UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA ELECTRÓNICA

Tesis previa a la obtención del título de:

INGENIERO ELECTRÓNICO

TEMA:

"DISEÑO E IMPLEMENTACIÓN DE UN ASISTENTE MÓVIL CON DESPLAZAMIENTO AUTÓNOMO BASADO EN DISPOSITIVOS ANDROID"

AUTOR:

JUAN SEBASTIÁN OCHOA ZAMBRANO

DIRECTOR:

ING. VLADIMIR ROBLES BYKBAEV

Cuenca, Noviembre 2014

Ecuador

CERTIFICACIÓN

Ing. Vladimir Robles Bykbaev

Certifico:

Haber dirigido y revisado prolijamente cada uno de los capítulos de la tesis titulada "DISEÑO E IMPLEMENTACIÓN DE UN ASISTENTE MÓVIL CON DESPLAZAMIENTO AUTÓNOMO BASADO EN DISPOSITIVOS ANDROID", realizada por el estudiante Juan Sebastián Ochoa Zambrano, y por cumplir los requisitos autorizo su presentación.

Cuenca, Noviembre de 2014

Ing. Vladimir Robles Bykbaev

DIRECTOR DE TESIS

DECLARATORIA DE RESPONSABILIDAD

Yo, Juan Sebastián Ochoa Zambrano portador de la cédula de ciudadanía 0302455365, estudiante de la Carrera de Ingeniería Electrónica, certifico que los conceptos desarrollados, así como los criterios vertidos en la totalidad del presente trabajo, son de exclusiva responsabilidad del autor.

A través de la presente declaración cedo los derechos de propiedad intelectual correspondiente a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la Normativa Institucional Vigente.

Cuenca, Noviembre 2014

Juan Sebastián Ochoa Zambrano

DEDICATORIA

Dedico esta tesis a todas aquellas personas quienes piensan que lo “imposible” es una realidad, pues diré, que todo es posible cuando uno lo quiere y lo demuestra con voluntad, sacrificio y perseverancia, no dejemos que nuestros miedos y fracasos sean los obstáculos que nos impidan crecer, sino al contrario, se conviertan en las herramientas que nos motiven aquello. Finalmente de manera muy especial esta tesis va dedicada a mis padres sin quienes las frases que acabo de mencionar nunca se hubiesen escrito.

AGRADECIMIENTOS

Hoy que cumplo una meta más de las que en el camino tengo que alcanzar, recuerdo todo lo que viví a lo largo mi trayectoria universitaria y siento la necesidad de agradecer a todas aquellas personas que me acompañaron durante este largo camino. En primeria instancia agradezco de manera infinita a mi Padres, quienes con su sacrificio, fueron los que me motivaron y apoyaran incondicionalmente durante todo este arduo proceso; agradezco a mi Hermana que ha sido mi compañía y fiel testigo, de todo el tiempo y esfuerzo invertido. Agradezco de manera especial a mi Tía, quien como madre me acogió durante cuatro años en la universidad, ganándose mí eterno cariño. Mil gracias a mis abuelos que de alguna u otra forma y de la manera que fuese siempre me apoyaron. Agradezco a mis familiares, primos y amigos que siempre han estado presentes en todas las etapas de mi vida. Quiero agradecer también a mi Director de tesis el Ing. Vladimir Robles, a quien le brindo mi eterna gratitud, pues sin su guía, apoyo, y más importante su amistad, este trabajo no hubiese sido posible. Finalmente pero no menos importante agradezco a todas las personas que integran el grupo de Investigación en Inteligencia Artificial y Tecnologías de Asistencia con quienes, aparte de haber compartido muchos momentos gratos y crear nuevas amistades, conozco y he vivido todo el trabajo que ahí se invierte para crear tecnologías a favor de una sociedad más equitativa.

- Modelling domain knowledge of Speech and Language Therapy with an OWL ontology and OpenEHR archetypes. Robles Bykbaev, V. E., López Nores, M., Pazos Arias, J. J., García Duque, J. & Ochoa-Zambrano J., (2015). In *8th International Conference on Health Informatics (HEALTHINF). Lisbon, Portugal : INSTICC.*

- 

Bluetooth Terminal: con 10.133 descargas activas hasta el 24 de Noviembre de 2104 y 26.793 descargas totales. Entre los países con más descargas se encuentran: Estados Unidos, Alemania, Rusia, Reino Unido, Brasil, México, Corea del Sur, España, Japón e India. Posee una valoración de 4/5 estrellas. Disponible en:

https://play.google.com/store/apps/details?id=ptah.apps.bluetoothterminal&hl=es_419.

Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: INTRODUCCIÓN Y REVISIÓN DEL ESTADO DEL ARTE.

1.1 Introducción

Gracias al gran avance que han tenido los asistentes personales digitales (*smarthphones* o *tablets*), en la actualidad, ha permitido que éstos pasen a ser parte integral de la vida cotidiana. Asimismo, el vertiginoso desarrollo de aplicaciones móviles también ha tenido un gran impacto en el que hacer de cada persona, ya sea para realizar tareas simples como organizar agendas, realizar consultas en internet, uso de mapas, hasta tareas más personalizadas dependiendo de cada usuario. Android al ser un sistema operativo para dispositivos móviles de código abierto, patrocinado por Google, proporciona múltiples herramientas para el desarrollo de aplicaciones, por lo que el número de desarrolladores para esta plataforma crece cada día alrededor de todo el mundo.

Un autómata digital es un sistema complejo que consta de un sistema mecanismo que le permite el desplazarse, manipular objetos o realizar una cierta cantidad de tareas de manera automática, gracias a la presencia de un computador abordo que se encarga de controlar todos los procesos, así como la adquisición de datos que recibe del ambiente o del usuario para obtener como respuesta la ejecución de alguna tarea específica. Este tipo de autómatas tiene fines múltiples, ya sea en el área médica, militar o para la asistencia en el hogar, pudiendo ser incluidos en prácticamente cualquier parte del ámbito social en donde se requiera su ayuda.

En esta tesis se describe la elaboración de un plataforma móvil que permite a un dispositivo móvil basado en Android convertirse en un autómata, obteniendo como resultado un asistente personal, con la capacidad de desplazarse y realizar tareas básicas mediante el uso de técnicas de visión por computador y reconocimiento por voz. La plataforma que se desarrolla permite que sea actualizable, además se aprovecha las grandes prestaciones en hardware que poseen los actuales dispositivos móviles; a medida de que los *smarthphones* o *tablets* crecen en su capacidad, también se podrá ir mejorando el desempeño del autómata como tal.

1.2 Objetivos

1.2.1 Objetivo General

Diseño e implementación de un asistente móvil con desplazamiento autónomo basado en dispositivos Android.

1.2.2 Objetivos Específicos

- Investigar sobre todos los recursos que ofrecen los dispositivos móviles con plataforma Android que pueden ser utilizados con el fin de convertirlo en un autómata interactivo.

- Investigar sobre las técnicas de visión por computador disponibles para Android, centrado el enfoque al reconocimiento facial y de objetos.

- Investigar sobre las APIs de reconocimiento de voz disponible para dispositivos Android.

- Diseñar y desarrollar una estructura adecuada, sobre la cual el dispositivo móvil será montado y le brindará la capacidad de desplazarse de manera autónoma, con capacidad para hacer manipulación básica de objetos (pinza).

- Diseñar e implementar la aplicación para sistemas Android, encargada de controlar todo el sistema, con la capacidad de realizar funciones básicas de autómata interactivo y con una interfaz amigable con el usuario, mediante el uso de un personaje interactivo (avatar).

- Implementar dentro de la aplicación un conjunto básico de instrucciones que se activarán por comandos de voz y que servirán como un método de interacción con el usuario.

