NxtAR: Un Sistema de Control para Robots Móviles Basado en Realidad Aumentada

Miguel Astor Romero[1], Walter Hernández[2], David Pérez Abreu[1]
miguel.astor@ciens.ucv.ve, walter.hernandez@ciens.ucv.ve, david.perez@ciens.ucv.ve

[1] Centro CICORE, Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela
[2] Centro de Computación Gráfica (CCG), Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela

Resumen: La proliferación de los dispositivos móviles de alto desempeño para consumidores ha traído con sigo la posibilidad de difundir tecnologías como la realidad aumentada, algo que en tiempos anteriores era sumamente engorroso e incluso inviable. Así mismo, el campo de la robótica se ha beneficiado de la aceptación por parte del público general de los robots para entornos caseros y usuarios aficionados, como los distribuidos por The LEGO Group. Dada la amplia disponibilidad de estos dispositivos, se hace factible el desarrollo de arquitecturas complejas que combinen ambas tecnologías manteniendo un bajo costo. En base a esto el presente trabajo de investigación plantea el diseño y desarrollo de un sistema de contol en hardware y software para robots móviles, el cual hace uso de las posibilidades provistas por la realidad aumentada para crear experiencias de control enriquecidas por interacciones virtuales. Adicionalmente se presenta una implementación de este sistema de control para dispositivos basados en el sistema operativo Android, incluyendo las consolas de video-juegos OUYA y utilizando un robot móvil LEGO Mindstorms NXT. Este sistema de contol funciona sobre redes de area local inalámbricas y permite ejecutar un escenario donde el usuario debe desarmar una serie de artefactos explosivos virtuales a una tasa de ejecución usable.

Palabras Clave: Robótica; Realidad Aumentada; Robots Móviles; Sistema de Control; LEGO Mindstorms; Android; OUYA.

Abstract: The proliferation of high performance mobile devices for consumers has brought with it the possibility of disseminating technologies such as augmented reality, something that was considerably cumbersome and even unfeasible before. Likewise, the field of robotics has been benefited from the general public's acceptance of robots for home environments and amateur users, like those distributed by The LEGO Group. Given the high availability of these devices, the development of complex architectures combining both technologies becomes feasible, while maintaining low costs. Based on this, the present research proposes the design and development of a hardware and software control system for mobile robots, making use of the possibilities provided by augmented reality in order to create control experiences enriched by virtual interactions. Additionally an implementation of this control system for Android based devices is shown, making use of a LEGO Mindstorms NXT mobile robot. This control system functions on wireless local area networks and allows the execution of a proof-of-concept scenario where the user must defuse a series of virtual explosive devices at a usable framerate.

Keywords: Robotics; Augmented Reality; Mobile Robots; Control System; LEGO Mindstorms; Android; OUYA.

1. Introducción

Hoy en día la alta proliferación de dispositivos móviles programables de alto desempeño coloca en las manos de los usuarios y de los desarrolladores la posibilidad de investigar y desarrollar a plenitud el uso de tecnologías que dadas su naturaleza se benefician ampliamente de las prestaciones de estos dispositivos. Una de estas tecnologías es la Realidad Aumentada, la cual a pesar de no ser una tecnología reciente ha presentado dificultad en su difusión dadas sus necesidades con respecto a movilidad, disponibilidad de sensores y alto desempeño computacional. Los equipos y dispositivos necesarios para garantizar dichos requerimientos, antes de la llegada de los *smartphones* y *tablets*, se caracterizaban por ser pesados e incómodos, sumamente costosos, o por poseer un desempeño muy bajo.

Por otro lado, la robótica de bajo costo también ha disfrutado de un gran empuje en tiempos recientes gracias a proyectos enfocados a llevar esta tecnología a las masas. Algunos proyectos comerciales ubicados en este ámbito son los proyectos Arduino[1], Raspberry Pi[2] y LEGO Mindstorms[3]. Estos proyectos permiten a usuarios profesionales y aficionados realizar desarrollos que involucren robots y sistemas de sensores a muy bajo costo y con una gran cantidad de herramientas de apoyo.

La disponibilidad de las tecnologías mencionadas permite el desarrollo de proyectos de investigación que las combinen para obtener sistemas complejos. Una categoría de estos sistemas

[1] http://www.arduino.cc
[2] http://www.raspberrypi.org
[3] http://mindstorms.lego.com

son los sistemas de control, los cuales permiten a un operador humano manipular a uno o más robots para lograr un objetivo que generalmente involucra el acceso a áreas difíciles o tareas de alto riesgo físico para la integridad de un humano.

Actualmente han sido desarrollados múltiples sistemas de control para robots móviles que hacen uso de Realidad Aumentada para enriquecer la simulación presentada al usuario. Ejemplos de esto son los sistemas *Augmented Colliseum* de Minoru Kojima y colaboradores [1], los sistemas *VisiCon* y *CoGame* de Kazuhiro Hosoi y colaboradores [2], [3] y el *framework* ARATG de Tian Xie y colaboradores [4]. Sin embargo, todos estos trabajos hacen uso ya sea de herramientas *ad hoc* diseñadas para el problema en cuestión o de dispositivos comerciales para propósitos específicos. Esto limita a quienes desean realizar investigaciones en esta área dado que dichas soluciones, si es que pueden ser adquiridas en primer lugar, suelen implicar un desembolso económico elevado. Esto nos lleva a preguntarnos si será posible aprovechar las capacidades provistas por los dispositivos móviles para consumidores, por ejemplo los *smartphones* y las *tablets*, junto a robots modulares de bajo costo para desarrollar un sistema de control por realidad aumentada con características sofisticadas.

En base a lo planteado, en este trabajo de investigación se diseño y desarrolló un sistema de realidad aumentada para *tablets* y *smartphones*, el cual involucra un robot móvil controlado por teleoperación. El sistema fue diseñado de manera que se puedan utilizar dispositivos comerciales de bajo costo como los mencionados anteriormente. Así mismo, El sistema permite controlar un robot móvil de forma remota, presentando al usuario un entorno virtual con el cual es posible simular distintos escenarios que en la realidad pueden ser muy costosos o que incluso representen un peligro para la integridad física del usuario. Muchas aplicaciones pueden desarrollarse a partir de la solución mostrada en esta propuesta, pasando por campos tan diversos como la exploración, el entretenimiento, el entrenamiento, entre otros.

El resto del artículo está organizado en las siguientes secciones. La Sección 2 detalla los antecedentes y los trabajos relacionados con esta investigación. En la Sección 3 se describen las características y prestaciones de las distintas herramientas de hardware y software utilizadas durante el desarrollo del trabajo. La Sección 4 contiene una descripción detallada del sistema de control de robots propuesta para cumplir con los objetivos establecidos anteriormente. En la Sección 5 se detalla el proceso de desarrollo de una implementación del sistema propuesto para dispositivos móviles basados en el sistema operativo Android. A continuación, en la Sección 6 se presentan los resultados de una serie de pruebas realizadas sobre la implementación de referencia. Finalmente, en la Sección 7 se presentan las conclusiones del trabajo, describiendo los aportes realizados y las limitaciones encontradas, así como el planteamiento de posibles trabajos futuros.

2. Trabajos Relacionados

En 1993 Paul Milgram y colaboradores [5] presentaron un trabajo de investigación y desarrollo en el cual hacían uso de un sistema de realidad aumentada para controlar a un robot manipulador industrial[4]. Este trabajo consistía en una estación de trabajo conectada a un brazo robótico el cual era filmado por una cámara de video. En la pantalla de la estación de trabajo se mostraba al usuario una imagen en la cual un modelo 3D del robot era desplegado en modalidad de mallado sobre una captura de video del mismo. El usuario tenía la posibilidad de manipular el modelo del robot, cambiándolo de posición o rotando sus distintos componentes; luego el robot real imitaría los movimientos producidos por el usuario en el modelo. Variaciones recientes sobre este trabajo fueron presentadas por Chong y colaboradores en el año 2009 [7], y por Fang y colaboradores entre los años 2012 y 2013 [8], [9].

Una investigación similar orientada al área de los robots móviles se encuentra en el trabajo de Alonzo Kelly y colaboradores [10], quienes presentaron una serie de técnicas y herramientas que tratan de minimizar la dificultad inherente al control remoto de robots móviles en enlaces inalámbricos; particularmente en entornos donde las distancias o la latencia de las comunicaciones en red hace imposible el obtener una respuesta inmediata del robot. El sistema funciona construyendo un modelo 3D del entorno del robot en tiempo real basándose en imagenes de video capturadas por el robot, utilizando la información obtenida de los otros sensores del robot como complemento.

