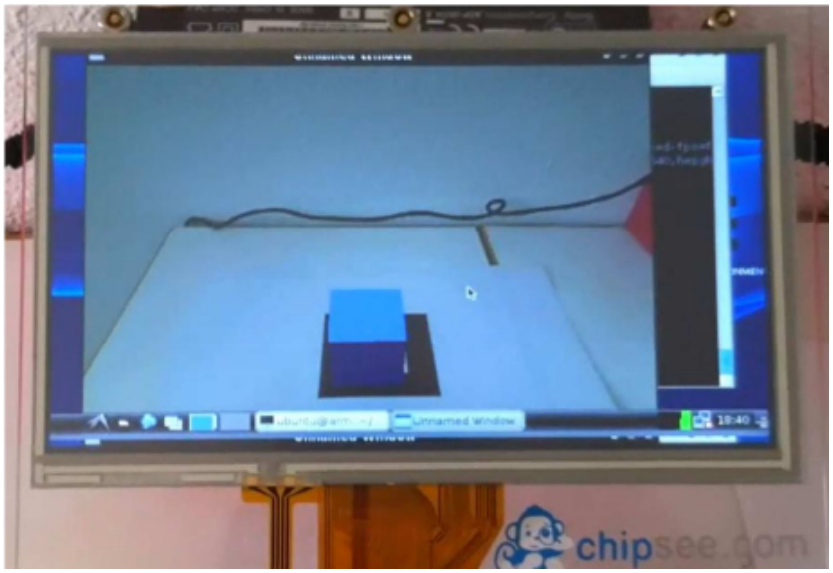

Figura 4.3: ARToolKit portado en la BeagleBoard xM.

- **SimpleLite:** El ejemplo anterior se ejecutó con una resolución de 640 x 480 píxeles. Se decide bajar la resolución a 320 x 240 y se obtiene una tasa de 4 fps. Para este caso se ejecuta el ejemplo **simpleLite**, como se observa a continuación.

[3]Hiro es el marcador básico, utilizado en aplicaciones de RA. Al reconocer este marcador, el sistema genera el render de un objeto.

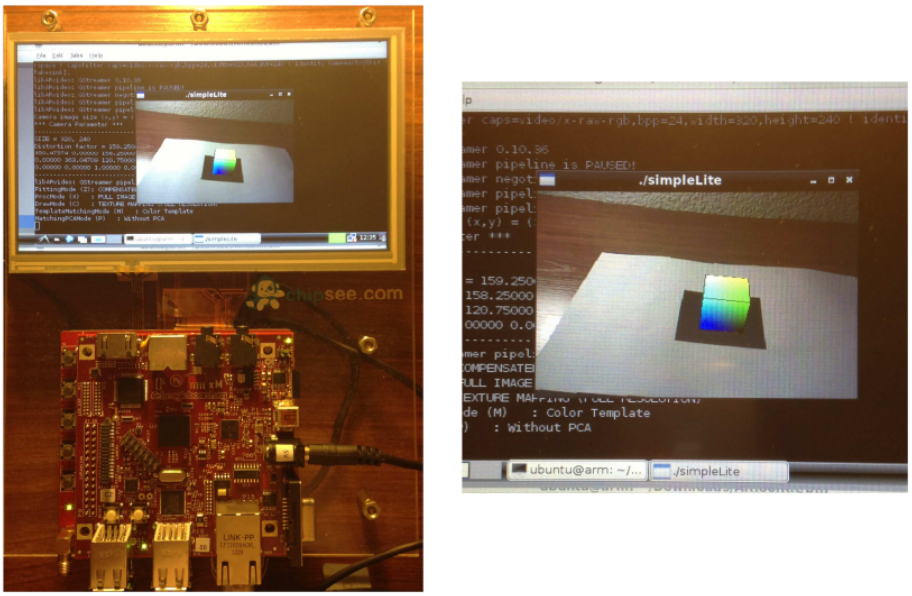

Figura 4.4: Cubo de Colores con Marcador **Hiro**, en BeagleBoard xM.

- **Exviewd:** Utilizando el ejemplo **exviewd** se calcula y se muestra la posición de la cámara con respecto al marcador **Hiro**. Las coordenadas de la cámara se muestran en color rojo y en la esquina inferior izquierda se observa el objeto renderizado ubicado en el plano virtual.

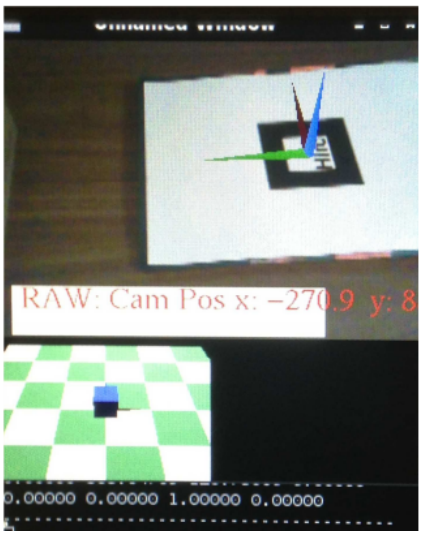

Figura 4.5: Cálculo de las coordenadas de la cámara.

- **Varios ejemplos y marcadores:** Se ejecutan varios ejemplos con los marcadores **Hiro** e **Hito**[1]. En la Figura 4.6 superior izquierda se observa una esfera dibujada sobre el marcador **Hiro**, en la figura superior derecha se dibuja una esfera sobre **Hiro** y un cono sobre **Hito**. Se hace uso de diferentes marcadores para comprobar que

el reconocimiento se hace de manera correcta, ejecutando **paddleTestd** se dibuja una raqueta de paddle sobre **Hiro**, luego al ejecutar **modeTestd** se dibuja un cubo sobre **Hiro**, un toroide sobre **Hito**, un cono sobre **samppatt1**[1] y una esfera sobre **samppat2**[1].

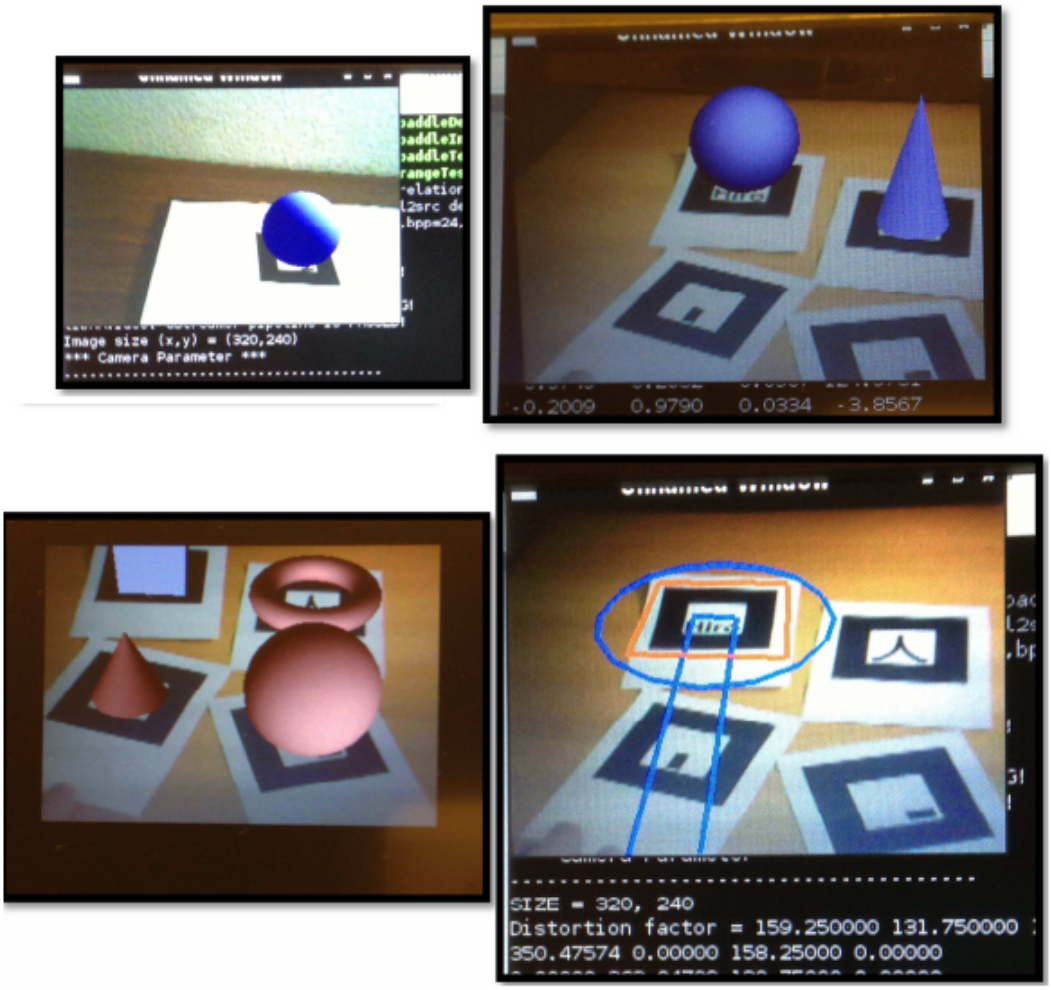

Figura 4.6: Diferentes objetos y marcadores en la BeagleBoard xM.

- **RangeTestd:** Finalmente se ejecuta **rangeTestd**, donde existe interacción entre la cámara y los marcadores. Dibuja una tetera sobre la plantilla **Hiro** y a medida que se acerca el marcador a la cámara, el objeto se hace más pequeño.

[1]Marcadores básicos para RA

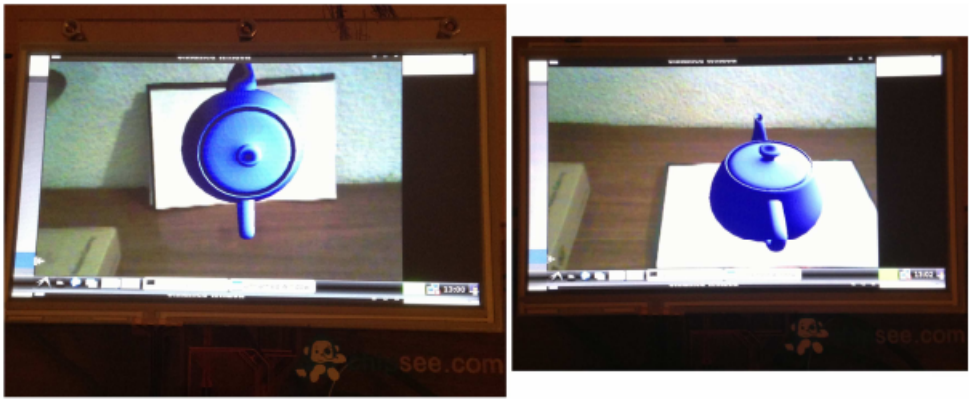

Figura 4.7: Tetera con diferentes tamaños.