- Proveer al autómata la capacidad de reconocer un número limitado de objetos y rostros, mediante la aplicación de técnicas de visión por computador.

1.3 Asistentes personales y autómatas

1.3.1 Asistentes personales

- Definición de asistente personal:

El asistente personal se define aquella persona que apoya a otra persona que sufra de algún tipo de discapacidad que limite la acción de sus actividades en el hogar [1].

Por lo tanto el asistente personal es aquella persona que colabora de manera parcial o total en las tareas cotidianas del hogar, a personas que por su diversidad funcional no las pueden realizar de manera normal [1].

El asistente personal surge por la necesidad, de brindar la misma calidad de vida a toda aquella persona que sufra de alguna discapacidad, con los mismos derechos, la misma dignidad, equidad y oportunidades que posee el resto de la ciudadanía [2].

- Tareas del asistente personal:

Las tareas que debe realizar un asistente personal son variadas de acuerdo a las necesidades que tenga la persona a la que brinda soporte. Podemos citar algunas de las tareas de un asistente personal [1] [2]:

- Personales.
- Hogar.
- Acompañamiento.
- Conducción.
- Interpretación.
- Coordinación.
- Excepcionales.
- Especiales.

1.3.2 Asistentes personales digitales

- Definición del Asistente Personal Digital:

Un asistente personal digital o PDA por sus siglas en inglés, es un dispositivo portátil que consta de algunos elementos tales como procesador, memoria, pantalla táctil y una de las más importantes que es la funcionalidad de acceder a una red inalámbrica, todo esto dentro de un dispositivo compacto que se puede llevar en el bolsillo [3].

Una de las primeras definiciones de PDA surgiría en el año de 1992, con el lanzamiento del "Apple Newton" desarrollado por Apple, que aunque fue un fracaso total en su época, debido a las limitaciones tecnológicas que existían en aquellos tiempos, sembraría el precedente para lo que son los actuales PDAs. [4]

Hace algún tiempo la funcionalidad principal de los PDA, era únicamente de servir como organizadores personales, con la capacidad de manejar agendas, directorios de contactos, calendario, y posteriormente ha ido creciendo en su aplicabilidad, de tal forma que actualmente sirven tanto como aparatos en los que se puede leer un libro, hasta el uso de la navegación por medio de mapas [5].

Los Actuales Asistentes Personales Digitales *smarthphones* y *tablets*, poseen una gran capacidad de hardware respecto a sus predecesores y además una gran cantidad de recursos a su favor, lo que los convierte en herramientas muy útiles con distintos fines, de acuerdo a la necesidad de cada usuario.

- Características y usos del Asistente Personal Digital:

Un PDA posee recursos de hardware como memoria RAM, ROM, procesador, pantalla táctil, funcionalidades de red [3], acceso a cámara, GPS, como alguno de los recursos básicos.

Los usos que se le puede dar a este tipo de asistente son variados y dependerán mucho del hardware y software que posea, pudiendo llegar a tener aplicaciones ya sea en área personal, profesional, médica, militar, automovilística, etc.

1.3.3 Autómatas

Los primeros autómatas conocidos datan de hace algunos siglos atrás, cuando el hombre tratando de imitar animales, plantas o el movimiento de los cielos, inventaba intricados mecanismos que pudiesen imitar de manera realista dichos movimientos. Desde aquel momento surgirían las primeras bases, que nos permitirían llegar a los autómatas que conocemos hoy en día.

- Historia de los autómatas:

En la antigüedad se creaban artefactos que fuesen capaces de realizar tareas comunes para el hombre, con el fin de facilitarles la vida cotidiana [6], pero no todos estos artefactos tenían una utilidad determinada, algunas máquinas simplemente eran inventadas con el único fin de entretener a sus dueños [6].

A lo largo de la historia se dieron muestra de artefactos, existiendo los primeros precedentes de autómatas en Egipto. Desde la XII dinastía se encontraron muñecos articulados que pueden ser considerados como los primeros proto-autómatas [7].

Uno de los primeros ejemplos de autómatas se registra en la antigua Etiopía. En el año 1500 AC, Amenhotep, hermano de Hapu, construye le estatua o el coloso de Memnón [6], dicha estatua emitía sonidos parecidos a los que emite las cuerdas de una lira, cuando sobre ella incidían los rayos del Sol naciente [8].

Figura 1.-Coloso de Memnón [8]

King-su Tse, en el año 500 AC, en China, inventa una urraca voladora, elaborada tan solo de madera y bambú, además un caballo de madera que era capaz de saltar. Así también existen registros que entre al año 400 a 397 AC, Archytar de Tarento construye un pichón de madera, que se mantenía suspendido de un pivote, el cual era capaz de rotar gracias al uso de un surtidor de agua o vapor, simulando de esta manera el vuelo. En el año 206 AC, se descubrió el tesoro de Chin Shih Hueng Ti, que consistía en una orquesta mecánica de muñecos [6].

En el 400 a.c. en Alejandría se crearon algunos de los primeros autómatas conocidos como el teatro de autómatas y los Pájaros [9].

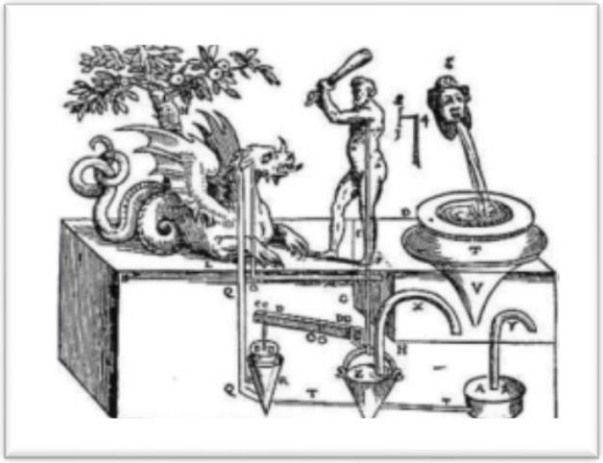


Figura 2.-Teatro de autómatas [9]

Un autómata más reciente fue el Gallo de Estrasburgo que funcionó desde 1352 hasta 1789, siendo el autómata más antiguo que se conserva intacto hasta la actualidad, que formaba parte del reloj de la catedral de Estrasburgo y que imitaba el movimiento de un gallo moviendo las alas y el pico cuando daba la hora [6] [9].

Figura 3.-Gallo de Estrasburgo [6]

Entre los siglos XVII y XVIII, se originaron un gran número de autómatas que poseían algunas de las características que poseen lo robots actuales. Dichos autómatas fueron creados en su gran mayoría por relojeros muy ingeniosos. La única misión que poseían estos dispositivos era la de entretener a las personas de los estratos sociales altos y servir de atracción en las ferias [6].

Dentro de uno de los grandes artefactos se encuentra el pato de Vaucanson, que era capaz de imitar de manera muy realista los movimientos de un pato real; podía beber, chapotear graznar, así como también podía imitar los gestos que hace un pato cuando traga con

precipitación. Los alimentos que ingería el pato, eran digeridos por disolución, para posteriormente poder excretarlos [6].

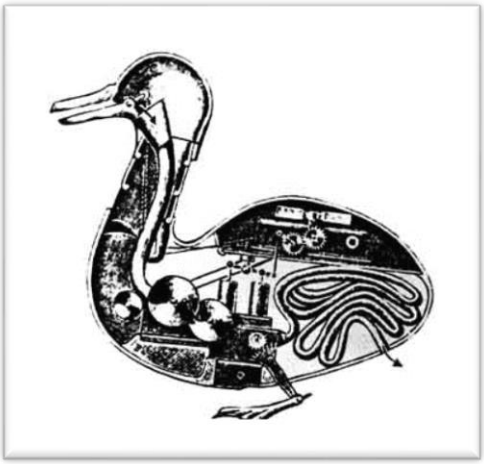


Figura 4.-Pato de Vaucanson [6]

Sería en la revolución industrial cuando los autómatas comenzarían a sustituir al hombre en ciertas tareas, siendo Jacquard el primero en utilizar tarjetas perforadas para realizar un trabajo automatizado, creando los primeros telares mecánicos [6] [9].