El trabajo de Kelly y colaboradores [10] ha sido continuado por otros investigadores, quienes han desarrollado soluciones que extienden el alcance del mismo. Ejemplos de esto son las investigaciones realizadas por José Guivant [11] y Fumio Okura [12], [13] junto a sus respectivos colaboradores. Guivant y colaboradores desarrollaron una extensión del trabajo de Kelly la cual permite el control del robot a través de Internet. Por otra parte Okura y colaboradores desarrollaron un método que permite observar el entorno 3D generado a partir de los datos sensados por el robot desde puntos de vista arbitrarios de manera que sea posible controlar el robot con una cámara de tercera persona, es decir, viendo al robot desde un punto de vista externo al mismo. Otro trabajo relacionado es el desarrollado por Tian Xie y colaboradores [4], quienes presentaron un *framework* para el control remoto de robots por realidad aumentada llamado ARATG (*Augmented Reality Aided Teleoperation Guidance* - Orientación en Teleoperación Asistida por realidad aumentada) usando los resultados de Kelly y colaboradores como base.

Estos sistemas de control basados en realidad aumentada fueron estudiados desde el punto de vista del operador humano por Scott Green y colaboradores en [14]. Ellos realizaron un estudio experimental en el cual se compararon dos sistemas

[4]Un manipulador industrial es un tipo de robot utilizado en la industria para manipular piezas en lineas de ensamblado [6]. Son un tipo particular de brazo robótico.

de control con robots simulados: uno basado en una cámara que graba el punto de vista del robot y otro basado en realidad aumentada que permite al operador observar al robot dentro del contexto de su entorno de trabajo. Los resultados del estudio mostraron que el sistema basado en realidad aumentada mejoraba la capacidad de planificación de los operadores permitiéndoles realizar las tareas planteadas por el estudio con un mejor desempeño, eliminando la necesidad de inferir las condiciones del entorno de trabajo del robot como es necesario con el sistema básado en la cámara.

Este trabajo de investigación se basa en un trabajo anterior desarrollado por los autores, titulado "Visión por Computador para Robots Mindstorms NXT" [15]. En dicho trabajo se modificó la aplicación MINDdroidCV para el sistema operativo Android, la cual permite controlar un robot LEGO Mindstorms NXT utilizando un *smartphone* como control remoto. Dicha modificación, basada en el trabajo de Richárd Szabó [16], consistió en incorporar el *smartphone* al robot, aprovechando la cámara del dispositivo móvil como un sensor adicional, el cual a su véz actúa como un mecanismo de control autónomo.

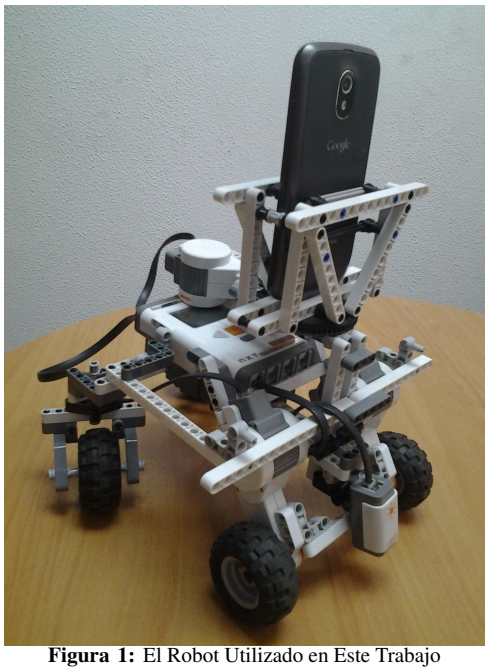


Figura 1: El Robot Utilizado en Este Trabajo

3. Marco Tecnológico

En esta sección se describen las diferentes herramientas y dispositivos utilizados durante el desarrollo de este trabajo de investigación. Estas descripciones son dadas fuera del contexto en el se utilizaron dichas herramientas, centrándose únicamente en las especificaciones técnicas de cada una. El uso específico de cada herramienta se expone con más detalle en la Sección 5.

3.1. Herramientas de Hardware

El sistema diseñado y desarrollado en este trabajo de investigación abarca diversos dispositivos de hardware para realizar sus funciones de control del robot y despliegue de objetos virtuales. Entre los dispositivos utilizados se encuentran los robots LEGO Mindstorms NXT, un punto de acceso inalámbrico y varios dispositivos basados en el sistema operativo Android, inluyendo *smartphones*, *tablets* y la consola de video-juegos OUYA, los cuales fueron escogidos considerando las prestaciones y costo de los mismos.

a) El Robot LEGO Mindstorms NXT: El kit Mindstorms NXT es la segunda iteración del paquete Mindstorms disponible comercialmente desde julio de 2006, con una actualización llamada NXT 2.0 disponible desde agosto del 2009. Este kit consiste en un conjunto de 577 piezas entre las cuales se encuentran el *brick* controlador, los sensores, un CD con software asociado y *stickers*. Este kit fue descontinuado comercialmente en el año 2013 con la introducción del robot LEGO Mindstorms EV3.

El *brick* está compuesto por un procesador ARM Atmel de 32 bits que funciona a 45 MHz. Este procesador dispone de 64 KB de memoria RAM, y 256 KB de memoria flash para almacenamiento persistente [17]. Adicionalmente el *brick* contiene un coprocesador AVR Atmel de 8 MHz. Se puede establecer comunicación con el *brick* utilizando ya sea la interfaz USB 2.0 o Bluetooth. Para lo segundo el *brick* dispone de un chip Bluecore 4 que implementa el perfil de puerto serial de Bluetooth 2.0 [18]. El kit producido por *The LEGO Group* incluye 4 sensores: un sensor táctil, un sensor de sonido, un sensor ultrasónico y un sensor puntual monocromático de luz.

Los robots Mindstorms NXT pueden ser programados usando una gran cantidad de tecnologías y lenguajes de programación. El entorno básico es provisto por *The LEGO Group* con el kit y consiste en un *IDE* (*Integrated Development Environmet* - Entorno de Desarrollo Integrado) visual, llamado NXT-G. Con este entorno los robots se programan combinando series de bloques que se ejecutan linealmente sobre carriles paralelos. Estos bloques controlan los distintos componentes electrónicos del robot y pueden combinarse utilizando diferentes estructuras de control como lo son condicionales y ciclos.

De manera alternativa, dada la calidad abierta del hardware y el software de los kits Mindstorms NXT, existe una gran variedad de entornos de desarrollo extraoficiales. Algunos de los más conocidos son el proyecto LejOS[5], que suplanta al firmware estándar de los *bricks* NXT y permite ejecutar programas escritos en un subconjunto del lenguaje Java *Micro Edition* sobre el robot.

La configuración del robot utilizada en este trabajo puede observarse en la Figura 1. El robot fue diseñado con un sistema de locomoción diferencial, es decir, de manera que sus ruedas motoras sean independientes una de la otra, incorporando adicionalmente dos ruedas locas que le dan estabilidad. Así mismo, se incorporó una jaula controlada por un servo-motor la cual sirve para sujetar un *smartphone*, el cual es utilizado

[5]http://lejos.sourceforge.net

Tabla I: Dispositivos Móviles Utilizados

	G. Tab 2	G. Nexus	G. S4	Pantech Burst
Procesador	Cortex A9 1 Ghz	Cortex A15 1.2 Ghz	Cortex A9 1.6 Ghz	Snapdragon 1.5 Ghz
GPU	PowerVR	PowerVR	PowerVR	Adreno
Memoria	1 GB	1 GB	2 GB	1 GB
Cámara	3.5 MP	5 MP	13 MP	5 MP
Bluetooth	v3.0	v3.0	v4.0 + LE	v3.0
WLAN	b/g/n	b/g/n	b/g/n/ac	b/g/n

como sensor de visión, tal como se describe en la Sección 4.1. Este diseño está basado en el trabajo desarrollado por Carlos Tovar y Paolo Tosiani [19].

b) Dispositivos Móviles: Para el desarrollo de la implementación de referencia se utilizaron dos *tablets* Samsung Galaxy Tab 2, una de 7 pulgadas y otra de 10.1 pulgadas, así como un *smartphone* Samsung Galaxy Nexus. Para las pruebas se utilizaron adicionalmente un *smartphone* Samsung Galaxy S4 y un *smartphone* Pantech Burst, así como una consola de videojuegos OUYA (que se describe más adelante en esta sección). La Tabla I resume las características de hardware de los dispositivos utilizados. Los datos de la Tabla I fueron recuperados de http://www.gsmarena.com. La columna Galaxy Tab 2 en la Tabla I incluye tanto la Galaxy Tab 2 7.0 como la 10.1 dado que la única diferencia relevante entre ambas es el tamaño de sus pantallas.

c) La Consola de Juegos OUYA: La consola de videojuegos OUYA es un dispositivo basado en el sistema operativo Android 4.1 desarrollada por la empresa OUYA Inc[6]. Este dispositivo actua como una consola de videojuegos de sobremesa completamente modificable y programable, diseñada principalmente para desarrolladores de videojuegos independientes y aficionados. La consola esta basada en el procesador Tegra 3 de Nvidia el cual implementa el API definido por el estandar OpenGL ES 2.0. El procesador funciona a 2 Ghz de frecuencia e incluye 8 Gigabytes de almacenamiento interno y 1 Gigabyte de memoria RAM. Esta consola cuenta con conectividad 802.11 n y Bluetooth 2.0, así como un puerto Fast Ethernet, un puerto USB 2, un puerto mini USB y salida de video HDMI (*High Definition Multimedia Interface* - Interfaz Multimedia de Alta Definición) a 1080p.