Una vez realizadas las diferentes pruebas, se observa que el renderizado de los objetos se ejecuta correctamente en la BeagleBoard xM. Sin embargo los fps no superan el valor de 10 y se consumen todos los recursos de la placa. Este hecho puede ser consecuencia de la capacidad de la memoria RAM y la velocidad del procesador ARM, además que el ARToolKit no tiene optimizada la adquisición de vídeo. Por esta razón se decide realizar pruebas con una librería diferente donde se logre optimizar los recursos en la adquisición y el procesamiento de la imagen.

4.1.2. OpenCV

De acuerdo a lo expuesto en la Sección 2.4, se considera una buena opción utilizar la librería OpenCV para implementar aplicaciones de RA en la BeagleBoard xM.

Instalación y Configuración

Se instaló la última versión de OpenCV con soporte para OpenGL, soporte de lectura y escritura de vídeos, acceso a la cámara Web, interfaces en Python, C y C++.

Para empezar la instalación, es necesario obtener primero todas las dependencias requeridas:

```
> sudo apt-get install build-essential libgtk2.0-dev libjpeg-dev
libtiff4-dev libjasper-dev libopenexr-dev cmake python-dev
python-numpy python-tk libtbb-dev libeigen2-dev yasm libfaac-dev
libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev
libvorbis-dev libxvidcore-dev libx264-dev libqt4-dev
libqt4-opengl-dev sphinx-common texlive-latex-extra libv4l-dev
libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev
```

Se obtiene el código fuente del enlace:

```
> cd ~
> wget http://downloads.sourceforge.net/project/opencvlibrary/opencvunix
```

```
/2.4.2/OpenCV-2.4.2.tar.bz2
> tar -xvf OpenCV-2.4.2.tar.bz2
> cd OpenCV-2.4.2
```

Luego, se genera el Makefile usando la instrucción cmake. En este punto se define que parte de OpenCV se requiere compilar. Si se utilizará Python, TBB, OpenGl, una cámara Web, entre otros. En este caso se utilizan todas las herramientas para el desarrollo de las aplicaciones gráficas.

```
> mkdir build
> cd build
> cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON-D
WITH_V4L=ON -D INSTALL_C_EXAMPLES=ON -D
INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON -D WITH_QT=ON -D
WITH_OPENGL=ON ..
> make
> sudo make install
```

Se debe editar el archivo de configuración de OpenCV, escribiendo la ruta correspondiente (/usr/local/lib).

Es necesario de igual manera configurar la variable PATH con la ruta correspondiente a la librería OpenCV. Finalmente se debe comprobar que no se produce ningún error en la instalación y que en particular se reporte FFMPEG como activo o de lo contrario no será posible leer o escribir vídeos. Python, TBB, OpenGL, V4L, OpenGL y Qt deben ser detectados.

Una vez se ha hecho lo anterior, se comprueba con un ejemplo básico de detección de rostro en la fotografía de lena.jpg incluida en los ejemplos, que las librerías de OpenCV funcionan correctamente en la BeagleBoard xM.

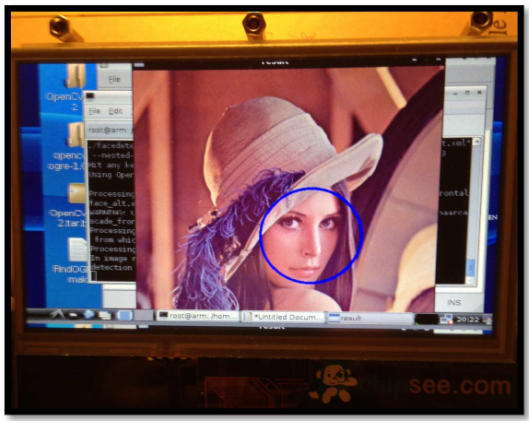

Figura 4.8: Reconocimiento de Rostros con OpenCV en la BeagleBoard xM.

Para configurar la cámara Web y la entrada de vídeo, se desarrolla un script en Code::Blocks que permite activar la cámara con las herramientas de OpenCV. Se comprueba que la adquisición de vídeo funciona correctamente y con buena velocidad.

4.1.3. ArUCO

Se realizaron pruebas con ArUCO debido a que está basada en openCV y por lo tanto la adquisición de vídeo está optimizada.

Para profundizar en la descripción de librería, consultar la Sección 2.4.

Instalación y Configuración

Al igual que en apartados previos, es necesario obtener algunas dependencias y paquetes antes de realizar la instalación de ArUCO, para Ubuntu 12.04 se requiere lo siguiente:

```
> sudo apt-get libicu-dev freeglut3 freeglut3-dev libgstreamer0.10-dev
libgstreamer-plugins-base0.10-dev libxine-dev
```

Las librerías de ArUco se descargan en:

```
> wget http://sourceforge.net/projects/aruco/files/1.2.4/aruco-1.2.4.tgz
```

Se compilan e instalan las librerías. Se comprueba que la instalación ha sido correcta ejecutando un ejemplo para crear una tabla de marcadores:

```
> ./aruco_create_board 5:2 board.png board.yml
```

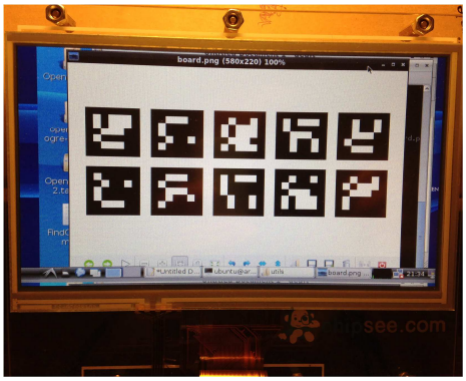

Figura 4.9: Marcadores Generados en ArUCO.

Calibración de la Cámara

En ArUCO es necesario calibrar la cámara para obtener parámetros específicos y así generar objetos renderizados de acuerdo a los mismos. En la calibración de la cámara, se obtienen los valores que permiten determinar el punto 3D que se proyecta espacialmente en el sensor de la cámara. Existen parámetros intrínsecos y extrínsecos. OpenCV permite obtener los parámetros intrínsecos de la cámara. Solo se deben presentar frente a la misma, varias imágenes de un tablero de ajedrez con dimensiones conocidas como el que se muestra en la Figura 4.10.

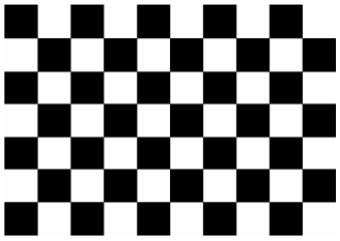

Figura 4.10: Tablero para calibrar la cámara en ArUCO.

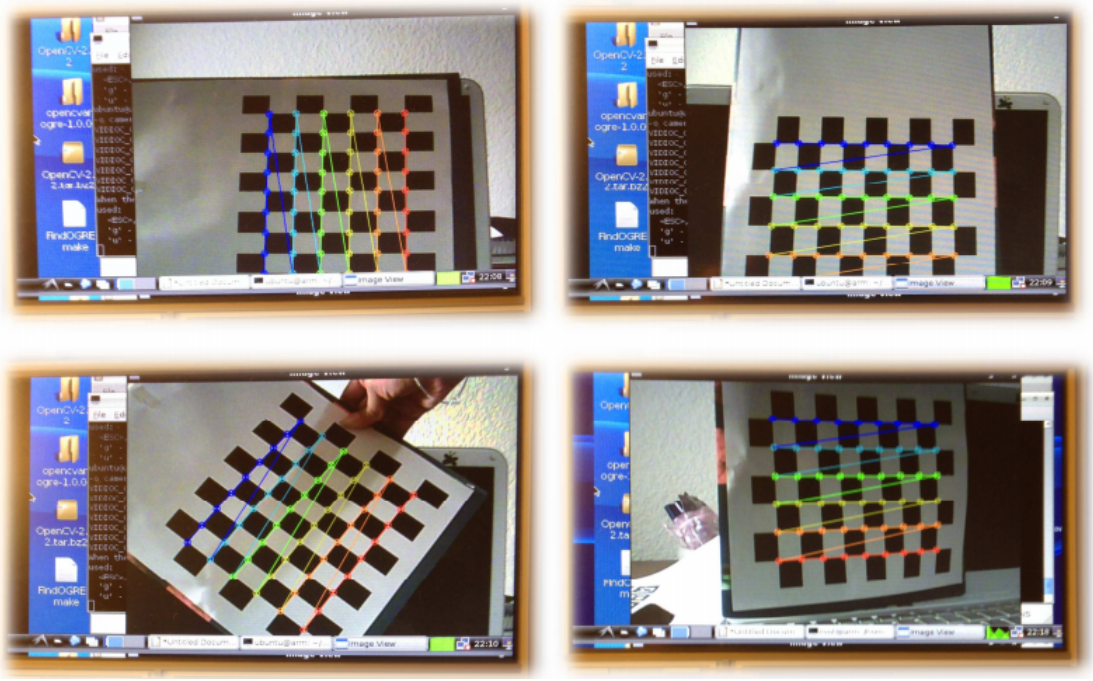

Figura 4.11: Calibración de la cámara en ArUCO.

La cámara se calibra de acuerdo a los siguientes parámetros:

```
> ./calibration 0 -w 9 -h 6 -s 0.025 -o camera.yml -op ?oe
```

Donde, 0 es el parámetro de la cámara, -w es el ancho del tablero, -h la altura, -s el tamaño en metros de los cuadrados y **camera.yml**, el fichero de salida para la ejecución de las aplicaciones. ArUco permite trabajar con más de 1024 marcadores, en la Figura 4.12 se presenta una prueba realizada con uno de ellos. En la prueba se reconoce el marcador, el sistema sobrepone los ejes de coordenadas y un cubo sin aristas sobre el área del marcador, lo que quiere decir que la cámara se calibró correctamente.

```
> cd aruco-1.2.4/
> cd build/utils
> ./aruco_test live camera.yml 0.025
```

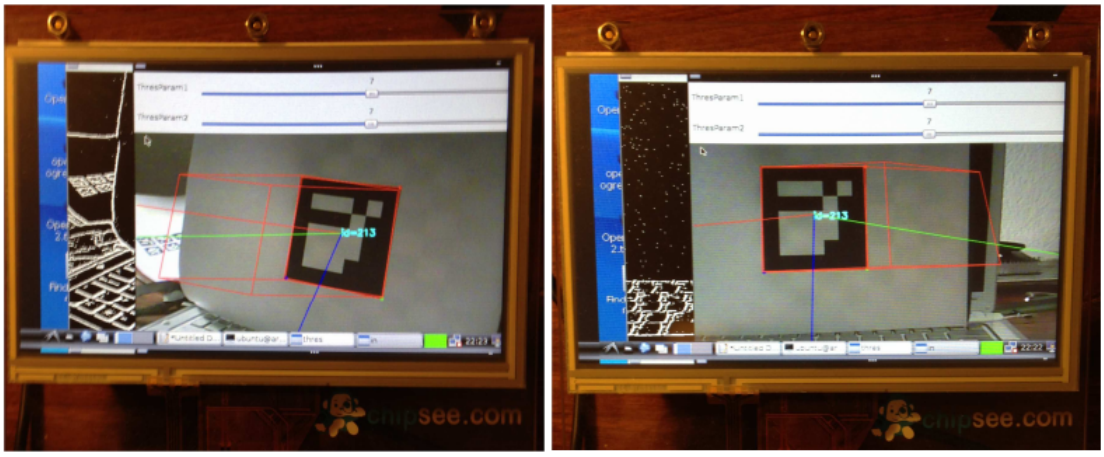

Figura 4.12: Detección de un marcador con ArUco y OpenCV.