Ya en los años 40 a 50 surge el movimiento de la cibernética, siendo el más destacado el neurólogo y robotista Grey Walter [10], que gracias a la invención de los transistores y circuitos integrados logra crear "las tortugas", que podían esquivar obstáculos y volver al punto de recarga antes de que se les agotasen las baterías [9].

Figura 5.-Tortuga de Gray Walter [9]

De ahí en adelante el desarrollo de computadores y las grandes prestaciones que poseen, permitiría dar un paso gigante en el desarrollo los autómatas, hasta llegar a los autómatas tan complejos que conocemos hoy.

Los autómatas actuales son sistemas, que constan de mecanismos complejos , que conjuntamente con un computador que funciona como operador lógico es capaz de realizar tareas específicas para distintas aplicaciones, aunque últimamente con el desarrollo de procesadores más rápidos y de mayor capacidad, se ha vuelto más común la implementación de algoritmos más elaborados, como las redes neuronales, lógica difusa, etc.

La capacidad de aprender es una de las cualidades en las cuales más se trabaja actualmente, para brindarles a los robots mayor autonomía, y es en este punto en donde interviene la Inteligencia Artificial. Esta propone como fundamento base la creación de sistemas que posean un comportamiento inteligente [11], Alan Turing propuso uno de los primeros métodos, para el desarrollo de inteligencia artificial, estudiando el comportamiento de las personas, llegando a concebir el famoso "Test de Turing" o "Test de Imitación", que sería el centro en la definición de un elemento artificial [11].

1.4 Dispositivos móviles inteligentes

El dispositivo móvil Inteligente, teléfono inteligente o del inglés "*smarthphone*", es un dispositivo que tuvo como origen brindar la movilidad que los antiguos teléfonos cableados no poseían.

Es un dispositivo basado en la tecnología de ondas de radio, se encuentra entre un teléfono móvil clásico y una PDA, ya que permite realizar llamadas y enviar mensajes de texto como un móvil convencional, pero además posee características similares a las de un ordenador [12].

Una característica importante de casi todos los teléfonos inteligentes es que permiten la instalación de múltiples aplicaciones, que incrementa la cantidad de usos que se le pueda dar al mismo [4].

1.4.1 Evolución de las redes móviles

El primer teléfono móvil fue desarrollado por Motorola, el *Dynatac 8000x* inventado en 1983 por Martin Cooper [13]. Este teléfono móvil transmitía simplemente la voz de manera analógica sin ninguna codificación, lo que sería denominada como la primera generación o 1G, desde ese punto evolucionaria a la red digital, en donde la voz sería digitalizada y codificada, esta red seria denominada como la segunda generación o 2G. En la siguiente tabla podemos observar la evolución de las redes móviles, así como la tecnología y las velocidades que maneja cada una.

Redes	1G	2G	2.5G	3G	3.5G	4G
Definición	Analógica	Digital Narrowband Circuit Data	Packet Data	Digital Broadband PacketData	Digital Broadband PacketData	Digital Broadband PacketData IP
Tecnología	AMPS	TDMA CDMA GSM	GPRS	CDMA 2000 UMTS EDGE	HSPA	LTE
Velocidad	14.4 kbps	9.6/14.4kbps	20-40kbps	400-700kbps	1-3Mbps	3-5 Mbps

Tabla 1.- Evolución de las redes Móviles

1.4.2 Características de los dispositivos móviles inteligentes

El término teléfono inteligente, se refiere a un teléfono que reúne un cierto número de características, que lo denominan como tal, entre ellas está la capacidad de instalar en el dispositivo un sistema operativo completo con determinadas aplicaciones, cada una de ellas con funcionalidades diversas, para la ejecución de tareas simples o complejas [14].

Dentro de algunas características básicas de los smarthphones son el uso de pantallas táctiles, conectividad a Internet, correo electrónico [12], organizadores de agenda, calendarios, cámara digital integrada, administración de contactos, programas de navegación, así como editores de texto para varios formatos [4].

Todos los *smartphones* constan de un sistema operativo, que les brinda características similares a las que posee un ordenador, por tanto lo que los vuelve igual de vulnerables a virus que ataquen el sistema operativo [4].

Casi todos los *smarthphones* en el mercado actual (2014) poseen características similares, que varían un poco de acuerdo a los criterios de cada fabricante. De las características que lo convierten en un teléfono inteligente como tal podemos destacar:

- Sistema Operativo
- Servicios Multimedia: reproducción de audio y video en distintos formatos, grabación de videos, captura y edición de imágenes, manipulación de documentos [12].
- Funcionalidad de teléfono, como realizar y recibir llamadas.
- Correo electrónico [14].
- Una de las características más destacables, reside en la capacidad de poder instalar aplicaciones, lo incrementa su funcionalidad en gran medida [14].
- Interfaz gráfica para el ingreso de datos [14].
- Conexión a Internet por distintos medios inalámbricos [14].
- Agenda digital [14].

- Lectura y edición documentos de texto [14].
- Sistema de Posicionamiento Global: GPS
- Instrumentos de Medición: Acelerómetro, Giroscopio, Barómetro, Temperatura.
- Sensores Biométricos

1.4.3 Microprocesadores de los dispositivos móviles inteligentes

La gran mayoría de smarthphones posee procesadores basados en tecnología ARM, diseñados por una empresa inglesa también llamada ARM [14].

Dentro de ARM tenemos la generación de gama-baja ARM7, gama-media ARM11, gama-alta Cortex-A8, y algunos fabricantes ya están usando procesadores Cortex-A9 [14]. A continuación se presenta una comparación la velocidad de los procesadores en DMIPS (Dhrystone Million Instructions Per Second) [14]:

- ARM11: 1,2 DMIPS por MHz
- ARM Cortex A8: 2 DMIPS por MHz
- ARM Cortex A9: 2,5 DMIPS por MHz

Claramente notamos la diferencia de velocidad entre las distintas gamas, obviamente la velocidad de cada procesador dependerá de la frecuencia de reloj que se utilice.

A continuación se presenta una tabla de los procesadores más comunes usados por los dispositivos móviles inteligentes, así como la arquitectura ARM que utilizan los mismos

PROCESADOR	ARQUITECTURA
SAMSUNGHUMMINGBIRD (S5PC110)	ARM Cortex A8
APPLE A4	ARM Cortex A8
SNAPDRAGON	ARM Cortex A8
OMAP	ARM Cortex A8
NVIDIA TEGRA	ARM Cortex A9-Doble Núcleo
SAMSUNG ORION	ARM Cortex A9-Doble Núcleo
OMAP 4	ARM Cortex A9-Doble Núcleo

Tabla 2.-Arquitectura de procesadores más usados [14]

Dentro de los sistemas operativos para smarthphones más influyentes en el mercado actual podemos citar:

- Android
- IOS
- BlackBerry
- Windows Phone

En la Figura 6 se muestra el porcentaje de dispositivos móviles inteligentes por sistema operativo vendidos en Latinoamérica. Los datos fueron provistos por *Kantar Worldpanel* para el 2013.