La consola es manipulada por un control inalámbrico que se conecta con la misma utilizando Bluetooth. Este control sigue el modelo típico para *gamepads*, contando con 4 botones identificados por letras (O, U, Y y A), un *pad* direccional, dos palancas analógicas, dos gatillos digitales, dos gatillos analógicos y un botón de inicio. Adicionalmente el control incluye un pequeño panel táctil que funciona como un *trackpad* el cual puede utilizarse para emular controles de pantalla táctil.

d) El Punto de Acceso Inalámbrico: Durante el desarrollo de este trabajo de investigación se utilizó un punto de acceso inalámbrico D-LINK modelo 634. Este es un punto de acceso doméstico con soporte para IEEE 802.11 b y g. Este modelo incluye soporte para la tecnología MIMO (*Multiple Input and Multiple Output* - Entrada Múltiple y Salida Multiple) con dos antenas, además de soporte para el protocolo propietario WiFi Super G. El punto de acceso incluye 4 puertos *Fast Ethernet* y tiene un alcance de aproximadamente 100 metros en la interfaz inalámbrica.

[6]https://www.ouya.tv

3.2. Herramientas de Software

El desarrollo de la parte de software del sistema de control planteado en este trabajo hizo uso de múltiples bibliotecas y componentes, algunos de los cuales son necesarios para trabajar con el hardware antes descrito y otros que fueron escogidos por diversas razones que se exponen a continuacón. Casi la totalidad de la implementación de referencia fue desarrollada con el lenguaje de programación Java, el cual es el lenguaje de programación principal utilizado por el sistema operativo Android. El resto de la aplicación fue desarrollada en el lenguaje C++, en particular las funcionalidades relacionadas con procesamiento digital de imágenes.

a) OpenCV: Este es el acrónimo utilizado por el proyecto *Open Computer Vision* (Visión por Computador Abierta) llevado a cabo por las empresas Willow Garage e Itzees para desarrollar una biblioteca eficiente y sofisticada de procesamiento digital de imágenes. OpenCV es utilizada para manipular y obtener información de imágenes ya sean estáticas u obtenidas a partir de video. Esta biblioteca cuenta con una amplia gama de funciones que permiten trabajar a distintos niveles de abstracción, ya sea con operaciones aritméticas sobre imágenes y filtros de convolución básicos, hasta distintos algoritmos de detección de características utilizando clasificadores y máquinas vectoriales de soporte (del inglés *support vector machines*).

En este trabajo se utilizó esta biblioteca para desarrollar las funcionalidades referentes a la calibración de cámaras de video y detección de marcadores para Realidad Aumentada. Fue escogida por tres razones fundamentales: la primera, la biblioteca es de software libre y gratuito; la segunda, esta biblioteca es reconocida por ser altamente eficiente; y la tercera, posee soporte para múltiples plataformas de hardware y sistemas operativos, en particular para el sistema operativo Android, que como se mencionó anteriormente fue la plataforma escogida para el desarrollo del proyecto.

La principal ventaja del soporte multiplataforma de OpenCV es que utiliza exactamente la misma interfaz de programación de aplicaciones independientemente del sistema operativo o hardware para el que se programe. Esto permitió desarrollar y probar los distintos algoritmos de calibración de cámaras, detección y decodificación de marcadores en una computadora de escritorio, facilitando enormemente la depuración de los mismos, con la seguridad de que el mismo código fuente podría luego ser portado directamente al sistema operativo Android sin tener que aplicar modificaciónes posteriores.

b) LibGDX: A pesar de su nombre LibGDX es un *framework* más que una biblioteca y tiene como objetivo el simplificar el desarrollo de videojuegos múltiplataforma en

el lenguaje Java. Es desarrollado por el estudio de desarrollo de videojuegos independiente Badlogic Games[7]. LibGDX es una agrupación de un gran conjunto de diferentes bibliotecas de desarrollo de aplicaciones gráficas. Entre las distintas bibliotecas incorporadas por LibGDX se encuentran:

- **OpenGL**: para despliegue de gráficos 2D y 3D. Dependiendo de la plataforma en la que se ejecute la aplicación desarrollada LibGDX se encarga de utilizar ya sea OpenGL estándar u OpenGL ES;
- **Box2D**: para realizar simulaciones físicas en dos dimensiones;
- **Bullet**: para simulaciones físicas en tres dimensiones;
- **FreeType**: para la manipulación de fuentes y despliegue de textos;
- **OpenAL, Vorbis y Mpg 123**: para reproducción de sonido;
- **GWT, LWJGL y RoboVM**: para las funcionalidades dependientes del sistema operativo.

Se utilizó LibGDX en este trabajo de investigación principalmente por las simplificaciones que provee a la hora de trabajar con OpenGL sobre los dispositivos basados en el sistema operativo Android. Adicionalmente, la naturaleza multiplataforma de LibGDX hace factible la posibilidad de reutilizar la misma base de código en múltiples plataformas de hardware, lo que nos permitiría portar la implementación desarrollada en este trabajo a nuevos dispositivos de control con considerable facilidad.

c) Otras Herramientas de Software: Aparte de las herramientas ya mencionadas, en este trabajo de investigación se hizo uso de otras bibliotecas adicionales para el desarrollo de características puntuales de la implementación de referencia. Entre estas bibliotecas resaltan las siguientes:

- **Artemis *Entity-System Framework***: para la implementación eficiente del patrón de diseño Entidad-Componente-Sistema;
- **Java *Universal Tween Engine***: para implementar efectos de transición entre las distintas vistas de la solución, mediante la interpolación de atributos numéricos en objetos de Java.

4. Sistema de Control Basado en Realidad Aumentada

En esta sección se presenta el diseño del sistema de control de robots propuesto. La descripción del mismo esta dada siguiendo un esquema *top-down*, comenzando con el diseño general y enlaces de comunicación de los componentes de hardware y con su correspondiente modelo de casos de uso, para luego detallar cada una de las funcionalidades planteadas, junto a los módulos de software que las implementan.

4.1. Diseño General del Sistema

El sistema de control para robots móviles basado en realidad aumentada diseñado en el transcurso de este trabajo de investigación se compone por 4 elementos de hardware: el robot móvil, un dispositivo programable con cámara de video incorporada, un dispositivo de control centralizado y un punto de acceso inalámbrico. Estos tres dispositivos deben poseer la capacidad de comunicarse entre sí de forma inalámbrica ya sea utilizando enlaces Bluetooth o el punto de acceso inalámbrico. La Figura 2 ilustra estos componentes y la relación entre ellos. Este sistema está diseñado alrededor de un robot móvil programable con conectividad inalámbrica. Este robot debe poseer un esquema de locomoción diferencial, y debe incluuir algún mecanismo que permita incorporarle físicamente el dispositivo con cámara como un sensor adicional. Como requisitos mínimos, el robot debe poseer dos motores separados para su locomoción (un motor derecho y otro izquierdo), aunque se puede incorporar un tercer motor que permita girar el dispositivo con cámara de forma que el robot pueda "ver" en una dirección diferente a su dirección de movimiento.

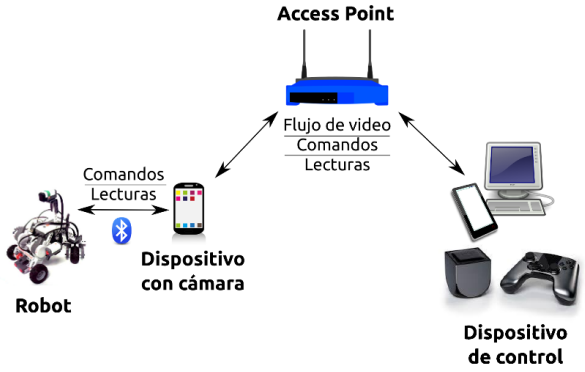


Figura 2: Diseño Arquitectónico del Sistema Propuesto

El dispositivo con cámara mostrado en la Figura 2 corresponde con cualquier dispositivo programable capaz de grabar video o tomar fotografías con rapidez. Este dispositivo además actúa como un puente entre el robot y el dispositivo de control, siendo este último el dispositivo utilizado directamente por el usuario para controlar al robot. Este dispositivo de control puede ser cualquier dispositivo programable capaz de recibir interacción por parte del usuario, conectarse con el dispositivo con cámara y desplegar imágenes a color por pantalla; ejemplo de esto pueden ser las *tablets* y los *smartphones*, las computadoras de escritorio o incluso las consolas de videojuegos.

4.2. Modelo de Casos de Uso

Para establecer los casos de uso del sistema propuesto se definen como posibles actores al usuario encargado de controlar al robot, que llamamos jugador y al diseñador de escenarios. El jugador es el usuario que se encarga de ejecutar un escenario que haya sido desarrollado por el diseñador de escenarios, utilizando las diversas interacciones propuestas por el sistema para completar algún objetivo específico. Por su parte el diseñador de escenarios es el encargado de diseñar y programar los escenarios que podrá utilizar el usuario jugador.