La siguiente prueba se realizó con el tablero de marcadores completo.

```
> ./aruco_test_board live board.yml camera.yml 0.025
```

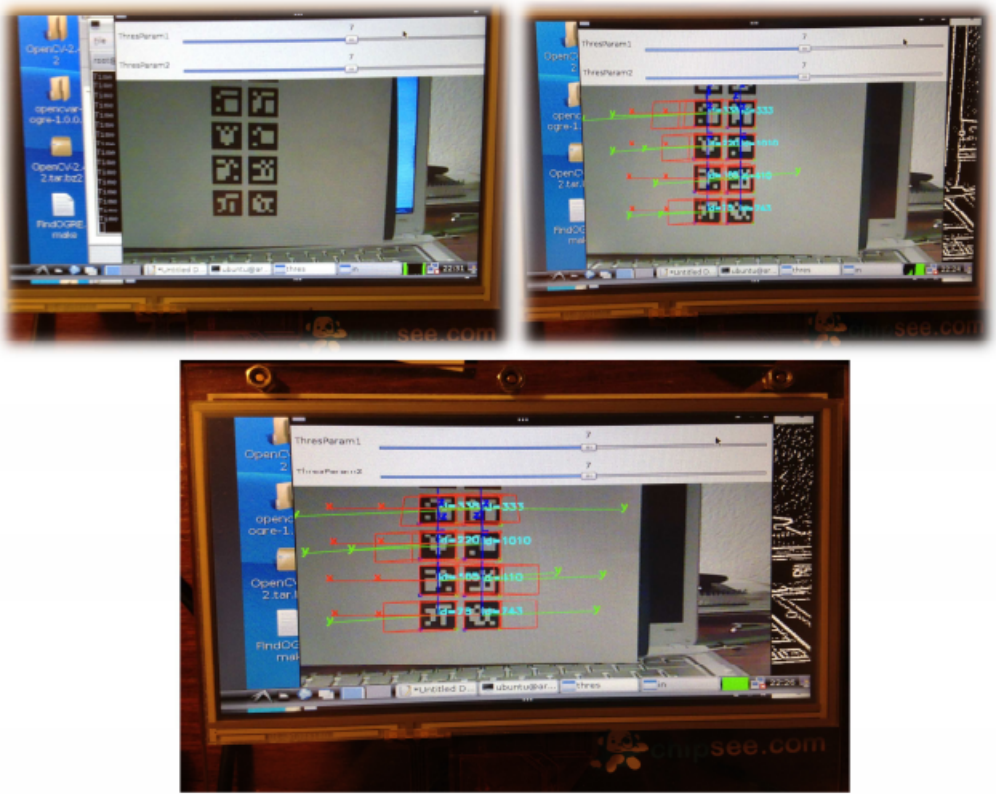

Figura 4.13: Diferentes marcadores en ArUco con OpenCV.

Se comprueba que ArUCO tiene mejor rendimiento que las librerías anteriores, ya que tiene una tasa de 15 fps.

Por otro lado, para generar objetos render es necesario integrar ArUco con librerías de renderizado como OpenGL y OGRE.

Renderizado de objetos con OGRE

La librería de ArUco incluye un ejemplo de OGRE para ser ejecutado con diferentes marcadores. Se debe instalar una versión específica de Ogre3D para ArUco. El proyecto ha sido probado únicamente en Linux, por los desarrolladores, por lo tanto es posible que no funcione en un sistema operativo diferente.

Se procede entonces a instalar OGRE.

```
> sudo add-apt-repository ppa:ogre-team/ogre
> sudo apt-get update
> sudo apt-get install libogre-dev
> sudo apt-get install ogre-samples-media ogre-samples-bin
```

Al realizar el procedimiento anterior, se observa que existen múltiples errores en los repositorios, están desactualizados y no permiten la descarga de los ejemplos y objetos render. Esta versión de OGRE para ArUCO, está disponible y funciona únicamente para Ubuntu 10, y como ya se explicó al inicio del Capítulo, dicha versión de Ubuntu no tiene soporte de Cannonical y por lo tanto no es conveniente instalarla en la BeagleBoard xM.

En la aplicación, al reconocer el tablero de marcadores, el sistema generaría el render animado que se observa en la siguiente imagen.

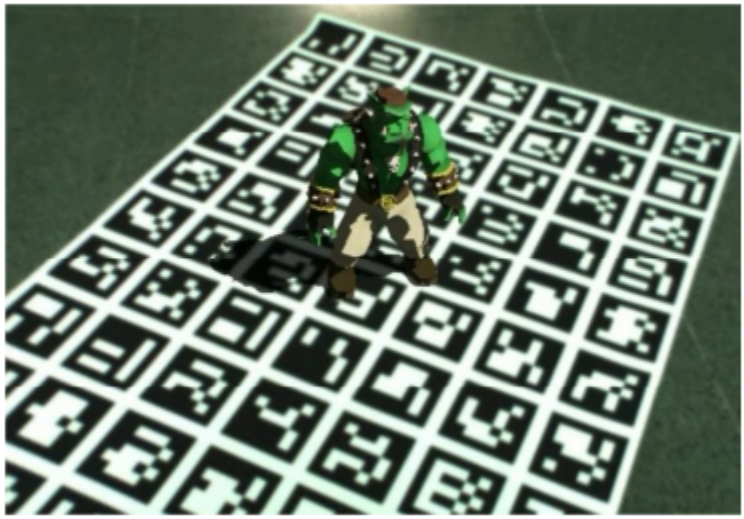

Figura 4.14: RA con ArUco y OGRE.

4.1.4. Pygame - Numpy

De las pruebas anteriores se concluye que las aplicaciones de RA requieren por lo general del uso de librerías render para generar objetos 3D. Dichos objetos consumen la mayor parte de los recursos de la tarjeta. Por lo tanto se decide trabajar con una aplicación que no requiera del uso de objetos render, con el fin de observar la velocidad de procesamiento y rendimiento de la BeagleBoard xM.

Como se mencionó en la Sección 2.4, la librería Pygame se compone de un conjunto de módulos de Python y está diseñada para escribir juegos básicos. Dicha librería permite desarrollar aplicaciones gráficas sencillas, de RA o de reconocimiento de patrones. Su instalación es simple y se realizó en el apartado de OpenCV (Ver 4.1.2), dentro de la instalación de los demás paquetes.

Características de la Aplicación

Está disponible para la descarga gratuita, una aplicación de RA, desarrollada con Python, Pygame y Numpy. El funcionamiento de dicha aplicación consiste en entrenar como marcador[1] un objeto real de forma simple, por medio del reconocimiento de colores. Una vez entrenado, el marcador puede interactuar con objetos virtuales 2D, como pelotas, y a su vez activa o desactiva botones virtuales que ejecutan acciones determinadas.

Funcionamiento

El programa está contenido en un solo script. Es necesario guardar las librerías requeridas en la misma ubicación de dicho script. Para ejecutar la aplicación simplemente se escribe la siguiente instrucción en la ruta correspondiente:

```
> Python Aplicación.py
```

Al ejecutar la aplicación, se abre una interfaz que muestra el área donde se debe ubicar el objeto a calibrar y que actuará como marcador. Dicha área está representada por un recuadro rojo, por otro lado, en la esquina superior izquierda se ubica un cuadro con el color del objeto contenido en el área de calibración. El recuadro rojo se puede adaptar a la forma del objeto, aumentando de tamaño con las teclas +/- o desplazándose de izquierda a derecha y de arriba hacia abajo con las flechas del teclado. En la Figura 4.15 se observa la zona de calibración.

Para empezar, se debe ubicar un objeto de forma simple frente a la cámara. Con el teclado se ajustan las 3 componentes RBG del color, aumentando o disminuyendo los niveles de las mismas hasta que el objeto se distinga claramente del fondo; como se ilustra en la Figura 4.16, para varios objetos circulares de color rojo, azul y amarillo.

Una vez se ha calibrado el color correctamente la aplicación reconocerá el objeto como un marcador.

[1] Símbolos o imágenes donde se superpone información aumentada. En este caso se denomina marcador al objeto a calibrar.

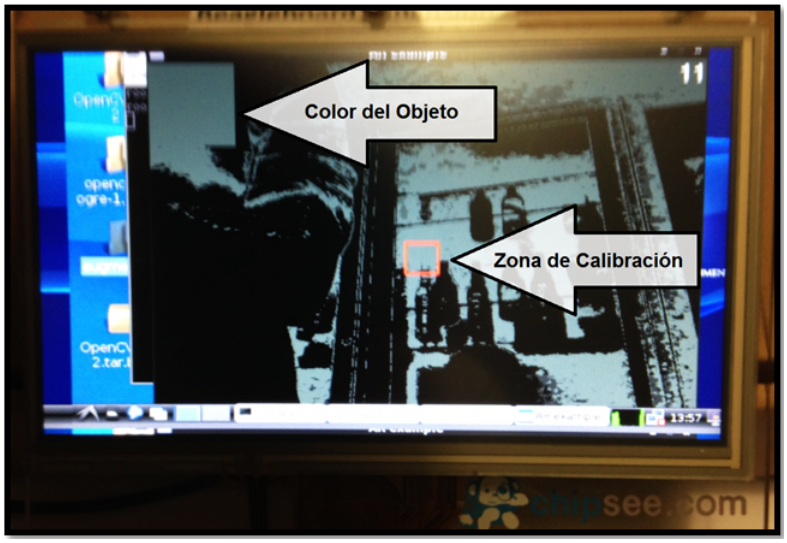

Figura 4.15: Zona de Calibración.