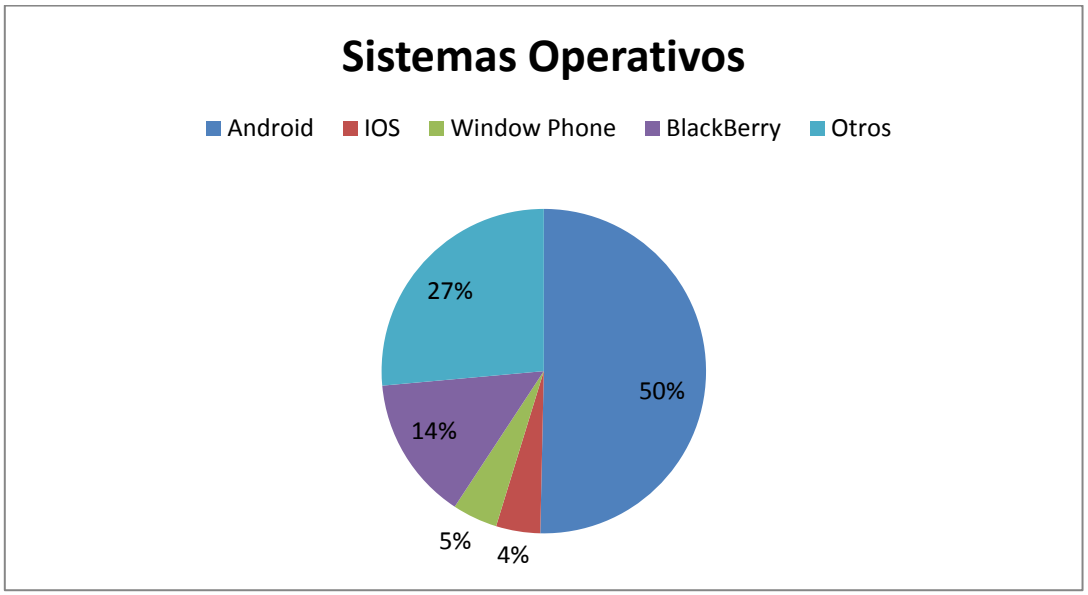


Figura 6.-Porcentaje Dispositivos Móviles Inteligentes por Sistema Operativo, vendidos para los tres meses a Septiembre de 2013 [15].

Como podemos observar en la Figura 6, notamos que el principal sistema operativo para móviles inteligentes es Android, en parte gracias a que es una plataforma de código abierto, lo que permite a desarrolladores de todo el mundo crear infinidad de aplicaciones para este Sistema Operativo.

1.5 Casos de estudio de los asistentes personales y autómatas

En esta sección se describirá brevemente los asistentes personales disponibles para dispositivos móviles inteligentes, así como también los robots autómatas que cumplen tareas de asistencia personal.

1.5.1 Asistentes personales para dispositivos móviles inteligentes

- **Google Now:**

Es un asistente personal inteligente, desarrollado por Google con el cual se puede interactuar mediante comandos por voz. Este servicio te permite obtener información sobre el estado del tráfico antes de salir a trabajar, rutas de navegación, sitios cercanos populares, etc. Permite el acceso a las aplicaciones instaladas en el teléfono tan solo con hablarle [16].

Utiliza la hora del día, la ubicación actual y el historial de ubicaciones para presentar información actualizada que sería útil durante el transcurso del día. También utiliza la información de los servicios de Google, como el historial web, para mostrar información sobre deportes o vuelos, o de las entradas de calendario sincronizadas para presentar recordatorios de citas [16].

- **Siri:**

Siri es el asistente personal desarrollado por Apple, se puede usar la voz para enviar mensajes, programar reuniones o hacer llamadas [17].

No solo es capaz de reconocer cada palabra, sino que es tan inteligente como para captar el contexto global de toda la oración, con la capacidad de almacenar tus preferencias más destacadas [17].

Posee funciones de dictados, sabe qué aplicación tiene que usar para cada cosa que le pidas y encuentra respuestas a tus preguntas y además se está integrando con algunos fabricantes de coches, para facilitar la conducción gracias a la presentación Eyes Free [17].

- **Indigo:**

Indigo es un asistente personal desarrollado por Artificial Solutions, se encuentra disponible para IOS, Android y Windows Phone [18]. Al igual que Google Now y Siri responde a órdenes por voz.

Permite administrar aplicaciones preinstaladas, como contactos, tiempo, mapas, calendarios, recordatorios y alarmas, así como componer y enviar mensajes de texto [18].

Se conecta a una cuenta en Internet, para tener un registro de sus preferencias y poder mostrar información relevante cuando sea necesario [18].

1.5.2 Autómatas como asistentes personales

- **Autom™:**

Autom es un entrenador personal para la pérdida de peso, fue desarrollado por la empresa *Intuitive Autómata*. Diseñado para motivar a un usuario para seguir con su dieta durante mucho más tiempo que las soluciones de pérdida de peso existentes [19].

Se ha demostrado que aumenta en gran medida las probabilidades de tener éxito en una dieta, lo cual ayudará a las personas a reducir significativamente los graves problemas de sobrepeso tales como la obesidad, que alcanzan cifras crónicas en todo el mundo. *Autom* está diseñado para prevenir este tipo de enfermedades, manteniendo un control diario tanto de las actividades físicas, como de la ingesta de alimentos [19].

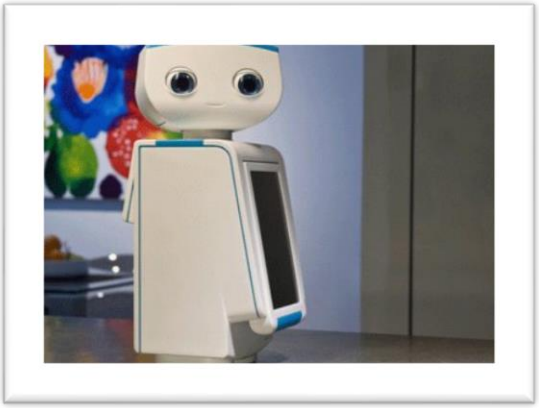


Figura 7.-Autom (Image © Intuitive Automata 2009) [19]

- **PR2:**

La empresa de *Georgia Tech* conjuntamente con los Ingenieros de *Willow Garage*, diseñaron el PR2, un asistente personal para Henry Evans, quien sufre de tetraplejia. Se construyó una interfaz de usuario basada en los bosquejos que Henry había realizado en una presentación Power Point y por primera vez en mucho tiempo Henry pudo afeitarse por sí solo; se añadió la piel sensible al PR2, de modo que el robot podía sentir cuando se

tocaba a Henry. Finalmente se implementó una interfaz sencilla para que Henry pudiese hacer del PR2 un títere. [20]

Figura 8.-PR2-Robots for Humanity(Robots para la Humanidad) [20]

- **Docomo:**

Es un asistente, con el cual se puede interactuar verbalmente, que además puede recomendar música, o actividades. Maneja el concepto de "nube personal" capaz de realizar búsquedas de la información solicitada y transmitirla al smartphone del usuario. [21]

- **Asimo:**

Representa la culminación de dos décadas de investigación robótica humanoide por los ingenieros de Honda. ASIMO puede correr, caminar en pendientes y superficies irregulares, girar suavemente, subir escaleras, alcanzar, agarrar objetos, así como también puede comprender y responder a comandos de voz simples. ASIMO implementa reconocimiento facial, pudiendo reconocer el rostro de un grupo limitado de individuos. Puede llevar un registro de los objetos estacionarios entorno, brindándole la capacidad de evitar obstáculos a medida que avanza a través de un ambiente [22].

Figura 9.-Imagen frontal del robot Asimo [22]

- **NAO:**

NAO es un robot humanoide de 58 centímetros de alto; está diseñado para ser un compañero amistoso alrededor de la casa. Se mueve, reconoce, escucha, e incluso habla [23].