[7]http://www.badlogicgames.com

Para el caso del usuario jugador se establecen tres actividades que permiten llevar a cabo un escenario concreto. La primera es la calibración de la cámara que tiene que realizarse para asegurar un despliegue de objetos virtuales correcto. La segunda actividad es la ejecución del escenario manual del robot que representa el juego en sí. El escenario de control manual debe permitir al usuario el manipular al robot y a un "brazo mecánico" virtual mediante controles que se describen más adelante, de forma que utilizando ambos el usuario tenga la posibilidad de cumplir el objetivo propuesto por el escenario. Finalmente el usuario debe tener la posibilidad de ejecutar un algoritmo de control autónomo para el robot el cual dependerá del escenario que se esté ejecutando.

Por su parte el diseñador de escenarios tiene la posibilidad de desarrollar los componentes que ejecuta el jugador, siendo opcional aunque recomendado el implementar las pantallas de resumen que se le muestran al jugador al completar cada módulo de juego.

4.3. Control del Robot

El mecanismo de interacción con el robot depende de la naturaleza del dispositivo de control. Por ejemplo, en una *tablet* el control debería realizarse por medio de la pantalla táctil de la misma, o en el caso de una computadora de escritorio por medio del teclado y el ratón.

Independientemente del mecanismo físico mediante el cual se manipule al robot, este debe soportar los siguientes controles:

- Avanzar y retroceder cada motor por separado;
- Mover el brazo virtual;
- Retornar el motor del dispositivo con cámara a su posición inicial.

El último control enunciado es opcional, pudiendo ser ignorado por el robot en caso de no poseer el tercer motor para el dispositivo con cámara mencionado anteriormente.

4.4. El Brazo Virtual

El brazo virtual es una representación de un efector lógico añadido al robot de forma que este pueda interaccionar con los objetos virtuales del escenario de juego. Este brazo debe poder desplazarse libremente en un plano paralelo a la pantalla del dispositivo de control, y además debe poder avanzar y retroceder en un vector perpendicular a la misma.

4.5. Escenarios Jugables y Despliegue de Objetos Virtuales

Un escenario consiste en un conjunto de reglas que definen un objetivo, el cual debe cumplir el jugador utilizando las herramientas de control descritas anteriormente. Como mínimo un escenario debe definir un conjunto de objetos virtuales con los cuales puede interaccionar el jugador y las condiciones para completar el escenario, ya sea en victoria o derrota. Los objetos virtuales deben consistir en modelos 3D los cuales deben ser desplegados por pantalla alineados con marcadores. La información que codifican los marcadores es responsabilidad de cada implementación. La interacción de los objetos virtuales con el robot se debe basar en la detección de colisiones de dichos objetos con el brazo virtual.

5. Desarrollo del Sistema

En esta sección se describe en detalle una serie de aplicaciones desarrolladas que implementan el diseño del sistema descrito en la sección anterior. Esta implementación abarca tres aplicaciones: dos para el sistema operativo Android y una para el robot LEGO Mindstorms NXT. Las aplicaciones desarrolladas para dispositivos basados en el sistema operativo Android son llamadas NXTAR-cam y NXTAR-core e implementan los módulos del dispositivo de visión y el dispositivo de control respectivamente. La aplicación para el robot es llamada NXTAR-bot y tiene como objetivo el aplicar las instrucciones de control que se generan desde NXTAR-core.

Las aplicaciones desarrolladas son software libre[8] publicado bajo la licencia Apache 2.0[9]. Las aplicaciones para el sistema operativo Android tienen soporte para dispositivos móviles como *tablets* y *smartphones*. En el caso particular de la aplicación para el dispositivo de control (NXTAR-core) se posee soporte adicional para la consola de videojuegos OUYA.

5.1. Organización de los Módulos

a) NXTAR-*core:* Este módulo se divide en 3 subpaquetes principales: interfaces, escenarios y estados. El paquete de interfaces define los puntos de acceso a través de los cuales el núcleo de la aplicación y el *frontend* para Android pueden comunicarse. En particular este paquete se utiliza para abstraer las funcionalidades del sistema operativo de forma que el núcleo pueda tener acceso a las mismas independientemente de la naturaleza del dispositivo de control. Así mismo, este paquete define una interfaz genérica que permite realizar el procesamiento de las imágenes obtenidas por el dispositivo con cámara independientemente de como esten impliementados los algoritmos de visión por computador definidos más adelante en esta sección.

El subpaquete de escenarios de NXTAR-core contiene un conjunto de clases abstractas y una clase estática las cuales definen el escenario de juego disponible al jugador. Las clases abstractas representan los requisitos que debe cumplir todo escenario y deben ser implementadas por el usuario diseñador de escenarios. La clase estática por su parte consiste en un conjunto de apuntadores a las implementaciones de las clases abstractas definidas anteriormente. Los escenarios en sí se definen dentro de un subpaquete diferente.

El subpaquete de estados de NXTAR-core contiene las implementaciones de los estados, valga la redundancia, que componen la máquina de estados finitos de la aplicación y del escenario. Esta máquina de estados finitos es descrita en detalle más adelante en esta sección.

[8]Los códigos fuente pueden encontrarse en https://github.com/miky-kr5
[9]http://www.apache.org/licenses/LICENSE-2.0

Tabla II: Protocolo de Control del Robot

Bit	Operación
0x01	Control del motor en el puerto A
0x02	Control del motor en el puerto B
0x04	Control del motor en el puerto C
0x08	Dirección del movimiento de los motores
0x10	Regresar el motor en el puerto B a su posición original
0x20	Operación de usuario 1
0x40	Operación de usuario 2
0x80	Operación de usuario 3

b) La Aplicación NXTAR-*bot:* NXTAR-bot es una pequeña aplicación desarrollada en el lenguaje Java para ser ejecutada en robots LEGO Mindstorms™ NXT que posean el firmware del sistema operativo LeJOS. Esta aplicación se encarga de realizar dos únicas funciones, la primera de las cuales es reportar las medidas de los sensores del robot. Este reporte se realiza automáticamente entre las aplicaciones NXTAR-bot y NXTAR-cam, siendo esta segunda aplicación la que se encarga de redirigir las medidas tomadas a la aplicación de control utilizando un enlace de comunicación TCP. La aplicación de control puede luego consultar dichas medidas mediante un esquema de *polling*.

La otra función de esta aplicación es el ejecutar las instrucciones de control generadas por el usuario jugador o por el sistema de control automático del robot. Estas instrucciones toman la forma de un mensaje de dos bytes transmitido vía Bluetooth al robot desde la aplicación NXTAR-cam. El primero de estos bytes codifica la instrucción de control mientras que el segundo representa un parámetro de la misma. Las instrucciones de control son codificadas utilizando los bits del primer byte del mensaje como se indica en la Tabla II.
El byte de parámetro se utiliza actualmente para indicar la potencia que se debe aplicar a los motores cuando se incluyen instrucciones de control en el rango de bits 0x01 a 0x04. Las demás instrucciones utilizan una potencia constante predefinida en 50 por ciento. La instrucción de dirección (bit 0x08) se utiliza como un parámetro adicional de las tres instrucciones anteriores, indicando que el motor debe avanzar cuando el bit está encendido y retroceder cuando está apagado.

Los bits de operaciones de usuario se utilizan para indicarle al robot que debe ejecutar ciertas operaciones que deben ser definidas por el programa de la aplicación que utilice este protocolo. En el caso del módulo NXTAR-bot solo se definió la operación de usuario 1, la cual se utiliza para rotar el motor conectado al puerto B del robot 90 grados en sentido contrario a las agujas del reloj.

Utilizar máscaras de bits de esta manera permite componer instrucciones de control más complejas que las definidas por el protocolo LCP (*LEGO Communication Protocol* - Protocolo de Comunicaciones de LEGO) implementado por el *firmware* estándar incluido con los robots LEGO Mindstorms NXT; siendo posible encender más de un motor por mensaje de control. Este protocolo fue implementado como una biblioteca

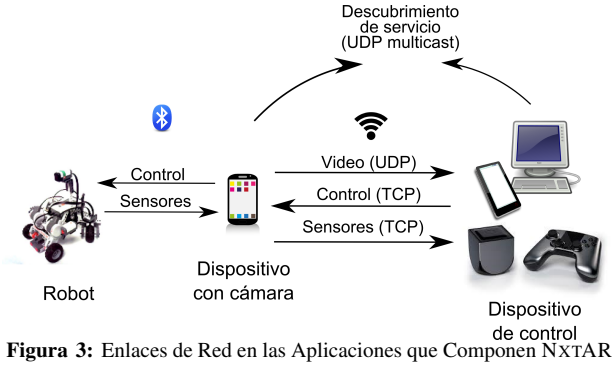


Figura 3: Enlaces de Red en las Aplicaciones que Componen NXTAR

externa a la aplicación NXTAR-bot de manera que pueda ser utilizado por otras aplicaciones.

c) NXTAR-*cam:* Esta es el módulo encargado de implementar el dispositivo con cámara mostrado en el diseño del sistema mostrado en la Figura 2. Su principal funcionalidad es la de capturar y transmitir un flujo de video en tiempo real hacia la aplicación de control, al mismo tiempo que retransmite hacia el robot las instrucciones de control generadas por el usuario.