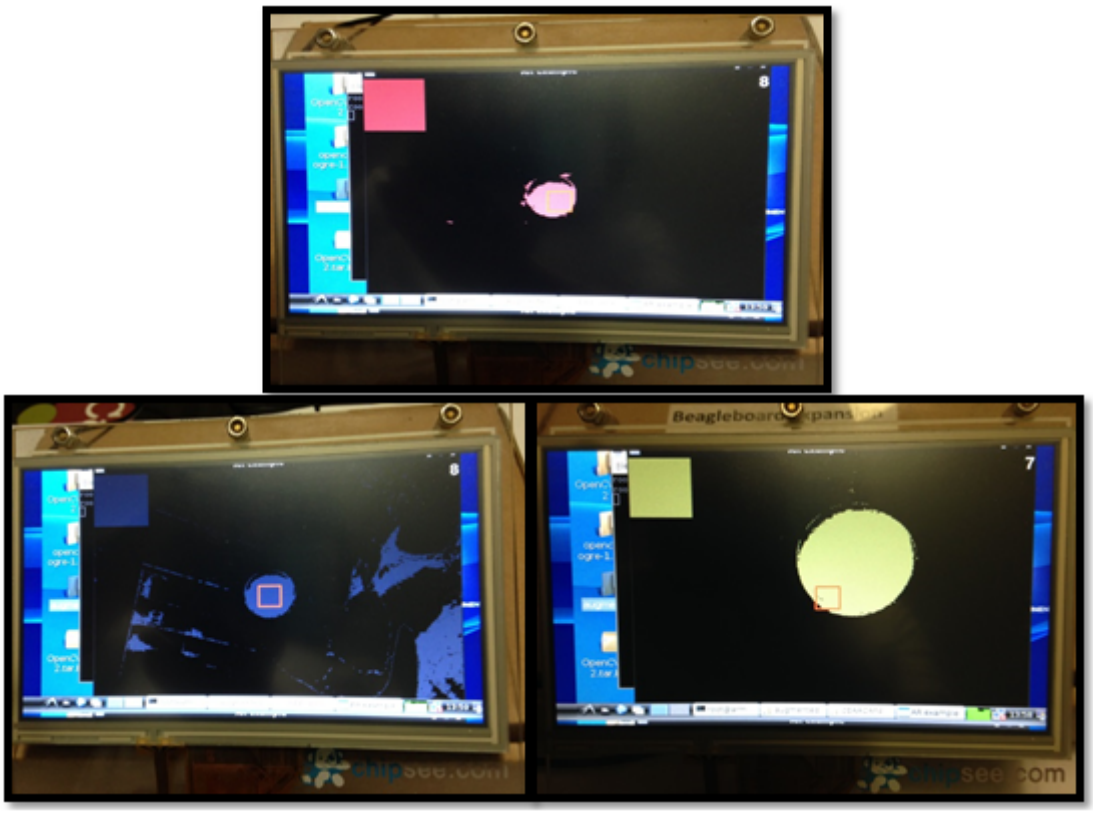

Figura 4.16: Reconocimiento de color con Pygame - Numpy.

Ahora, en modo de ejecución, el marcador es capaz de interactuar con objetos 2D, como las pelotas que se generan en la siguiente imagen.

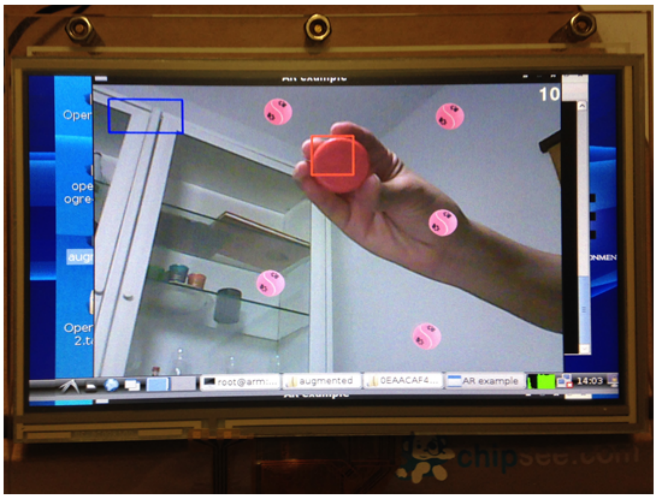

Figura 4.17: Interacción con objetos 2D.

El marcador puede golpear las pelotas para que se desplacen de un lado a otro. También es posible utilizar el marcador sin el recuadro rojo. En la siguiente figura se realiza una prueba con un objeto de color amarillo.

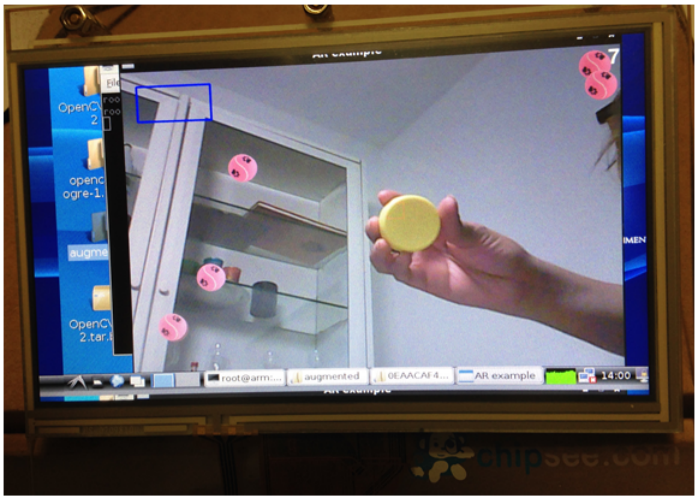

Figura 4.18: Marcador Amarillo Calibrado.

De igual manera, el marcador interactúa con botones. En la siguiente figura, se observa que al estar en contacto con el botón azul aparecen dos botones más, uno verde y uno rojo. Al estar en contacto con el botón verde aparece nuevamente el azul y al estar en contacto con el botón rojo se cierra la aplicación.

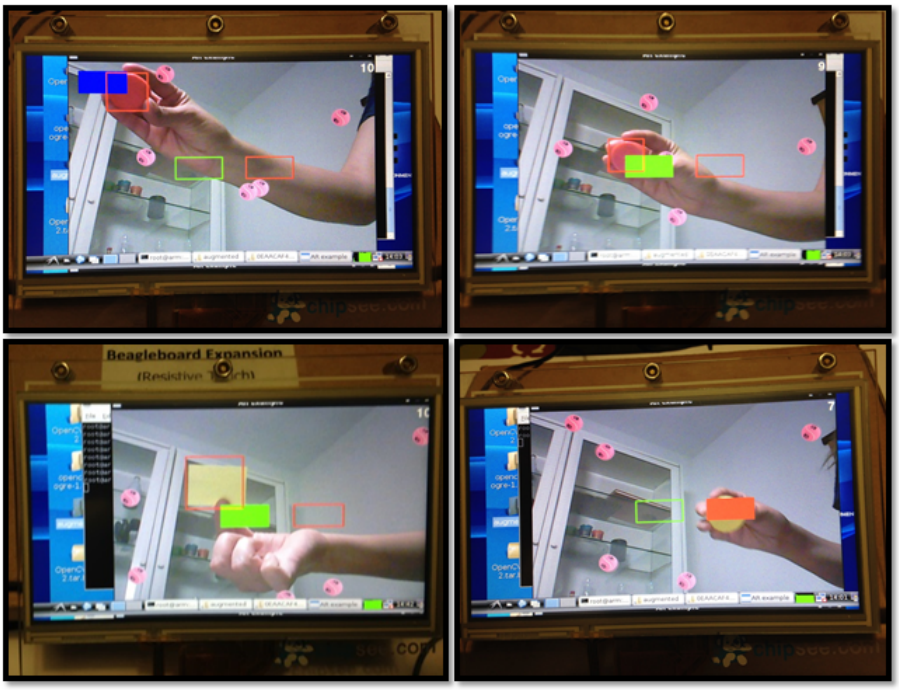

Figura 4.19: Interacción del Marcador con Botones.

La aplicación permite reconocer cualquier color de objetos sólidos con formas básicas. En algunas ocasiones la iluminación varía el color de los objetos. Por lo tanto las pruebas se deben realizar con una buena fuente de luz. La aplicación se ejecuta a una tasa de 12 fps; un rendimiento mayor al de las aplicaciones ejecutadas en los apartados anteriores.

Finalmente se realizaron pruebas con otros objetos de diferentes formas y colores, como se ilustra en la Figura 4.20. En dichas pruebas se obtuvo resultados similares, en el reconocimiento y rendimiento.

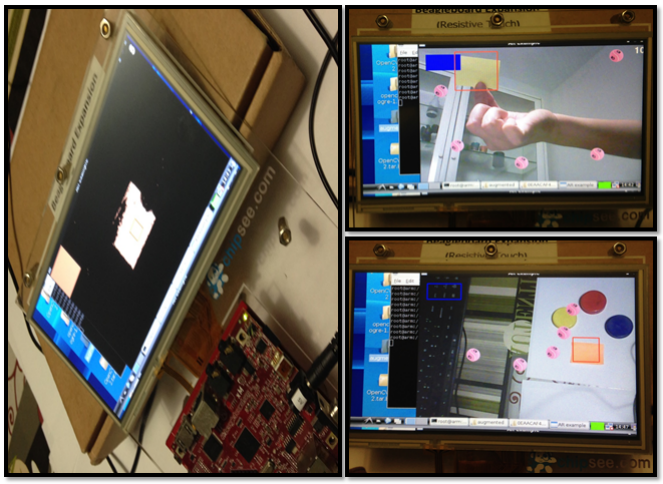

Figura 4.20: Aplicación de RA en Pygame - Numpy.

En la Tabla 4.1 se presenta un resumen del rendimiento de las librerías utilizadas para las pruebas en un Ubuntu embebido.

Ubuntu portado en la BeagleBoard Xm		
Librerías	FPS (Máx Frames Por Segundo)	
	Objetos 2D	Objetos 3D
ARToolKit	10	4
ArUCO	15	-
Pygame- Numpy	12	-

Tabla 4.1: Rendimiento de Gráficos en Ubuntu Embebido.

4.2. Aplicaciones Gráficas en Android Embebido

Existe una gran variedad de herramientas libres para el desarrollo de aplicaciones gráficas en Android. Sin embargo para la versión embebida no es posible utilizar todas las herramientas. En los siguientes apartados se describen los alcances de algunas aplicaciones desarrolladas para Android, desde las más básicas, como juegos sencillos, hasta aplicaciones de interacción con el mundo real con una cámara.