NAO posee una combinación única de hardware y software; consiste en sensores, motores y software impulsado por Naoqi, su propio sistema operativo [24].

Su programación, así como bibliotecas de movimiento están disponibles a través de herramientas de como *Choregraphe* y otros softwares de programación avanzada [24]. Dichos programas permiten a los usuarios crear comportamientos complejos, acceder a los datos adquiridos por los sensores y controlar el robot como tal [24].

El Objetivo de Aldebaran la empresa desarrolladora de NAO, es poner un robot de alto rendimiento accesible en el mercado que pueda servir como plataforma de exploración para cualquier persona que quiera crear nuevas aplicaciones [24].

- **HSR:**

El Robot de soporte Humano (HSR) de Toyota está siendo desarrollado para ayudar a las personas en sus actividades cotidianas. En el futuro se espera que el HSR coexista los con miembros de la familia dentro del hogar, colaborando en las actividades diarias con el fin de mejorar la calidad de vida [25].

Posee un cuerpo compacto y ligero, capaz de adaptarse mejor a distintos ambientes; gracias a un brazo articulado y cuerpo telescópico, HSR puede cubrir un amplia área de trabajo [25]. Asimismo está dotado de un mecanismo de interacción segura, ya que el brazo del robot usa poca energía y se mueve lentamente para prevenir accidentes y lesiones [25].

HSR implementa una interfaz de interacción simple, ya que puede ser controlado de forma intuitiva a través de comandos de voz o por medio de una sencilla interfaz gráfica [25].

Figura 10.-HSR de Toyota [25]

Tecnología [25]:

- Brazo Articulado.
- Mano Flexible.
- Reconocimiento de Objetos.
- Reconocimiento de Ambiente.
- Funciones Remotas.

Es importante mencionar que Toyota desarrolló el prototipo de HSR, con el fin de ayudar a las personas con movilidad limitada, siendo el objetivo principal, mejorar la calidad de vida de dichas personas. La investigación y el desarrollo del prototipo se realizó en colaboración con la Asociación Dog de Japón, a fin de poder identificar las necesidades y deseos de las personas con movilidad limitada [25].

En 2011, se realizaron ensayos en los hogares de personas con discapacidad en sus miembros, utilizando el robot HSR, en cooperación con la Fundación de Rehabilitación de Yokohama [25].

Además, en respuesta a la tasa de envejecimiento de la población en Japón, Toyota colaborará con organizaciones de investigación, como universidades, así como con personas que participan en enfermería o en el área de salud para investigar y desarrollar nuevos usos y funciones para el HSR [25].

CAPITULO 2: DESARROLLO DE APLICACIONES EN ANDROID

2.1 Introducción al desarrollo de aplicaciones en Android

2.1.1 Introducción a Android

Android es una plataforma para dispositivos Móviles de código abierto, promovida por Google y propiedad de Open Handset Alliance [26]. Android puede ser instalado en un gran número de dispositivos móviles y día a día el número de dispositivos que usan Android se incrementa gracias a que al ser un sistema operativo de código abierto brinda la posibilidad a que el número de desarrolladores para esta plataforma se incremente.

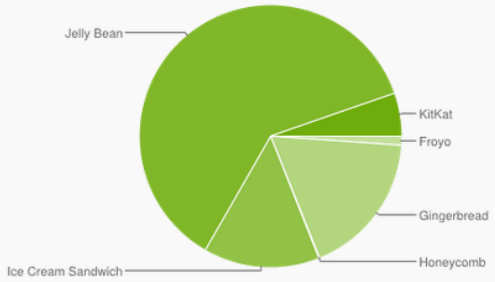
Android está escrito en su mayoría en código nativo que corre sobre su propia máquina virtual Dalvik. Se encuentra bajo licencia abierta, lo que permite a los desarrolladores indagar en lo más profundo del código y usarlo o modificarlo para diferentes propósitos sin violar ninguna ley de derechos de autor [26].

En la Tabla 3, se pueden apreciar las versiones existentes de Android hasta el momento (2014).

Nombre Código	API Nivel	Versión
	1	1.0
	2	1.1
Cupcake	3	1.5
Donut	4	1.6
Eclair	5	2.0
Eclair	6	2.01
Eclair	7	2.1
Froyo	8	2.2
GingerBread	9	2.3
Gingerbread	10	2.3.3
Honeycomb	11	3.0
Honeycomb	12	3.1
Honeycomb	13	3.2
Ice Cream Sandwich	14	4.0
Ice Cream Sandwich	15	4.0.3
Jelly Bean	16	4.1
Jelly Bean	17	4.2
Jelly Bean	18	4.3
KItKat	19	4.4

Tabla 3.- Versiones de Android

Version	Codename	API	Distribution
2.2	Froyo	8	1.1%
2.3.3 - 2.3.7	Gingerbread	10	17.8%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	14.3%
4.1.x	Jelly Bean	16	34.4%
4.2.x		17	18.1%
4.3		18	8.9%
4.4	KitKat	19	5.3%



Data collected during a 7-day period ending on April 1, 2014.
Any versions with less than 0.1% distribution are not shown.

Figura 11.- Distribuciones de Android más populares a lo largo del 2014 [27]

2.1.2 Configuración del entorno de desarrollo

El primer paso para la configuración del entorno de desarrollo es la descarga del JDK (*Java Development Kit*) y el JRE *(Java Run Time Environment*), para cada sistema operativo. El entorno de desarrollo se configura de manera muy similar tanto en Windows, GNU/Linux como Mac OS, con ligeras variaciones dependiendo de cada uno de los sistemas operativos. En la página oficial para desarrolladores se puede descargar el ADT Bundle de Android, que incluye Eclipse, el complemento ADT[1] y SDK dentro de un solo paquete previamente configurado, listo para ser usado y empezar el desarrollo de aplicaciones en Android lo más rápido posible.

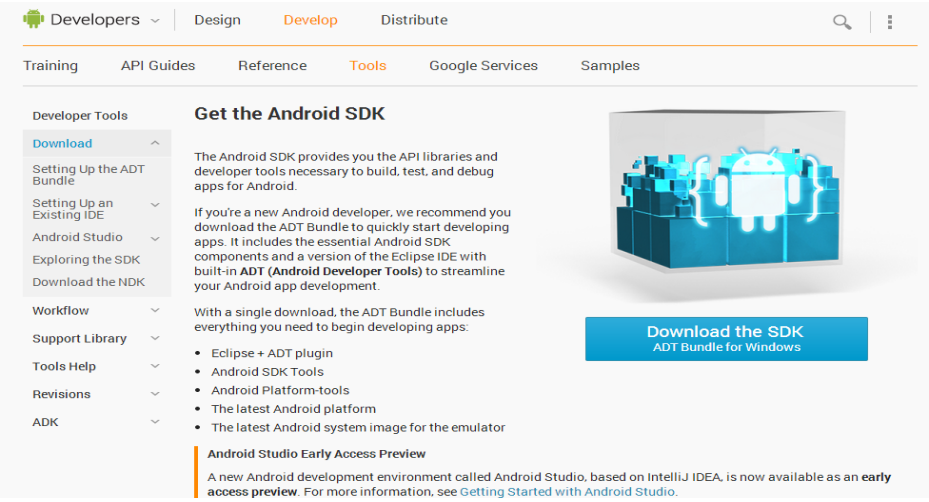


Figura 12.-Pagina para la descarga del ADT Bundle de Android

[1] ADT.- Herramientas de desarrollo Android diseñadas para Eclipse

La segunda opción es configurar el entorno de desarrollo manualmente, para ello es necesario descargar en primer lugar el IDE de Eclipse. Esto se puede realizar desde la página oficial de Eclipse y allí es factible encontrar varias distribuciones, de la cuales se puede escoger la más adecuada según el criterio de cada de desarrollador, para este caso se usó "*Eclipse IDE for Java EE Developers*", la versión "Kepler" que se encuentra actualmente (2014) disponible para la descarga. Una vez descargado Eclipse es necesario instalar el complemento ADT (*Andorid Development Tools*) en el mismo, para ello nos dirigimos a **Help>Install New Software**. Una vez ahí, se abrirá una ventana en donde procedemos a dar clic en la opción **Add,** lo que desplegara una nueva ventana en la cual ingresaremos el nombre del complemento y la dirección (URL) de la cual descargaremos el mismo.