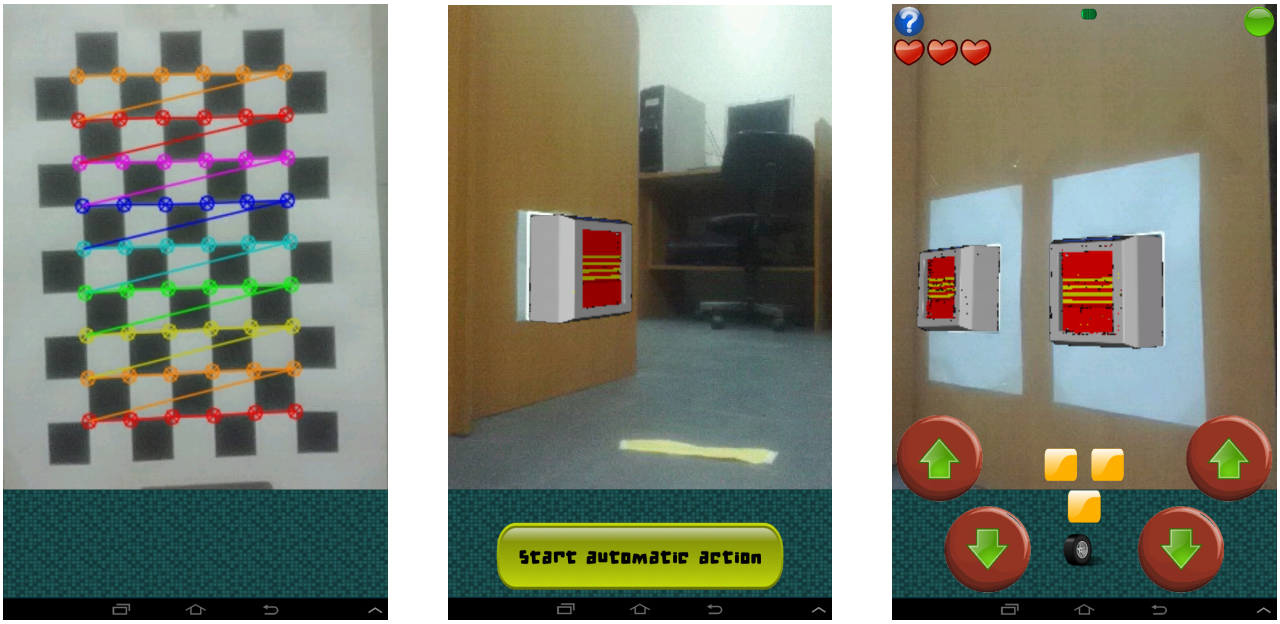
El flujo de video es capturado por la cámara del dispositivo a una tasa de cuadros por segundo definida por el sistema operativo Android dependiendo de las posibilidades del sensor de la misma. La captura se realiza cuadro a cuadro los cuales son colocados por el hilo principal de la aplicación, que es el encargado de realizar la captura, dentro de un objeto monitor de donde el hilo de transmisión los obtiene para enviarlos a la aplicación de control. El monitor implementa un esquema de doble *buffer* para que los hilos no tengan que funcionar a la rapidez del hilo más lento. Un esquema similar se utilizó en el lado receptor. La transmisión del video se realiza utilizando datagramas UDP sin esquemas particulares para control de flujo o corrección de errores.

Las instrucciones de control son enviadas directamente de la aplicación de control a NXTAR-cam utilizando enlaces TCP. Al ser recibidas son almacenadas en una cola con capacidad para 10 elementos para luego ser codificadas utilizando el esquema de máscaras de bits explicado anteriormente y finalmente ser transmitidas al robot con un enlace Bluetooth.

5.2. Comunicación Entre los Módulos

Para interconectar los distintos módulos entre sí se sigue el siguiente esquema:

1. Primero se inicia la aplicación NXTAR-bot en el robot y se realizan los pasos para calibrar el sensor de luz. Una vez listo esto la aplicación se coloca en modo de espera por una conexión Bluetooth;
2. Se inicia la aplicación NXTAR-core (la aplicación de control), la cual comienza a transmitir paquetes multicast anunciándose a la red;
3. NXTAR-cam se conecta con el robot. El robot debe estar emparejado de antemano con el dispositivo de captura de



(a) (b) (c)

Figura 4: Los Estados Principales de NXTAR-core en *Tablets*: (a) Calibración de la Cámara (b) Acción Automática (c) Control Manual

video utilizando las funcionalidades de comunicación por Bluetooth del sistema operativo Android;

4. Finalmente se indica a NXTAR-cam que busque al dispositivo de control utilizando un sencillo protocolo *ad hoc* de descubrimiento de servicios. La conexión entre NXTAR-cam y NXTAR-core se realiza de forma automática.

La Figura 3 ilustra los enlaces de comunicación utilizados por los tres módulos desarrollados.

5.3. Patrones de Diseño en la Aplicación de Control

NXTAR-core hace uso extenso de dos patrones de diseño. Estos patrones son los llamados *state pattern* (patrón de estados) y patrón Entidad-Componente-Sistema.

a) State Pattern: El *state pattern* es un patrón de diseño que consiste en modelar las distintas funcionalidades de una aplicación como una máquina de estados finitos [20]. La aplicación desarrollada define los siguientes siete (7) estados, los cuales coinciden con las diferentes vistas que posee la misma:

- Menú principal;
- Calibración de la cámara;
- En juego;
- Información del juego;
- Resumen del juego;
- Acción automática;
- Resumen de la acción automática.

Estos siete (7) estados a su vez se pueden agrupar en cuatro (4) macro-estados: menú principal, calibración, juego y acción automática. Como se puede deducir de su nombre, el estado menú principal se encarga de mostrar al usuario las distintas opciones que puede realizar al iniciar la aplicación. El estado de calibración se encarga de realizar las acciones de calibración de la cámara del dispositivo con cámara mostrado en el diseño del sistema. El macro-estado de juego o de control manual, agrupa los tres estados que controlan el desarrollo del escenario y permiten al usuario el controlar al robot directamente. Por último el macro-estado de acción automática permite que el robot realice acciones preprogramadas de manera autónoma.

El estado de calibración, que puede observarse en la Figura 4a permite al usuario obtener los parámetros de la cámara necesarios para poder realizar un despliegue correcto de los objetos virtuales sobre las imágenes obtenidas del flujo de video.

El proceso de calibración de la cámara consiste en apuntar la misma hacia un patrón impreso similar a un tablero de ajedrez como puede apreciarse en la Figura 4b. La aplicación captura automáticamente diez (10) muestras de dicho patrón y las utiliza para calcular los parámetros mencionados anteriormente. El patrón de calibración debe obligatoriamente poseer cincuenta y cuatro (54) puntos de intersección divididos en nueve (9) filas por seis (6) columnas o viceversa.

Antes de poder pasar a los macro-estados de control manual o acción automática es obligatorio que el usuario realice primero la calibración de la cámara. Una vez realizada la calibración el usuario puede utilizar cualquiera de los macro-estados de forma indistinta, e incluso tiene la posibilidad de repetir el proceso de calibración de ser necesario.

El estado de acción automática puede verse en la Figura 4b y permite al usuario ejecutar un programa automático predefinido con el robot, el cual muestra un resumen al

culminar su ejecución. Este estado es genérico y necesita que cada escenario implementado incluya su propio código de acción automática y su propia definición de la pantalla de resumen.

Al igual que el estado de acción automática el estado de control manual, visible en la Figura 4c, también es genérico y necesita que cada escenario implemente un conjunto de entidades y sistemas de procesamiento definidos con más detalle en la Sección b). Este estado permite al usuario jugar el escenario programado en la aplicación utilizando diferentes modos de control para manipular al robot dependiendo de la plataforma en la que se ejecute la aplicación. La versión implementada incluye soporte para control por pantalla táctil, *gamepads* y teclado con mouse. Adicionalmente este estado permite manipular el brazo virtual mencionado anteriormente, el cual puede utilizarse para interaccionar con los diferentes objetos virtuales que implementa el escenario.

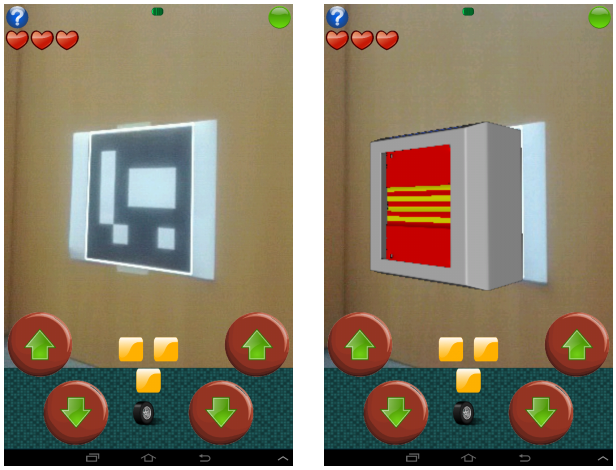
b) Patrón Entidad-Componente-Sistema: El patrón Entidad-Componente-Sistema, también conocido como Entidad-Componente [21], es un patrón de diseño arquitectónico que trata de minimizar el uso de relaciones de herencia entre las clases en un sistema de software orientado a objetos, sustituyéndolas con relaciones indirectas entre clases sencillas llamadas componentes las cuales se agrupan dentro de unidades lógicas llamadas entidades.