Antes de empezar, es importante mencionar que en la BeagleBoard xM no se pueden adquirir las aplicaciones a través de la tienda de Android, ya que no tiene asociada una cuenta de correo.

4.2.1. Programación de Videojuego Básico

Para crear un videojuego y ejecutarlo en la BeagleBoard xM, se utilizaron las herramientas comunes de programación en Android.

Software Requerido

Google ha diseñado un paquete de software Android SDK, que incorpora todas las herramientas necesarias para el desarrollo de aplicaciones en Android. En él se incluye conversor de código, debugger, librerías, emulador, documentación, entre otros[1]. Para desarrollar las aplicaciones se deben obtener las siguientes herramientas:

- Máquina virtual Java http://java.com/es/download/

[1]Más información sobre herramientas para Android en: http://www.androidcurso.com/index.php/tutoriales-android-fundamentos/31-unidad-1-vision-general-y-entorno-de-desarrollo/100-instalacion-del-entorno-dedesarrollo

- Android Studio http://developer.android.com/sdk/installing/studio.html
- Android SDK de Google http://developer.android.com/sdk

Desarrollo del Videojuego

Se programó un juego de tres en raya para ejecutarlo en la BeagleBoard xM. El juego se diseñó para un 1 jugador y multijugador. Como se menciona anteriormente, se utilizó Android Studio. Por lo tanto el código fuente fue escrito en eclipse. Para la simulación se hace uso de la herramienta Android Device Manager, de la interfaz de eclipse, donde es posible seleccionar diferentes dispositivos móviles con características específicas. De esta manera se puede crear una BeagleBoard xM virtual. Dicha interfaz, se abre en la opción Window de la barra de herramientas superior. En la siguiente figura se muestra la interfaz para configurar los emuladores.

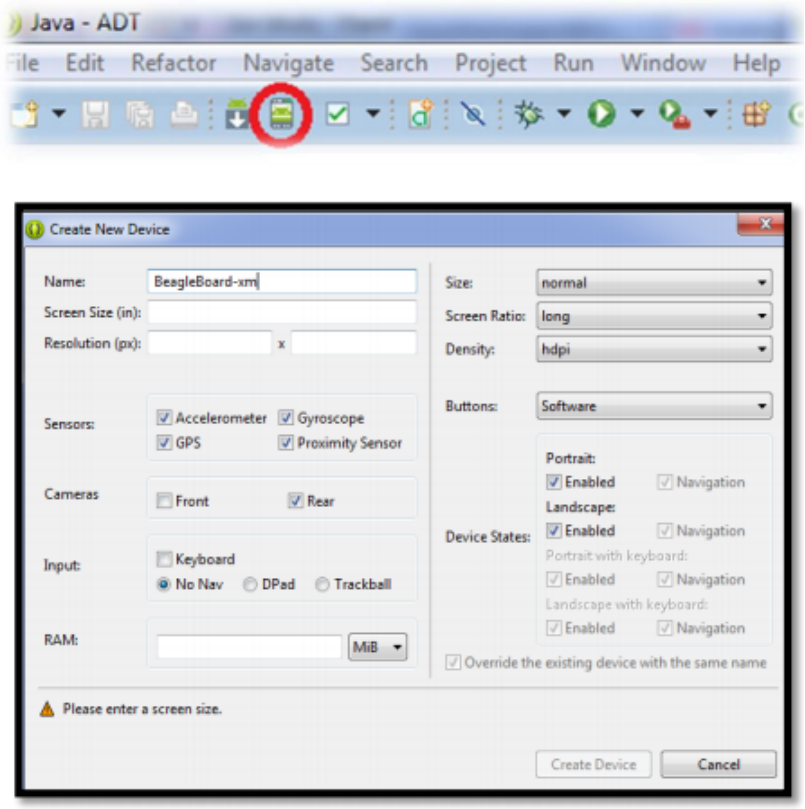

Figura 4.21: Android Virtual Device Manager.

Para empezar, se crea un proyecto nuevo en Android[1]. El juego se diseñó con la plataforma de desarrollo 4.0.3 de Android[2].

[1]Para crear un proyecto en Android consulte: http://www.androidcurso.com/index.php/147
[2]Más información sobre las plataformas de desarrollo para Android en: http://www.androidcurso.com/index.php/463

Descripción del Juego Tres en Raya

Una vez creado el proyecto se diseñan el ícono del juego y los layouts correspondientes. En el layout principal se ubican los botones para un jugador, multijugador y para salir del juego. El objetivo del juego es formar tres objetos en raya, para este caso los objetos son galletas o donuts. Los tres objetos pueden formar líneas de tres elementos de manera vertical, horizontal y en diagonal.

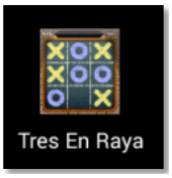

Figura 4.22: Ícono del Juego.

Cada botón del layout corresponde a una clase que implementa una actividad. En el modo de 1 jugador, el usuario juega con la máquina en diferentes niveles de dificultad. En el modo multijugador pueden jugar dos usuarios cambiando el turno cuando lo indique la máquina y finalmente el botón salir permite cerrar la interfaz del juego.

Figura 4.23: Layout principal.

Al utilizar el modo de 1 jugador se observan los tres niveles de dificultad; básico, intermedio y experto. En el nivel básico la máquina lanza tiros aleatorios. En el nivel intermedio se ha implementado inteligencia al juego de manera que bloquea algunos de los caminos para el usuario. Finalmente en el modo experto el usuario debe hacer un doble juego para engañar a la máquina y ganar.

Figura 4.24: Layout para 1 jugador.

En la Figura 4.25 se observa el juego en ejecución, el usuario juega en modo básico. En este caso, es el ganador al formar las tres galletas en raya.

Figura 4.25: Modo para un Jugador.

En la Figura 4.26 se presenta el juego en modo multijugador. Cada jugador escoge un objeto y lanza cuando sea su turno. En este caso, el turno es para el jugador de la Dónut.

Figura 4.26: Modo Multijugador.

Es posible instalar el juego en cualquier dispositivo con Android, con una versión inferior a la 4.0.3. Finalmente se genera el archivo con extensión .apk y se instala en la BeagleBoard xM. Esto puede hacerse guardando la aplicación en el correo electrónico o en Dropbox y desde allí se descarga y se ejecuta en la placa.

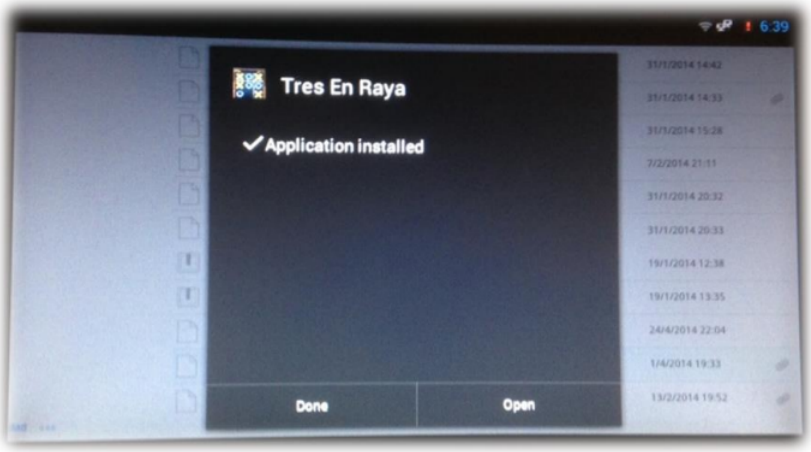

Figura 4.27: Instalación del Juego en la Beagleboard xM.

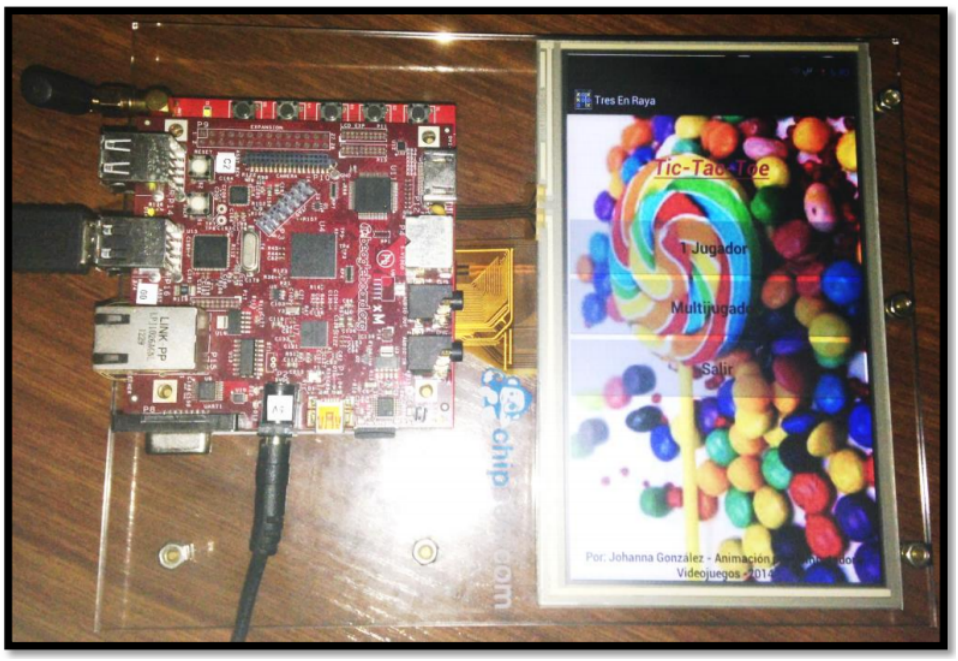

Figura 4.28: Ejecución del Juego en la BeagleBoard xM.

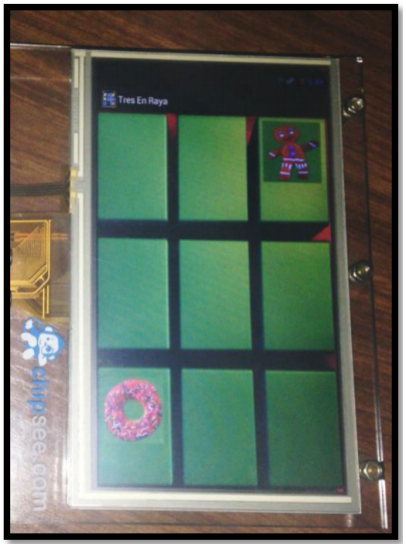

Figura 4.29: Ejecución del Juego en la BeagleBoard xM.

De lo anterior se concluye que un juego básico para Android, se ejecuta correctamente en la BeagleBoard xM, a una velocidad adecuada.