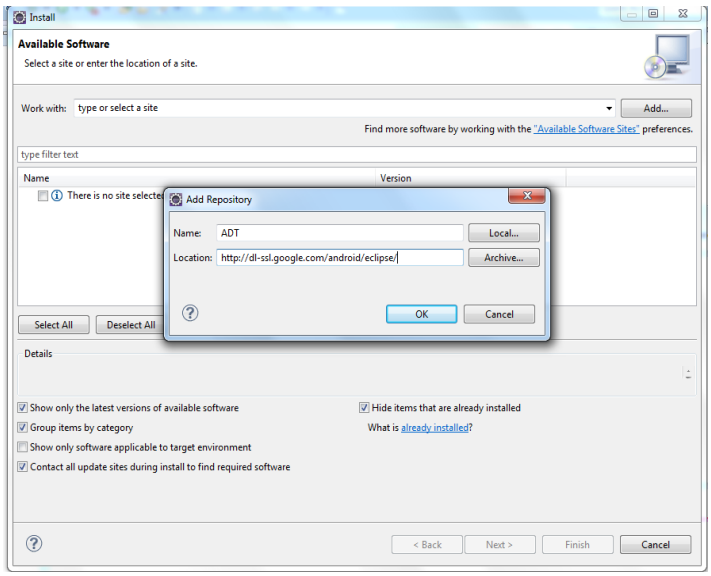


Figura 13.- Instalación del plugin ADT

La dirección del repositorio en donde se encuentra el complemento es: *https://dl-ssl.google.com/android/eclipse/* . Una vez agregado el nombre y la dirección tomara un tiempo hasta que Eclipse calcule las dependencias y los paquetes a instalar. Finalizado el paso anterior, indicará los paquetes que se pueden instalar, entre ellos se encuentra el "*Developers Tools*" que es el paquete principal para el desarrollo de aplicaciones, además presentara el "*NDK plugin*" necesario para el desarrollo de aplicaciones escritas parcial o totalmente en código nativo(C/C++).

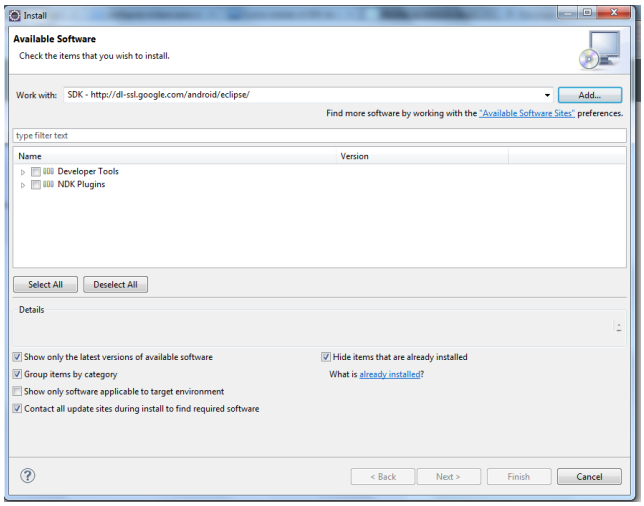


Figura 14.- Selección de paquetes del plugin ADT

Una vez se termine la instalación de los paquetes necesarios, será preciso reiniciar Eclipse para que los cambios tomen efecto, finalmente el último paso será indicar la ruta del SDK de Android previamente descargado, para ello en Eclipse nos dirigimos a **Window>Preferences>Android** e ingresamos la ruta en donde se encuentre el SDK.

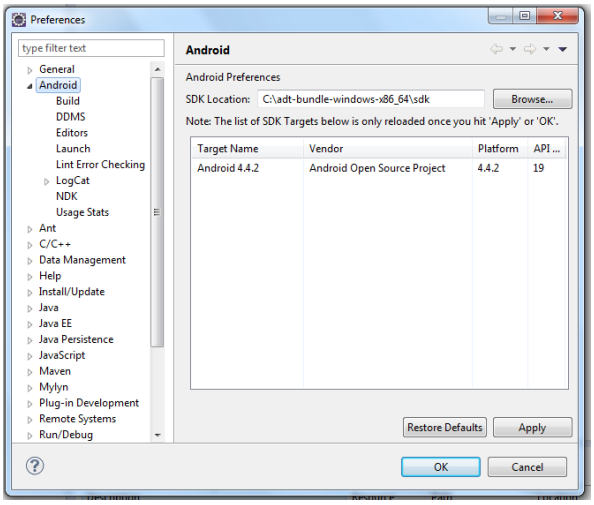


Figura 15.- Ubicación del directorio del SDK de Android

Después de completar todos estos pasos el entorno estará listo para el desarrollo de aplicaciones en Android con todas las herramientas que dispone esta plataforma, tales como el *SDK Manager, AVD Manager y la Perspectiva DDMS.*

Es importante recordar que cuando se vayan a desarrollar aplicaciones usando un dispositivo físico, se debe activar la depuración USB en mismo.

2.1.3 Herramientas del entorno de desarrollo

En esta sección se describirá brevemente, las funcionalidades y las características de las herramientas más importantes que proporciona el SDK cuando está integrado correctamente en el entorno de desarrollo Eclipse.

2.1.3.1 SDK Manager

El SDK (*Software Development Kit)* es la herramienta necesaria para el desarrollo de aplicaciones en Android. Dentro de él se puede gestionar las versiones de Android para las que se quiera desarrollar aplicaciones, así como las herramientas de compilación y las imágenes del sistema operativo Android para los dispositivos móviles que se pueden emular. Cuando es instalado por primera vez viene con los paquetes esenciales, así como con la última versión de Android que se encuentre disponible, de ahí en adelante se puede descargar los paquetes que se crean necesarios. El SDK de Android tiene una estructura modular, lo que permite descargar los paquetes que requieran [28].

A continuación se describen los componentes que conforman el SDK manager [28]:

- **SDK Tools:** incluye varias herramientas que se usan para el desarrollo de aplicaciones. Son las herramientas esenciales para los desarrolladores, se puede considerar que es el núcleo de las herramientas.
- **Plataform Tools:** herramientas adicionales que son desarrolladas conjuntamente con el núcleo de la plataforma, son actualizadas cada vez que existe una nueva versión de la plataforma.
- **Android Plataform:** Una plataforma es creada para cada versión de Android que surge. Cada versión incluye las librerías de compilación, imagen del sistema, los temas que usará el emulador y las herramientas específicas para cada plataforma.
- **Google APIs:** son las librerías adicionales para tener acceso a servicios específicos de Google, como por ejemplo, los mapas.
- **Drivers:** contiene los controladores que permiten a un dispositivo Android comunicarse con el ordenador.
- **Samples and Documentation:** contiene ejemplos del código y documentación para cada versión de Android.
- **Add-ons:** son extras de terceros que contienen herramientas y librerías, incluyendo el paquete de soporte de Android y el *Analytics SDK*. También posee librerías especiales para determinados dispositivos.