El elemento fundamental de este patrón de diseño es el componente, el cual se representa como una clase sencilla únicamente compuesta por atributos públicos, o a lo sumo por un conjunto de métodos *get* y *set* para los distintos atributos. Un componente se encarga de representar una única propiedad que puede poseer una entidad, por ejemplo la posición de la entidad en el espacio, o un mallado que puede utilizarse para desplegar esa entidad en la pantalla. Entonces se puede definir a una entidad como una agrupación de componentes. Las entidades son luego procesadas por los sistemas, los cuales se encargan de implementar la lógica de alguna parte de la aplicación.

El sistema NxtAR hace uso del patrón Entidad-Componente-Sistema principalmente dentro del estado de control manual, separando las funcionalidades del sistema de control en diez (10) sistemas y doce (12) componentes básicos. De estos diez (10) sistemas ocho (8) son independientes del escenario de juego implementado; así mismo, los doce (12) componentes son independientes del escenario, aunque cada escenario tiene la posibilidad de definir componentes adicionales según sea necesario. No existen entidades por defecto en la implementación, siendo cada escenario responsable de crear todas las entidades que necesite. Otros estados de la aplicación también pueden hacer uso de los sistemas, activándolos o desactivándolos según sea necesario.

Los ocho (8) sistemas generales son los siguientes, listados en el orden en que son creados y procesados:

- Posicionamiento de objetos con marcador asociado;



(a) **(b)**

Figura 5: Colocación de Objetos Virtuales sobre un Marcador: (a) El Marcardor Como es Percibido por la Cámara (b) El Objeto Virtual Colocado y Alineado Sobre el Marcador

- Posicionamiento del brazo virtual del robot;
- Aplicación de transformaciones geométricas (escalamiento, rotación y traslación);
- Aplicación de animaciones;
- Detección de colisiones;
- Despliegue de objetos con marcadores;
- Despliegue del brazo del robot;
- Despliegue de efectos especiales.

Adicionalmente se definen dos sistemas que deben ser implementados por cada escenario. Estos son el sistema de lógica de juego y el sistema de procesamiento del jugador. El primer sistema se debe encargar de evaluar las reglas de juego que definen el escenario, modificando las distintas entidades del mismo dependiendo de las interacciones que detecten los sistemas anteriores, así como también se encarga de generar los distintos efectos especiales que pueda utilizar el escenario. Por su parte el sistema de procesamiento del jugador se debe encargar de evaluar las condiciones que determinan cuando el jugador completa el escenario ya sea de forma satisfactoria o insatisfactoria.

5.4. Mecanismo de Realidad Aumentada

El mecanismo de Realidad Aumentada utilizado en la implementación de referencia se basa en detección de marcadores. Este es un procedimiento que consiste en tres operaciones: la detección visual del marcador, la descodificación del mismo y la determinación de la transformación geométrica del marcador en el espacio virtual. Estas tres operaciones son llevadas a cabo por la biblioteca OpenCV y se basan en el algoritmo planteado en el Capítulo 2 del libro *Mastering OpenCV with practical computer vision projects* (Dominando OpenCV con proyectos prácticos de visión por computador) de Daniel Lelis Baggio [22]. El funcionamiento del algoritmo de detección y decodificación de marcadores puede verse en el Algoritmo 1.

```
Datos: Imagen i
Resultado: Lista de pares (Posición[4], código) por cada marcador m
           detectado.
Aplicar umbral adaptativo a i;
Identificar todos los contornos de la imágen umbralizada;
Aproximar contornos con secuencias de segmentos;
mientras Hay una secuencia de segmentos c por procesar hacer
    si c tiene exactamente cuatro aristas entonces
        Tomar subregión s de i definida por c;
        Deformar s para que tome forma cuadrada;
        Dividir s en una matriz ms de 7x7 subregiones;
        si Filas y columnas laterales de ms son mayormente negras
        entonces
            Tomar matriz mi de 5x5 interna de ms;
            mientras Haya rotaciones r de 90 grados de mi por procesar
            hacer
                si Código de bloques q decodifica mi correctamente
                entonces
                    Acumular par (p, q) para este marcador, donde p son
                    las posiciones en píxeles de las aristas de s;
                en otro caso
                    Descartar mi;
                fin
            fin
        en otro caso
            Descartar ms;
        fin
    en otro caso
        Descartar contorno;
    fin
fin
```

Algoritmo 1: Algoritmo de Detección y Decodificación de Marcadores

Los marcadores utilizados en este proyecto son los mismos definidos en el libro de Daniel Lelis Baggio [22]. Estos marcadores son imágenes impresas, similares a la que se puede observar en la Figura 5a. Los marcadores son utilizados para codificar indices enteros los cuales se utilizan para determinar cual objeto virtual debe colocarse sobre el mismo al momento del despliegue final. Un ejemplo del aspecto final de este proceso puede verse en la Figura 4c y la Figura 5b.

5.5. Bomb Game, el Escenario de Demostración

NXTAR incluye un escenario sencillo como demostración de las capacidades del sistema titulado *Bomb Game*. Este juego, como su nombre sugiere, consiste en desarmar una serie de artefactos explosivos o bombas, cada una de las cuales posee un mecanismo diferente para ser desactivada. El escenario incluye 3 tipos de bomba diferentes e implementa el modo de acción automática.

El modo de acción automática de *Bomb Game* consiste en un recorrido lineal sencillo en el cual el robot, avanzando en línea recta, trata de detectar si existen marcadores a su izquierda cuando pasa cerca de una marca en el suelo que sirve de indicador. Al detectar tantos indicadores como bombas han sido configuradas en el escenario el robot se detiene y se le muestra un resumen al usuario sobre cuántas bombas virtuales fueron detectadas y cuántas se esperaba encontrar[10].

[10]https://www.youtube.com/watch?v=hM_w54X8OKU

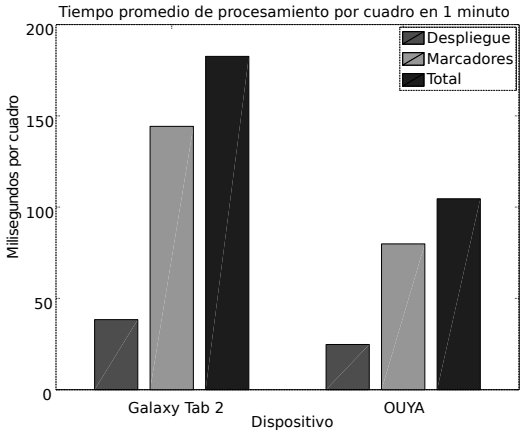


Figura 6: Estudio del Tiempo de Procesamiento Promedio por Cuadro Durante un (1) Minuto

En el modo de control manual se implementó el juego de desactivación de bombas, en el cual el usuario debe manipular al robot dentro del entorno buscando los distintos marcadores que representan las bombas y desactivarlas según sus diferentes mecanismos utilizando el brazo virtual con los controles provistos para tal fin[11].

Como se mencionó, el juego implementa tres bombas. La primera es la bomba de cables, la cual consiste en un conjunto de cables de colores de entre los cuales el usuario debe cortar uno (el azul) posicionando el brazo virtual para tal fin. Por su parte la bomba de combinación posee cuatro botones de colores los cuales deben ser presionados en un orden predeterminado (azul, rojo, negro y luego verde). Por último la bomba de nivel posee un único botón el cual debe ser presionado únicamente cuando el indicador de inclinación del dispositivo está en verde. En el caso de que la aplicación sea ejecutada en dispositivos como la consola OUYA, la cual no posee sensores que le permitan determinar su orientación, este tipo de bomba puede ser desactivada sencillamente con presionar su botón.