4.2.2. Videojuegos y Rendimiento de gráficos

En cuanto a los videojuegos, la BeagleBoard xM tiene la capacidad de ejecutar una gran variedad. Entre ellos cabe citar, Angry Birds, en las diferentes versiones para An-

droid, con un excelente rendimiento en la parte gráfica, como se puede observar en la Figura 4.30. Por otro lado es posible adaptar dispositivos de audio para el juego e interactuar con la pantalla táctil. Los videojuegos en la BeagleBoard xM, se ejecutan a una tasa aproximada de 30 fps.

Figura 4.30: Angry Birds en la BeagleBoard xM.

Otro ejemplo de videojuegos para la BeagleBoard xM, es el que se observa en la siguiente figura, donde se muestra un juego de carreras de vehículos. Para mover el vehículo de un lado a otro se utiliza el acelerómetro que tiene integrada la tarjeta de la pantalla táctil. El juego se ejecuta a una tasa de aproximadamente 30 fps, con un excelente detalle gráfico.

Figura 4.31: Acelerómetro en la BeagleBoard xM.

Pruebas de Gráficos

Se realizaron diferentes pruebas para evaluar el rendimiento de los gráficos en 2D y 3D para Android 4.0.3, en la BeagleBoard xM. Se utilizó la herramienta RowboPERF, que incluye demos en 2D y 3D, como los que se ilustran en la Figura 4.32. En cuanto a gráficos 2D, incluye ejemplos para; dibujar arcos, círculos y rectángulos; rellenar áreas; ejecutar animaciones con formato de mapa de bits y escribir texto. Para los gráficos en 3D incluye; aplicaciones para dibujar un cubo con colores, teteras, estrellas, entre otros

objetos, y por otro lado incluye demos para evaluar la iluminación, texturas, canal[1] alfa, efectos de la reflexión y multitexturas.

Figura 4.32: Pruebas de gráficos en 2D y 3D.

En la Figura 4.32 se presentan algunas de las diferentes demos 3D, incluidas en la herramienta RowboPERF. En la imagen superior derecha se observa la demo del hombre Camaleón, donde se muestra un personaje con la piel diseñada con matrices e iluminado con la técnica de 3 puntos por píxel. El modelo del hombre Camaleón tiene 19 huesos y un ciclo de animación de 16 frames. La figura inferior izquierda muestra un jarrón con reflexiones dinámicas en sus secciones metálicas. Finalmente, en la figura inferior derecha, la demo ilustra una colección de efectos de sombreado, que incluye efectos de textura, efectos basados en el medio ambiente e iluminación compleja aplicada objetos de geometría procesal.

En las tablas 4.2 y 4.3 se presentan los resultados de las pruebas realizadas con diferentes objetos para gráficos en 2D y 3D.

Una vez finalizadas las pruebas y analizados los resultados, se comprueba el alto rendimiento de la BeagleBoard xM bajo Android embebido.

4.2.3. Aplicación de la Cámara USB

La interacción con el mundo real es requerida en gran parte de las aplicaciones gráficas. Para ello es necesario el uso de una cámara. Para este caso se utilizó una cámara USB, teniendo en cuenta las razones expuestas en apartados anteriores.

Como se menciona en la Sección 3.2.2, para desarollar aplicaciones que requieran el uso de una cámara USB, es necesario configurar la aplicación para adquirir el vídeo a través del estándar UVC (USB Video Class).

[1]Es el proceso de combinar una imagen con el fondo para crear la apariencia de transparencia.

Gráficos en 2D (Objetos)	BeagleBoard xM	
	Frames por Segundo (fps)	Carga total en CPU (%)
Lienzo	59,05	33,94 (máx. 80.98)
Círculo	36,19	80.26 (máx. 88.83)
Círculo2	57,01	56.70 (máx. 93.60)
Recta	30,9	87.64 (máx. 99.03)
Arco	48,16	83.88 (máx. 93.20)
Imagen	54,66	91.95 (máx. 100.00)
Texto	58,61	57.02 (máx. 82.93)

Tabla 4.2: Rendimiento de Gráficos en 2D.

Gráficos en 3D (OpenGL)	BeagleBoard xM	
	Frames por Segundo (fps)	Carga total en CPU (%)
Cubo	58,39	28,16 (máx. 80.77)
Transparencia	59,71	48,74 (máx. 93.66)
Niebla	61,76	23,69 (máx. 78.33)
Tetera	60,60	42,49 (máx. 89.27)

Tabla 4.3: Rendimiento de Gráficos en 3D.

Se parte de la configuración del nodo del dispositivo de vídeo USB. Para la BeagleBoard xM, es el número 9, como se ha mencionado en apartados anteriores.

```
// Se utiliza /dev/videox (x=cameraId+cameraBase).
// Para la BeagleBoard xM /dev/video[9].
private int cameraId=9;
private int cameraBase=0;
```

Se configura la cámara web para una resolución de 640x480 con el formato YUV[1].

```
static final int IMG_WIDTH=640;
static final int IMG_HEIGHT=480;
```

Teniendo en cuenta que, para este caso es necesario escribir un método nativo para adquirir la imagen por el puerto USB, se deben utilizar algunas funciones nativas JNI[2], como las que se presentan a continuación:

```
// Funciones JNI
```

[1]YUV es un espacio de color, usado como parte de un sistema de procesamiento de imagen en color.
[2]Es un framework que permite la interacción entre programas de Java y C/C++

```
public native int prepareCamera(int videoid);
public native int prepareCameraWithBase(int videoid, int camerabase);
public native void processCamera();
public native void stopCamera();
public native void pixeltobmp(Bitmap bitmap);

    static {
        System.loadLibrary("ImageProc");
    }static final int IMG_WIDTH=640;
static final int IMG_HEIGHT=480;
```

Utilizando dichas funciones, se crea un método para adquirir una imagen de la cámara UVC, en formato bitmap. Una vez generado el .apk e instalado en la BeagleBoard xM, se ejecuta la aplicación como se ilustra en la Figura 4.33.

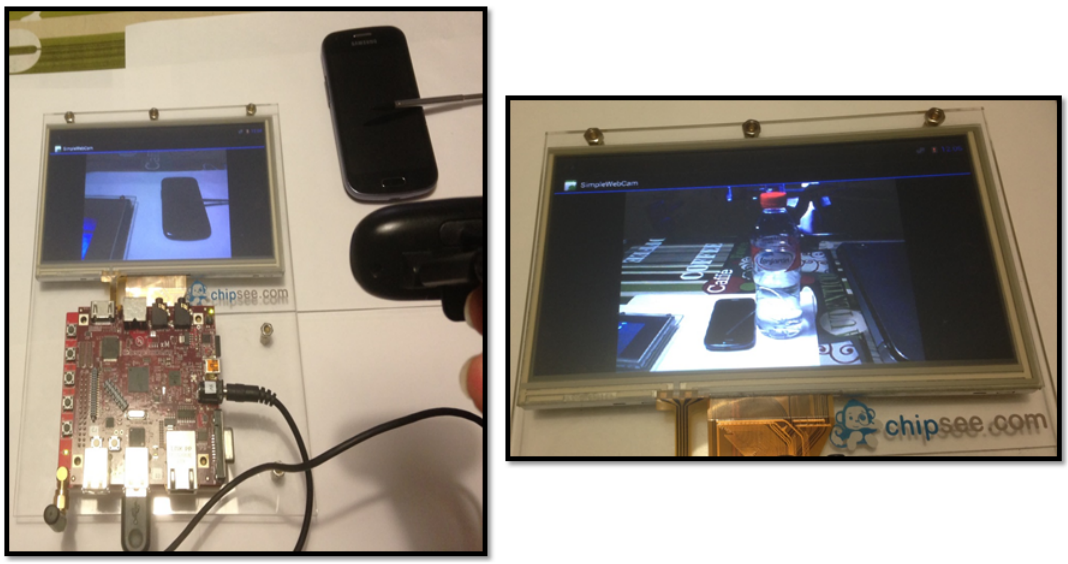

Figura 4.33: Cámara USB en la BeagleBoard xM portando Android.

4.2.4. Librerías de RA y OpenCV

Algunas herramientas para desarrollar aplicaciones de RA en Android, como AndAR, Vuforia y Metaio Mobile SDK (Serrano, 2012), están configuradas para las cámaras predeterminadas de los dispositivos móviles o tabletas. Por lo tanto no es posible utilizar una cámara USB para ejecutar aplicaciones diseñadas en dichos entornos.

La librería OpenCV tiene una versión para Android. Sin embargo sucede algo similar a lo mencionado, y aunque está disponible el código fuente y además es libre, el modificar la programación de la cámara para utilizar librerías nativas, como se describe en el apartado anterior, es una tarea compleja.

NyARToolKit es una librería para RA de uso libre (Serrano, 2012). Tiene disponible el código fuente para su descarga. Dicho código puede ser modificado y compilado sin inconvenientes, utilizando Android Studio.

Para adaptar la cámara USB al NyARToolKit, es necesario modificar la clase de la vista previa de la cámara, combinando la aplicación desarrollada en el apartado anterior con los métodos de dicha clase. Una vez realizadas las modificaciones, se compila el código, se genera el .apk correspondiente y se instala en la BeagleBoard xM. Al ejecutar la aplicación se genera un render animado que se desplaza sobre el marcador **Hiro**. El resultado obtenido se presenta en la Figura 4.34. La aplicación se ejecuta a una tasa aproximada de 15 fps.

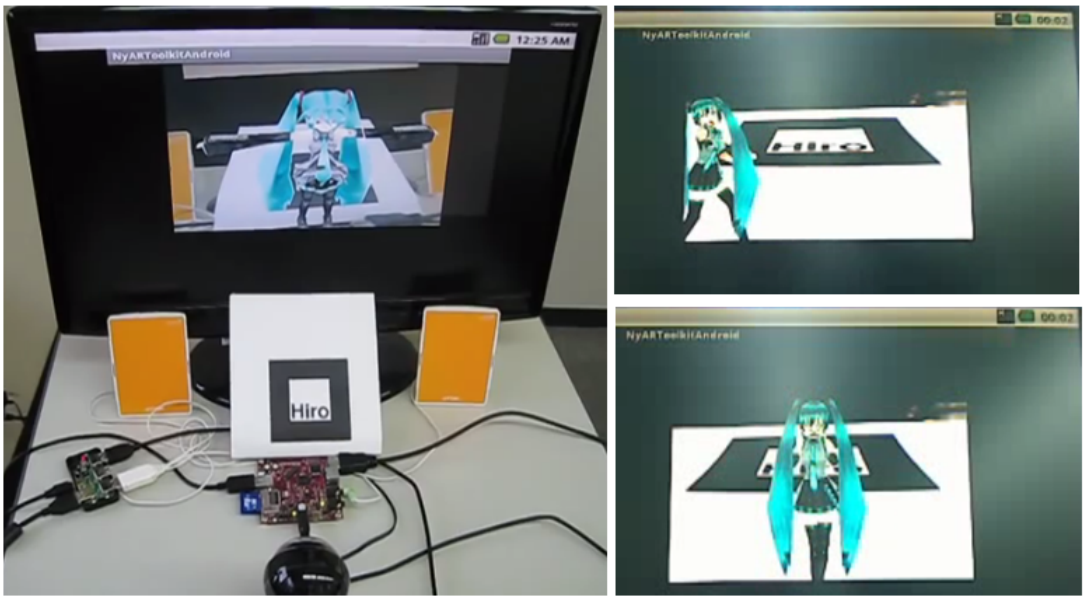

Figura 4.34: NyARToolKit en la BeagleBoard xM portando Android.