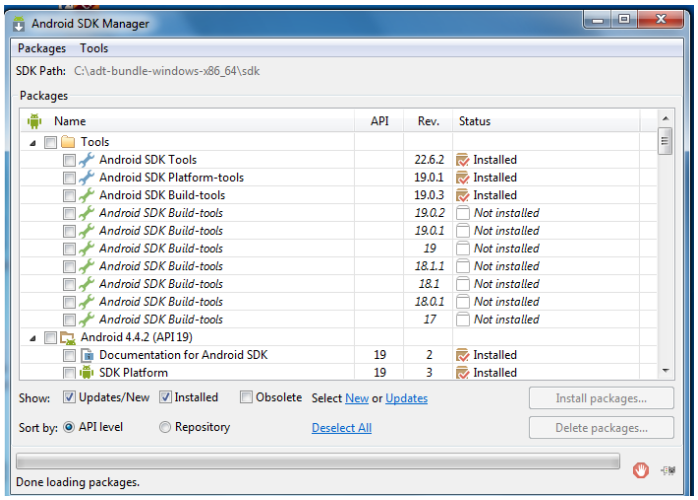


Figura 16.- SDK Manager

2.1.3.2 ADB

El ADB (Android Debug Bridge) es la herramienta que permite al desarrollador interactuar con el dispositivo físico o el dispositivo virtual, posibilitando instalar o desinstalar aplicaciones durante el desarrollo. El proceso ADB es Cliente-servidor, por lo que permite conectarse con varios clientes [28] [29].

2.1.3.3 AVD Manager

Permite Administrar los dispositivos virtuales que se pueden emular y que son usados para realizar pruebas de funcionamiento de las distintas aplicaciones que se desarrollen. Los AVD(Android Virtual Devices) están basados en los emuladores QEMU[2], permitiendo emular el Hardware de un dispositivo Android, con partes intercambiables para probar las aplicaciones desarrolladas en distintas configuraciones [29].Dentro del AVD podemos crear dispositivos con distintas características, tales como la versión de Android, el procesador, la resolución de la pantalla, la memoria RAM, ROM, además permite emular una cámara o usar la del computador si este la posee. Hay que tener en consideración que la memoria RAM y el tamaño de la tarjeta SD que se le asigne al dispositivo no es virtual.

[2] QEMU.- Emulador de Procesadores

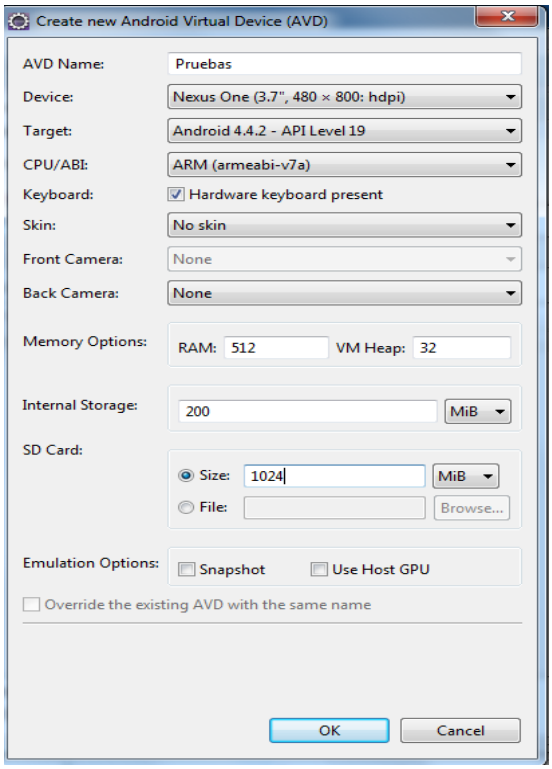


Figura 17.- Creación de un dispositivo Virtual

.

2.1.3.4 Perspectiva DDMS

La perspectiva DDMS (Dalvik Debug Monitor Server), es una de las herramientas que ayudan en gran medida al desarrollo de aplicaciones. Esa un director de tráfico entre el puerto único que Eclipse usa para conectarse a una JVM (Java Virtual Machine) y a los múltiples puertos de un dispositivo Android, para cada una de las instancias de la máquina virtual Dalvik [29]. Es una herramienta utilizada para monitorear el consumo de memoria dentro de un determinado tiempo, brindando gran cantidad de información sobre la cantidad de memoria y procesos que consume una aplicación en desarrollo [28].

Dentro de la perspectiva se puede monitorear ya sea el dispositivo virtual o el dispositivo físico. Dentro de las funcionalidades que posee esta perspectiva podemos citar las siguientes: capturas de pantalla, acceder a los archivos del dispositivo, monitorear todos los procesos que se están ejecutando, la carga del procesador, muestra información de los hilos y la pila de memoria.

Otra de las funcionalidades que ofrece esta perspectiva en el caso de dispositivos virtuales, es emular llamadas telefónicas, enviar mensajes y emular coordenadas GPS.

Finalmente, una de las más importantes características que incluye es poder leer los registros de una aplicación. La perspectiva DDMS brinda acceso al *LogCat*, herramienta

que nos permite observar todos los procesos que se están ejecutando cuando se lanza una aplicación e incluso aplicar filtros y solo ver los registros que se crean más importantes [30]. Es de gran utilidad al momento de detectar fallas en la programación.

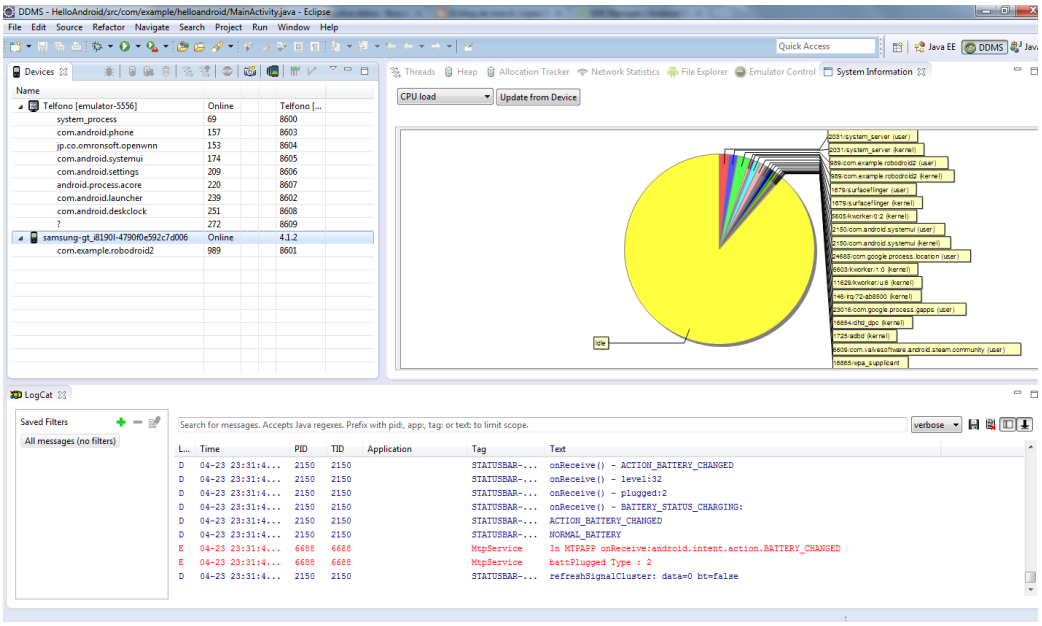


Figura 18.- Perspectiva DDMS

2.2 Librerías de desarrollo en Android

Antes de describir las librerías de Android es necesario conocer un poco sobre la arquitectura del sistema operativo como tal.