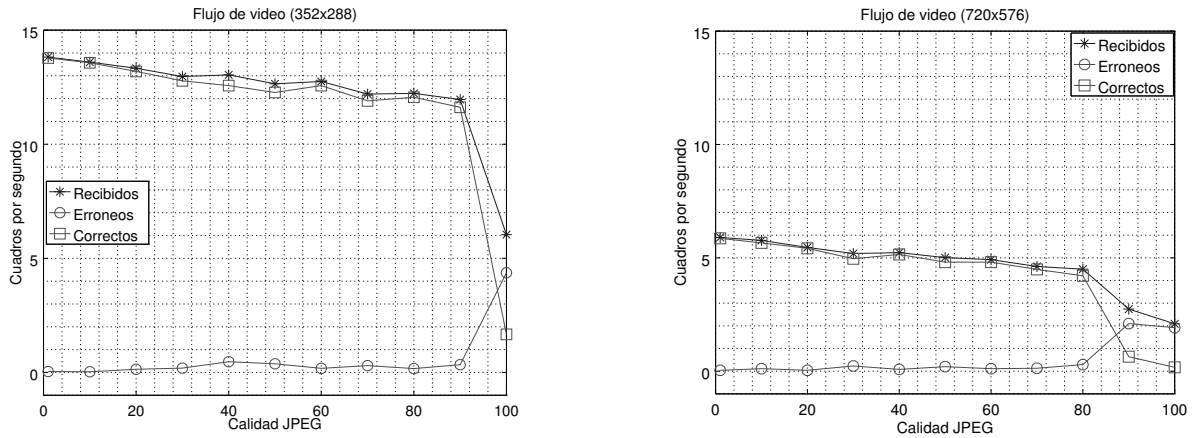
6. PRUEBAS Y RESULTADOS

En esta sección se describen las distintas pruebas a las que fue sometida la implementación de referencia. Estas pruebas se enfocaron principalmente en los aspectos más resaltantes de las aplicaciones desarrolladas, en particular lo referente a la transmisión de video y el despliegue de objetos virtuales.

6.1. Perfilado del Procesamiento en el Estado de Control Manual

Esta prueba fue realizada con la finalidad de determinar el tiempo consumido por las diferentes acciones llevadas a cabo durante el procesamiento de un cuadro en el estado de control manual. Estas acciones se dividieron en dos categorías. La primera es todo lo referente al algoritmo de detección de marcadores y la segunda es el resto del procesamiento del

[11]https://www.youtube.com/watch?v=S_mK-1KjqG4



(a) **(b)**

Figura 7: Tasas de Transmisión de Video en el Dispositivo de Control: (a) Para Imágenes de 352x288 Píxeles (b) Para Imágenes de 720x576 Píxeles

cuadro, incluyendo la lógica del juego y el despliegue de objetos 3D.

La prueba consistió en la medición del tiempo de procesamiento promedio por cuadro durante un minuto, desglosado en las categorías antes mencionadas. Se probaron dos escenarios, uno utilizando la Galaxy Tab 2 como dispositivo de control y otro utilizando la consola OUYA. Ambos escenarios fueron repetidos treinta (30) veces cada uno y luego promediados con la intención de mitigar el error estadístico. Todas las medidas fueron tomadas en milisegundos de procesamiento por cuadro. En ambos escenarios el dispositivo con cámara fue un *smartphone* Galaxy Nexus y el video fue fijado a una resolución de 352 por 288 píxeles, utilizando compresión JPEG con el parámetro de calidad fijo en noventa (90).

a) Resultados y Análisis: El primer análisis sobre los datos capturados fue el comparar el desempeño entre los dos (2) dispositivos utilizados, análisis que puede observarse en el gráfico de la Figura 6. Lo más relevante en este gráfico es el hecho de que el tiempo de procesamiento dedicado a la detección de marcadores consume la mayor parte del tiempo total dedicado a cada cuadro en ambos dispositivos. Así mismo se puede observar que en ambos dispositivos el tiempo total dedicado al procesamiento de cuadros es considerablemente alto, siendo en promedio aproximadamente 180 milisegundos por cuadro en la Galaxy Tab 2 y 100 milisegundos por cuadro en la consola OUYA. Esto permite procesar entre unos $5{,}5$ y 10 cuadros por segundo en cada dispositivo respectivamente.

Luego se calculó el porcentaje de tiempo consumido por ambas categorías de procesamiento. Se obtuvo como resultado que más del $75\,\%$ del tiempo de procesamiento se dedica a la detección de marcadores en ambos dispositivos. De este estudio y del anterior se puede concluir entonces que el algoritmo de detección de marcadores es candidato a ser optimizado, de forma que se pueda obtener una mejor tasa de cuadros por segundo en los dispositivos de control, lo que debería traducirse en mejor experiencia de usuario.

6.2. Tasas de Recepción de Video en el Dispositivo de Control

En esta prueba se midió la tasa de recepción de video para diferentes tamaños de imagen y calidades de compresión JPEG. El objetivo fue determinar el valor óptimo para ambos parámetros. Se probaron 3 escenarios con imágenes de dimensiones ubicadas entre 176x144 píxeles, 352x288 píxeles y 720x576 píxeles, variando la calidad de la compresión JPEG de los cuadros entre 1 y 100 inclusive, en pasos de 10 en 10 para cada escenario. De estos escenarios solo se conservaron los dos últimos dado que la biblioteca OpenCV fallaba al tratar de realizar la calibración de la cámara con las imágenes de tamaño 176x144 píxeles. Cada escenario fue repetido 10 veces por cada calidad de compresión, teniendo cada repetición una duración de 10 segundos y además utilizando un punto de acceso inalámbrico dedicado. Todas las pruebas fueron realizadas con la Galaxy Tab 2.

a) Resultados y Análisis: En la Figura 7 se pueden observar las tasas de transmisión promedio para los dos escenarios probados. Como puede observarse en ambos escenarios la tasa de recepción disminuye a medida de que se incrementa la calidad de la compresión, presentándose la mayor pérdida de cuadros de video cuando dicho parámetro supera el valor de 90. De hecho cuando la compresión alcanza el valor de 100 la pérdida de imágenes es casi absoluta. Otro detalle que resalta es el hecho de que la tasa de recepción disminuye a un poco menos de la mitad en el segundo escenario con respecto al primero.

En base a estos resultados concluimos que los valores óptimos para los parámetros en estudio son el tamaño de imágen de 355 por 288 píxeles y la calidad de compresión en 90. Escogimos estos valores porque de esta forma la tasa de recepción de imágenes de video se aproxima bastante al tiempo de procesamiento del juego en el mejor caso detectado (10 cuadros por segundo). Esto disminuye la cantidad de

imágenes desperdiciadas cuando el procesador del dispositivo de control no es capaz de procesarlas lo suficientemente rápido.

6.3. Determinación del Punto de Saturación del Video

Esta prueba se diseñó con el objetivo de examinar el comportamiento de la transmisión y recepción del video enviado entre los dispositivos móviles involucrados en el sistema. Dada la ubicuidad de los puntos de acceso inalámbricos en los entornos laborales y caseros se busca determinar la posibilidad de insertar el sistema desarrollado en una red existente. Para realizar este estudio se utilizó el punto de acceso inalámbrico descrito anteriormente como punto intermedio, y la *tablet* Galaxy Tab 2 como dispositivo de control. Se utilizó un enfoque *bottom-up*, midiendo las tasas de recepción y pérdida de cuadros de video por segundo para diferentes escenarios de utilización del canal de transmisión.

Para llevar a cabo la saturación del canal se utilizó el sistema D-ITG (*Distributed Internet Traffic Generator* - Generador de Tráfico para Internet Distribuido) descrito en [23]. En este estudio se generó un tráfico constante utilizando paquetes de 1500 bytes, y variando la tasa de paquetes transmitidos por segundo para saturar el canal con tasas de transmisión de bits que varían entre 0 y 9 Mbps, incrementando la tasa de transmisión de bits en pasos de 1 Mbps en cada escenario. Cada escenario constó de una duración de 10 segundos y fue repetido 10 veces, para luego ser promediado.

a) Resultados y Análisis: Una vez completado el estudio diseñado para esta prueba se obtuvieron los datos que pueden observarse en el gráfico de la Figura 8. Como se puede apreciar, cuando el punto de acceso está dedicado exclusivamente a la transmisión del video los resultados coinciden con los obtenidos en la prueba descrita en la Sección 6.2. Sin embargo, a medida que se comienza a inyectar tráfico en el canal el rendimiento de la transmición comienza a degradarse, obteniendo una tasa de pérdida de cuadros mayor que la tasa de cuadros recibidos correctamente a medida que el tráfico se acerca y eventualmente supera los 8 Mbps.

Tomando como un mínimo aceptable una transmisión de video de 10 cuadros recibidos por segundo[12] (correcta o incorrectamente), podemos establecer el punto de saturación de la transmisión de video alrededor de los 5 Mbps, punto en el cual la tasa de cuadros recibidos correctamente es considerablemente menor a 9 cuadros por segundo, lo cual implica que aproximadamente 2 de cada 10 cuadros por segundo poseen errores que no permiten su decodificación. Más allá de 5 Mbps la tasa de cuadros recibidos independientemente de su estado alcanza niveles muy bajos, a medida que la tasa de cuadros perdidos se hace aún mayor.