En la Tabla 4.4 se presenta un resumen del rendimiento de las librerías utilizadas para las pruebas en un Android embebido.

Android portado en la BeagleBoard Xm		
Librerías	**FPS (Máx Frames Por Segundo)**	
	Objetos 2D	Objetos 3D
Android Studio	60	-
OpenCV	60	58
NyARToolKit	-	15

Tabla 4.4: Rendimiento de Gráficos en Android Embebido.

4.3. Comparación de Resultados entre Ubuntu y Android

Luego de finalizar las pruebas realizadas con los dos sistemas operativos: Android y Ubuntu, se hace un análisis sobre las ventajas y desventajas de cada uno. Por una parte, Ubuntu permite fácilmente el uso de una cámara USB para las aplicaciones de reconocimiento de patrones o de RA. Sin embargo no tiene disponible una versión con pantalla táctil y con los demás dispositivos habilitados, como el acelerómetro, los botones de navegación y la antena de WiFi. Para ello hay que modificar y compilar el Kernel de Linux, una tarea larga y compleja. Además la conexión a la red debe realizarse por medio de un cable ethernet.

Por otro lado, Android no permite en un principio el uso de una cámara USB, por lo que se debe compilar el kernel nuevamente, añadiendo los controladores necesarios para el estándar de vídeo USB (UVC), y de igual manera se deben modificar todas las librerías en el caso de RA ya que vienen configuradas para otro tipo de cámaras. Sin embargo Android tiene un mejor rendimiento de gráficos 2D, 3D y una mejor interacción para el usuario con la pantalla táctil y la conectividad mediante WiFi, comparado con Ubuntu.

A continuación se presenta una tabla donde se resumen las características más importantes de cada sistema operativo, teniendo en cuenta los aspectos mencionados previamente.

	BeagleBoard xM	
	Ubuntu	**Android**
Licencia	Libre con restricciones de soporte para versiones antiguas	Libre para todas las versiones
Código Fuente	Modificable	Modificable
Pantalla Táctil	Disponible solo para la versión 11.04	Disponible para todas las versiones
Uso de periféricos (Teclado, Ratón)	Obligatorio, excepto para la versión 11.04	Opcional
Acelerómetro	No Disponible	Disponible
Botones de Navegación	No Disponibles	Disponibles
Cámara USB	Disponible	No Disponible (Se habilita en el Kernel)
Rendimiento de gráficos 2D Y 3D	Máximo 15 fps	Hasta 61 fps
Rendimiento de adquisición de video y Realidad Aumentada	Máximo 12 fps	Máximo 15 fps
Librerías	Se ejecutan correctamente	Es necesario modificar las que requieran el uso de una cámara
Conectividad	Ethernet	WiFi

Tabla 4.5: Tabla comparativa entre Android y Ubuntu portados en la BeagleBoard xM.

Capítulo 5

Conclusiones y Trabajo Futuro

La principal motivación de este trabajo de investigación consistió en construir un dispositivo hardware que permitiera ejecutar diferentes aplicaciones gráficas, con un rendimiento comparable a los dispositivos móviles o a las tabletas. Se seleccionaron los sistemas embebidos teniendo en cuenta sus altas capacidades de procesamiento y las múltiples herramientas disponibles para su programación. Por otro lado, se requería que el dispositivo se adaptara fácilmente a diferentes estructuras, de manera que fuera posible integrarlo a juguetes, vehículos u otro tipo de dispositivos para el entretenimiento, aprendizaje o procesos industriales.

5.1. Conclusiones

Durante el desarrollo del presente trabajo se han conseguido una serie de objetivos planteados en un principio:

1. Se realizó una búsqueda exhaustiva de diferentes sistemas embebidos que cumplieran con las condiciones requeridas para la construcción del dispositivo hardware. Se estudiaron las características y herramientas de programación de cada uno. Finalmente fue seleccionada la BeagleBoard xM, para la construcción de un prototipo de pruebas, teniendo en cuenta sus prestaciones y precio.

2. Se hizo una revisión bibliográfica sobre las aplicaciones gráficas existentes, desarrolladas en la BeagleBoard xM, con el fin de conocer sus alcances, limitaciones, rendimiento y las herramientas software utilizadas en los diferentes proyectos.

3. Una vez seleccionada la BeagleBoard xM, se buscaron diferentes accesorios para la construcción del prototipo, como la pantalla táctil, la cámara, la batería portátil, diferentes periféricos, entre otros. Por otro lado se analizaron algunos aspectos, como el tamaño, la conectividad y el precio final.

4. Se realizó un estudio sobre los diferentes sistemas operativos disponibles para la BeagleBoard xM, y luego de un análisis exhaustivo, se seleccionó Android y Linux para realizar las pruebas, teniendo en cuenta que son sistemas operativos libres y

con una gran variedad de herramientas gratuitas, disponibles para el desarrollo de aplicaciones.

5. Tanto Android como Ubuntu fueron portados en la BeagleBoard xM. Se trabajó con varias versiones y se describieron detalladamente los problemas presentados en la instalación y configuración de cada sistema operativo. Finalmente, se señalaron las ventajas y desventajas de cada uno.

6. Teniendo en cuenta la revisión bibliográfica realizada en un principio, se utilizaron diferentes herramientas software para evaluar el rendimiento de las aplicaciones gráficas de 2D y 3D en la BeagleBoard xM, haciendo énfasis en aplicaciones de interacción en tiempo real, como el reconocimiento de patrones y la RA.

Con respecto al primer objetivo, se encontró una gran variedad de sistemas embebidos, entre ellos están; las FPGAs, las plataformas OMAP, Arduino, Raspberry Pi y la Beagleboard. Las FPGAs y las plataformas OMAP tienen un coste elevado y su programación es compleja. Arduino se especializa fundamentalmente en aplicaciones de electrónica y robótica. Por lo tanto no está orientado hacia la parte gráfica. Raspberry Pi, es una tarjeta de bajo coste y especializada en multimedia. Pero su procesador no tiene la potencia suficiente para la ejecución de aplicaciones gráficas. Finalmente la BeagleBoard, combina su gran capacidad de rendimiento con un precio asequible, además de la gran cantidad de herramientas libres y disponibles para su programación. Teniendo en cuenta estos aspectos se seleccionó la última versión de la BeagleBoard para construir un prototipo de pruebas.

En el segundo objetivo, se encontraron una gran cantidad de aplicaciones gráficas desarrolladas en las diferentes versiones de la BeagleBoard, en su mayoría aplicaciones para el reconocimiento de patrones, como el color o la forma y el conteo de objetos. En cuanto a aplicaciones de RA se han hecho principalmente trabajos utilizando FPGAs y solo algunos utilizando la BeagleBoard. La librería más utilizada para las diferentes aplicaciones gráficas ha sido OpenCV, debido a que es gratuita y posee diversas herramientas útiles para el desarrollo de aplicaciones gráficas.

Teniendo en cuenta que se requería la portabilidad del prototipo, se encontraron varios accesorios de la BeagleBoard xM, para cumplir este objetivo. Se seleccionó un dispositivo de visualización con pantalla táctil, que incluye una antena WiFi, acelerómetro, y botones de navegación. De igual manera hay disponibles cámaras y baterías portátiles para la BeagleBoard xM. Sin embargo dichos accesorios incrementan el coste del prototipo significativamente. Por lo tanto se decidió utilizar una cámara web. Finalmente para la alimentación se utilizó un adaptador de 5 Volts.

Con respecto al cuarto objetivo, se encontró que existen varios sistemas operativos para la BeagleBoard xM, como Linux, VxWorks, Android, iOS y Windows CE. Teniendo en cuenta que el objetivo es desarrollar aplicaciones gráficas, se seleccionó Android y Linux, para hacer las pruebas en el prototipo, ya que son libres, gratuitos y han sido utilizados ampliamente en diferentes proyectos de reconocimiento de patrones, RA y videojuegos.

En el quinto objetivo se instaló Ubuntu y Android en la BeagleBoard xM. En cuanto a la instalación de Ubuntu, se presentaron problemas con las versiones inferiores a la 12, ya que no tienen soporte de Cannonical, y por lo tanto solo es posible utilizar las versiones LTS o con soporte de larga duración. Además, no existe una versión disponible de Ubuntu, con soporte, que tenga configurada la pantalla táctil, el acelerómetro y los botones de navegación. Para su funcionamiento, se debe cambiar la configuración del Kernel y compilarlo nuevamente. Por esta razón la interacción del usuario y la BeagleBoard xM, se debe realizar mediante el teclado y el ratón, lo que disminuye la portabilidad del prototipo. Finalmente, fue necesario instalar una interfaz gráfica para interactuar con el sistema operativo.

En cuanto a la instalación de Android, no se presentaron inconvenientes de soporte. Existen múltiples versiones disponibles de descarga gratuita para Android, con la pantalla táctil, el acelerómetro y la antena WiFi habilitada. En este caso se trabajó con la última disponible, que es la Ice Cream Sandwich 4.0.3. Sin embargo, en Android se presentaron problemas para utilizar la cámara web, ya que no viene habilitado el controlador por defecto. Para habilitar el controlador UVC que permite activar la cámara USB, fue necesario modificar y compilar el Kernel de Linux. Esta tarea fue compleja y requirió de una larga investigación y diferentes pruebas.

En el objetivo final, se realizaron diferentes pruebas con varias herramientas para el desarrollo de aplicaciones gráficas en Ubuntu y Android. En Ubuntu se utilizó NyARToolkit, OpenCV, ArUCO y Python. NyARToolKit se instaló y ejecutó correctamente en la BeagleBoard xM. Sin embargo su rendimiento no fue el esperado, ya que los gráficos en 2D se ejecutaron a una tasa máxima de 10 fps y los gráficos en 3D a una tasa máxima de 4 fps para una resolución de 320 x 240. ArUCO y Python utilizan OpenCV para activar el vídeo de la cámara, además de otras funciones. Por lo tanto el primer paso fue instalar la librería de OpenCV, dicha librería se instaló y configuró correctamente tras solventar algunos problemas y fue un proceso largo. En el caso de ArUCO, fue posible su instalación y el entrenamiento de los marcadores. Sin embargo la adaptación de OGRE como librería de objetos render, no se logró debido a problemas de repositorios desactualizados. Se decidió trabajar con aplicaciones más sencillas. Por ello se instaló una aplicación de RA sencilla con reconocimiento de color, desarrollada en Python. Dicha aplicación presentó un buen rendimiento y se ejecutó a una tasa de 12 fps.

En Android, se programó un juego básico con Android Studio, el juego se ejecutó correctamente. Por otro lado se observó que los videojuegos como Angry Birds y de carreras de coches tienen un alto rendimiento gráfico. Además pueden interactuar con el acelerómetro, integrado en la tarjeta de la pantalla táctil. También, se realizaron pruebas para evaluar el rendimiento de los gráficos 2D y 3D, utilizando la herramienta RowboPERF. Los gráficos en 2D se ejecutaron a una tasa máxima de 60 fps y los gráficos en 3D a una tasa máxima de 61 fps. De estas pruebas se concluye, que notablemente Android tiene un rendimiento gráfico superior a Ubuntu. Sin embargo, la principal desventaja de Android, está en las aplicaciones con interacción en tiempo real que requieren una cámara USB; como las de reconocimiento de patrones y las de RA, ya que es necesario configurar las librerías para el uso de una cámara web. En algunos casos esto es posible, como en NyARToolKit. Aunque es una tarea compleja debido al volumen de código y a que la do-

cumentación está escrita en Japonés. En otros casos como en AndAR, Vuforia y Metaio Mobile SDK se debe utilizar la cámara nativa.

Para finalizar, se realiza un análisis global de los resultados en la Tabla 5.1.

	Herramientas para el desarrollo de aplicaciones gráficas en la BeagleBoard Xm					
	Ubuntu Embebido			Android Embebido		
Librerías	**ARToolKit**	**ArUCO**	**Pygame - Numpy**	**Android Studio**	**OpenCV**	**NyARToolKit**
Instalación, requiere modificación del código	No	No	No	No	Sí	Sí
Herramientas Adicionales Requeridas	Ninguna	OpenCV, OGRE o cualquier librería Render	OpenCV	Ninguna	Librería nativa OpenCV	Android Studio
Rendimiento de Gráficos 2D	Medio	Medio	Alto	Alto	No es posible activar la cámara web	Medio
Rendimiento de Gráficos 3D	Bajo	No es posible integrar con librerías Render, Problemas de repositorios	Requiere librería Render	Alto	Requiere librería Render	Medio
Rendimiento de Aplicaciones de RA	Bajo		Alto	Requiere librerías de RA	Requiere librerías de RA	Medio
Recomendación	Utilizar cámara de alta velocidad	Modificar versión de Ubuntu	Implementar librería para gráficos 3D	Recomendado para el desarrollo de aplicaciones embebidas	Utilizar cámara nativa o modificar el código fuente	Utilizar cámara nativa o modificar el código fuente

Tabla 5.1: Resumen global de los resultados obtenidos en el prototipo de pruebas.

5.2. Trabajo Futuro

El trabajo presentado, es el primer paso en el desarrollo de prototipos hardware para la ejecución de aplicaciones gráficas. Como continuación de la investigación se proponen los siguientes puntos:

- **Modificar Versiones de Ubuntu.** Modificar el Kernel de manera que la última versión disponible de Ubuntu, permita el uso de la pantalla táctil, el acelerómetro y los botones de navegación. En el presente trabajo se describió el proceso para programar y compilar el kernel de Linux.
- **Modificar Herramientas Software.** Programar algunas de las herramientas software libres de Android, como OpenCV, para el uso de una cámara USB. Hecho que permitiría la ejecución de diversas aplicaciones de interacción con el mundo real en la BeagleBoard xM.
- **Cámara Integrada en la BeagleBoard xM.** Realizar pruebas con la cámara LI (Ver Sección 3.5), que está especialmente diseñada para la BeagleBoard xM. Dicha cámara tiene una mayor velocidad de procesamiento y al utilizarla posiblemente se superarían los problemas de programación que se presentan con la cámara USB.

Por otro lado, sería posible emplear las herramientas para Android, como AndAR, Vuforia y Metaio Mobile SDK en la BeagleBoard xM.

- **Batería Portátil.** Al utilizar una batería portátil recargable, se incrementaría la portabilidad del prototipo hardware.
- **Tarjeta PandaBoard.** La PandaBoard es una placa de desarrollo similar a la BeagleBoard xM. Sin embargo tiene un procesador más potente (ARM Cortex-A9). Por otro lado cuenta con un procesador gráfico integrado (SGX540), su precio solo se incrementaría en un porcentaje pequeño, con respecto a la BeagleBoard xM. Además, dicha tarjeta se ha utilizado con éxito en aplicaciones de RA donde se requiere un alto nivel en el procesamiento gráfico. Al emplear la PandaBoard para la construcción del prototipo, probablemente se incrementaría el rendimiento de las aplicaciones gráficas.

Bibliografía

Aby, P., Jose, A., Dinu, L., John, J., and Sabarinath, G. (2011). Implementation and optimization of embedded face detection system. In *Signal Processing, Communication, IEEE International Conference on Computing and Networking Technologies (ICSCCN)*, pages 250–253.

Al Afif, F., Rachmadi, M. F., Wibowo, A., Jatmiko, W., Mursanto, P., and Ma'sum, M. (2011). Enhanced adaptive traffic signal control system using camera sensor and embedded system. In *IEEE International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, pages 367–372.

Basu, A., Saupe, C., Refour, E., Raij, A., and Johnsen, K. (2012). Immersive 3dui on one dollar a day. In *IEEE Symposium on 3D User Interfaces (3DUI)*, pages 97–100.

Brinkman, W. F., Haggan, D. E., and Troutman, W. W. (1997). A history of the invention of the transistor and where it will lead us. *IEEE Journal of Solid-State Circuits*, 32(12):1858–1865.

Brock, D. C. and Laws, D. A. (2012). The early history of microcircuitry: An overview. *IEEE Annals of the History of Computing*, 34(1):7–19.

Celis, C. A. D. and Molano, C. A. R. (2012). Navegación de robot móvil usando kinect, opencv y arduino. *Prospectiva*, 10(1):71–78.

ChipSee Info and Tech Co., L. (2011). http://download.tigal.com/chipsee/BeagleBoard_user_guide.pdf. BealgeBoard xM Expansion V2.

Coombs, J. and Prabhu, R. (2011). *Texas Instruments*. OpenCV on TI's DSP+ ARM® platforms: Mitigating the challenges of porting OpenCV to embedded platforms.

EmbeddedAR (2011). Embeddedar. http://www.hitlabnz.org/index.php/component/jresearch/?view=project&task=show&id=32. Fecha de Consulta: 2014-07-16.

Faggin, F., Hoff, M. E., Mazor, S., and Shima, M. (1996). The history of the 4004. *Micro*, 16(6):10–20.

Ferguson, R. (2013). *Practical Algorithms for 3D Computer Graphics, Second Edition*. Taylor & Francis.

Foley, J. (1996). *Computer Graphics: Principles and Practice*. Addison-Wesley systems programming series. Addison-Wesley.

Grüner, C. M. (2012). *Diseño e implementación de un sistema de transmisión de imágenes de anaglifo sobre dos plataformas embebidas BeagleBoard xM*. PhD thesis, Instituto Tecnológico de Costa Rica. Escuela de Electrónica.

Kato, H. and Billinghurst, M. (1999). Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *2nd IEEE and ACM International Workshop on Augmented Reality. (IWAR '99)*., pages 85–94.

Kleidermacher, D. and Kleidermacher, M. (2012). *Embedded systems security: practical methods for safe and secure software and systems development*. Elsevier.

Langbridge, J. (2014). *Professional Embedded ARM Development*. Wrox professional guides. Wiley.

Lipinski, P., McCabe, S., Stark, A., Di Caterina, G., Clemente, C., and Soraghan, J. (2012). Robokinect—a low-cost mobile vision system for 2.5 d object detection. In *5th IEEE European DSP Conference (EDERC) in Education and Research*, pages 276–280.

Liquidware.Corp (2014). Beaglejuice. Fecha de Consulta: 2014-07-20.

Majumder, K. and Rathna, G. (2013). A beagleboard xm based gesture control input device for pc. In *IEEE Educators' Texas Instruments Conference in India (TIIEC)*, pages 359–361.

Munoz-Salinas, R. (2012). Aruco: A minimal library for augmented reality applications based on opencv. Universidad de Córdoba.

Nelson, R. (2014). Beagleboard ubuntu. http://elinux.org/BeagleBoardUbuntu#Quantal_12.10_armhf. Fecha de Consulta: 2014-07-21.

NyARTK (2012). Nyartoolkit project. http://nyatla.jp/nyartoolkit/wp/?page_id=198. Fecha de consulta: 2014-09-03.

Oxer, J. and Blemings, H. (2009). *Practical Arduino: cool projects for open source hardware*. Apress.

Pal, A. (2012). *Microcontrollers: Principles and Applications*. phi Philosophical Enterprises.

Parker, M. (2010). *Digital Signal Processing 101: Everything you need to know to get started*. Elsevier Science.

Peddie, J. (2013). Developing the computer. In *The History of Visual Magic in Computers*, pages 125–210. Springer.

Poudel, P. and Shirvaikar, M. (2010). Optimization of computer vision algorithms for real time platforms. In *42nd IEEE Southeastern Symposium on System Theory (SSST)*, pages 51–55.

Serrano, A. (2012). Herramientas de desarrollo libres para aplicaciones de realidad aumentada con Android. Análisis comparativo entre ellas. Tesina del máster en IARFID, DSIC, UPV.

Solak, S. and Bolat, E. (2013). Real time industrial application of single board computer based color detection system. In *8th IEEE International Conference on Electrical and Electronics Engineering (ELECO)*, pages 353–357.

Teoh, S. K., Yap, V. V., Soh, C. S., and Sebastian, P. (2012). Implementation and optimization of human tracking system using arm embedded platform. In *4th IEEE International Conference on Intelligent and Advanced Systems (ICIAS)*, volume 1, pages 353–356.

Wolf, W. and Madsen, J. (2000). Embedded systems education for the future. *Proceedings of the IEEE*, 88(1):23–30.