7. Conclusiones y Trabajos Futuros

En este trabajo de investigación se diseño y desarrolló un sistema de control para robots móviles sobre el cual es posible

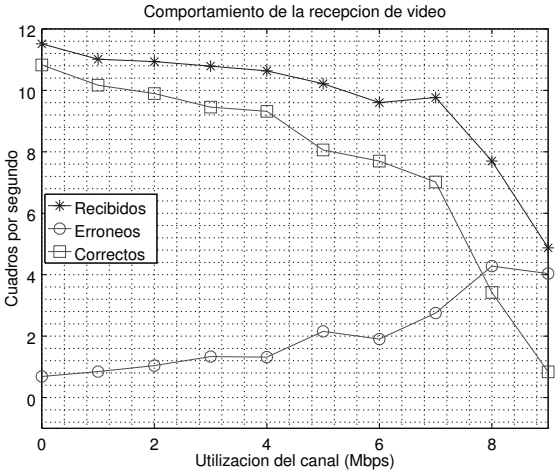


Figura 8: Comportamiento de la Recepción de Video

implementar escenarios que hacen uso de tecnologías de realidad aumentada. El sistema de control fue diseñado para ser genérico, es decir, independiente del hardware utilizado en su implementación, limitándose a plantear un esquema de comunicación entre los dispositivos involucrados, así como un conjunto de posibles casos de uso que se deben cumplir. Este sistema fue implementado luego en varios dispositivos móviles basados en el sistema operativo Android, utilizando un robot móvil LEGO Mindstorms NXT como robot de prueba. Para llevar a cabo este desarrollo se realizó primero un estudio del hardware disponible sobre el cual se realizó el mismo. La implementación fue desarrollada para soportar tanto *tablets* y *smartphones* basados en el sistema operativo Android, así como la consola de videojuegos OUYA funcionando como dispositivos de control del robot.

La principal contribución de este trabajo es la especificación del protocolo mostrado en la Sección 5.1. Este protocolo fue diseñado como una mejora al protocolo LCP diseñado por The LEGO Group, con la intención de minimizar la cantidad y el tamaño de los mensajes necesarios para controlar un robot móvil de locomoción diferencial con a lo sumo 3 motores. Dicho protocolo funciona como un mecanismo de bajo nivel que permite indicarle al robot que encienda o apague múltiples motores utilizando mensajes de 2 bytes de tamaño, entre otras funcionalidades. Este protocolo es independiente del robot a utilizar. Una implementación[13] del mismo para los robots LEGO Mindstorms NXT, utilizando el sistema operativo LejOS, es distribuida por los autores como software libre bajo los términos de la licencia Apache 2.0.

Adicionalmente se diseñó un conjunto de escenarios de prueba con el objetivo de evaluar distintos aspectos del sistema desarrollado. Con estas pruebas se pudo determinar las limitaciones de la transmisión de video, lo que nos permitió establecer diferentes parámetros para garantizar que este proceso no afecte la experiencia del usuario. Así mismo

[12] Una tasa menor a 10 cuadros por segundo se considera que provee una muy mala experiencia [24].

[13] Disponible en https://github.com/miky-kr5/libnxtarcontrol

se pudo determinar que el procesamiento del video es la operación más costosa en cuanto a tiempo de ejecución dentro de la aplicación de control. En el caso de la prueba de saturación de red se determinó que la implementación de referencia puede funcionar correctamente cuando la utilización del canal inalámbrico por parte de aplicaciones externas es menor a 4 Mbps.

La principal limitación de este trabajo radicó en la falta de más dispositivos móviles para utilizar como dispositivos de control, lo que limitó la realización de las pruebas a solo dos dispositivos completamente diferentes.

Tomando como referencia las pruebas realizadas y las limitaciones encontradas durante el desarrollo de este trabajo de investigación se proponen los siguientes trabajos a futuro:

- Mejorar el algoritmo de detección de marcadores para obtener menores tiempos de procesamiento;
- Diseñar e implementar nuevos escenarios de mayor complejidad, enfocados en dominios de aplicación específicos;
- Portar la implementación de referencia a otros dispositivos de control, en particular a computadoras de escritorio, aprovechando las capacidades del *framework* LibGDX;
- Evaluar la factibilidad de extender el sistema para soportar múltiples saltos de red entre los distintos dispositivos involucrados, en particular entre el dispositivo de control y el dispositivo con cámara;
- Evaluar la factibilidad de agregar soporte de realidad aumentada sin marcadores al sistema.

Referencias

[1] M. Kojima, M. Sugimoto, A. Nakamura, M. Tomita, H. Nii, y M. Inami. *Augmented coliseum: An Augmented Game Environment with Small Vehicles.* En Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006. First IEEE International Workshop on, pp. 6, IEEE, 2006.

[2] K. Hosoi, V. Dao, A. Mori, y M. Sugimoto. *Cogame: Manipulation Using a Handheld Projector.* En ACM SIGGRAPH 2007 emerging technologies, pp. 2. ACM, 2007.

[3] K. Hosoi, V. Dao, A. Mori, y M. Sugimoto. *Visicon: a Robot Control Interface for Visualizing Manipulation Using a Handheld Projector.* En Proceedings of the International Conference on Advances in Computer Entertainment Technology, pp. 99-106. ACM, 2007.

[4] T. Xie, L. Xie, L. He, y Y. Zheng. *A General Framework of Augmented Reality Aided Teleoperation* guidance. 2013.

[5] P. Milgram, S. Zhai, D. Drascic, y J. Grodski. *Applications of Augmented Reality for Human-Robot Communication.* En Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93), vol. 3, pp. 1467-1472, 1993.

[6] S. Saha. *Introducción a la Robótica*. Mc Graw-Hill, 2010.

[7] J. Chong, S. Ong, A. Nee, y K. Youcef-Youmi. *Robot Programming Using Augmented Reality: An Interactive Method for Planning Collision-Free paths.* Robotics and Computer-Integrated Manufacturing, vol. 25, no. 3, pp. 689-701, 2009.

[8] H. Fang, S. Ong, y A. Nee. *Interactive Robot Trajectory Planning and Simulation Using Augmented Reality.* Robotics and Computer-Integrated Manufacturing, vol. 28, no. 2, pp. 227-237, 2012.

[9] H. Fang, S. Ong, y A. Nee. *Orientation Planning of Robot End-Effector Using Augmented Reality.* The International Journal of Advanced Manufacturing Technology, vol. 67, no. 9-12, pp. 2033–2049, 2013.

[10] A. Kelly, N. Chan, H. Herman, D. Huber, R. Meyers, P. Rander, R. Warner, J. Ziglar, y E. Capstick. *Real-time Photorealistic Virtualized Reality Interface for Remote Mobile Robot Control.* The International Journal of Robotics Research, vol. 30, no. 3, pp. 384-404, 2011.

[11] J. Guivant, S. Cossell, M. Whitty, y J. Katupitiya. *Internet-Based Operation of Autonomous Robots: The Role of Data Replication, Compression, Bandwidth Allocation and Visualization.* Journal of Field Robotics, vol. 29, no. 5, pp. 793-818, 2012.

[12] F. Okura, Y. Ueda, T. Sato, y N. Yokoya. *Teleoperation of Mobile Robots by Generating Augmented Free-Viewpoint Images.* En Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13), pp. 665-671, 2013.

[13] F. Okura, Y. Ueda, T. Sato, y N. Yokoya. *Free-Viewpoint Mobile Robot Teleoperation Interface Using View-Dependent Geometry and Texture.* ITE Transactions on Media Technology and Applications, vol. 2, no. 1, pp. 82-93, 2014.

[14] S. Green, J. Chase, X. Chen, y M. Billinghurst. *Evaluating the Augmented Reality Human-Robot Collaboration System.* International Journal of Intelligent Systems Technologies and Applications, vol. 8, no. 1, pp. 130-143, 2010.

[15] M. Astor, D. Perez, y M. Villapol. *Visión por Computador Para Robots Mindstorms NXT.* En Memorias de la Primera Conferencia Nacional de Computación, Informática y Sistemas, pp. 138-145. Naiguatá, Venezuela, Octubre 2013.

[16] R. Szabó. *Controlling LEGO Mindstorms NXT Using OpenCV on Android.* http://www.jataka.hu/rics/nxt_android_opencv/index.html.

[17] The LEGO Group. *LEGO Mindstorms NXT Hardware Developer Kit.* 2006.

[18] The LEGO Group. *LEGO Mindstorms NXT Bluetooth Developer Kit.* 2006.

[19] C. Tovar y P. Tosiani. *Diseño, Construcción y Programación de Robots Móviles Para la Captura y Transmisión de Eventos Físicos Vía Bluetooth.* Trabajo de Grado, Universidad Central de Venezuela, Facultad de Ciencias, Escuela de Computación, 2007.

[20] E. Gamma, R. Helm, R. Johnson, y J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994.

[21] T. Alatalo. *An Entity-Component Model for Extensible Virtual Worlds.* Internet Computing, vol. 15, no. 5, pp. 30-37, IEEE, 2011.

[22] D. Baggio. *Mastering OpenCV with Practical Computer Vision Projects.* Packt Publishing Ltd, 2012.

[23] A. Botta, A. Dainotti, y A. Pescapè. *A Tool for the Generation of Realistic Network Workload for Emerging Networking Scenarios.* Computer Networks, vol. 56, no. 15, pp. 3531-3547, 2012.

[24] R. Apteker, J. Fisher, V. Kisimov, y H. Neishlos. *Video Acceptability and Frame Rate.* IEEE multimedia, vol. 2, no. 3, pp. 32-40, IEEE, 1995.