2.2.1 Arquitectura del sistema operativo Android

Android está basado en el Kernel Linux compuesto por varias capas, cada una de ellas con un propósito determinado. Podemos dividir la arquitectura de Android en tres capas, en la parte superior se encuentran las aplicaciones, en la parte central el armazón de la aplicaciones, la librerías y el *Android Runtime*, y finalmente en la parte inferior se encuentra el Kernel Linux [26] [30] [31].

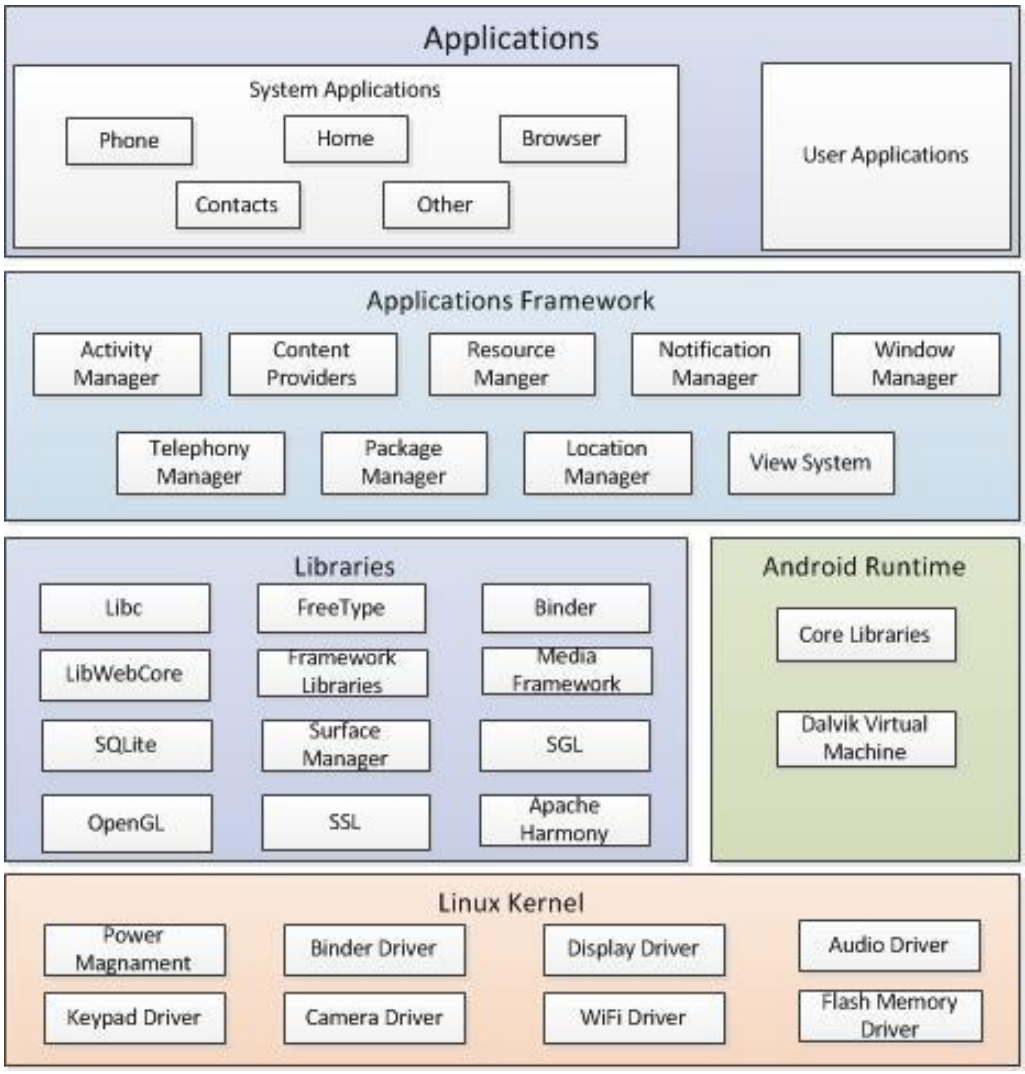


Figura 19.- Arquitectura del sistema operativo Android [31] [26]

2.2.2 Librerías de Android

Las librerías están escritas en C y C++, su función principal es la de brindar soporte a la capa *FrameWork Application* (base de las Aplicaciones) y a la final será con las cuales el desarrollador trabajará cuando cree nuevas aplicaciones. Algunas librerías son usadas para la construcción del Sistema Operativo Android y otras librerías son tomadas de otros desarrolladores para complementar el sistema operativo [26].

A continuación se describen algunas de las librerías más importantes:

- **Libc:** es una derivación del proyecto BSD, como implementación del estándar C, adaptada a las necesidades del sistema operativo Android [26] [31].
- **Free Type:** esta librería soporta el renderizado de *bitmaps* así como de vectores de representación de fuentes [31]
- **Binder:** es un proceso de comunicación interno, que permite a una aplicación Android hablar con otra [26]
- **Lib Web Core:** provee de un motor eficiente para navegadores web. Se encarga de controlar los navegadores web en Android. Está basado en el *WebKit*, y es usado por Google Chrome, Safari y otros [26] [31].
- **Framework Libraries:** son varias librerías designadas para soportar a los servicios del sistema, tales como localización, instalación de paquetes, telefonía, WiFi, etc. [26].
- **Media Framework.** Esta librería está basada en el paquete de video *OpenCore*, capaz de soportar la reproducción y grabación en varios formatos de audio y video, así como trabajar con imágenes. Soporta formatos de video como MPEG4 H.264, MP4 AAC. AMR, e imágenes en formatos JPEG y PNG [31].
- **SQLite:** esta librería provee un motor poderoso y eficiente de base de datos con todas las funcionalidades de SQL, se encuentra disponible para todas las aplicaciones que necesiten de él. Es usado por sistemas operativos como Mozilla FireFox e IOS [26] [31].
- **Surface Manager:** esta librería se encarga del manejo o acceso al subsistema de la pantalla, estando compuesto de las capas gráficas 2D y 3D [31]
- **SGL:** provee el soporte para el motor de gráficos en 2D [31].
- **OpenGL:** es la librería para gráficos 3D provista por *OpenGL*, es capaz de usar aceleración grafica 3D por Hardware si el dispositivo lo soporta [26] [31].
- **SSL:** esta librería provee la capa de seguridad en los puertos de comunicación, permitiendo conectividad segura punto a punto [26] [31].
- **Apache Harmony**: Una implementación de código abierto de la librerías de Java [26].

2.3 Estructura y características de una aplicación base

2.3.1 Componentes de una aplicación Android

Las aplicaciones en Android están compuestas por varios componentes, y se debe considerar que no todas las aplicaciones poseen todos los componentes. Los componentes implementados dependerán de la funcionalidad que provea la aplicación al usuario.

La arquitectura de Android permite a las distintas aplicaciones intercambiar o reusar recursos, lo que resulta en un ahorro significativo en el consumo de memoria del dispositivo [31]. Algunos componentes son el "*Service*" que se encarga de las tareas en segundo plano, los "*Content Providers*" para almacenamiento de información para múltiples aplicaciones, y los "*Broadcast Reciver*" para la escucha de los "*Intents*" por parte de otras aplicaciones [32].

En esta sección se tratarán los componentes más importantes que conforman una aplicación Android.

2.3.1.1 Activities

Una actividad es un componente que presenta una interfaz al usuario con la cual se puede interactuar con la aplicación, de esta manera cada actividad posee su interfaz de usuario y cada aplicación puede poseer varias actividades según lo requiera. La relación entre una actividad y la interfaz de usuario no siempre es uno a uno sino que se parece más al control en el paradigma MVC[3] [30].

Toda actividad posee un ciclo de vida, dicho ciclo de vida consta de siete métodos que se pueden sobrescribir a conveniencia, en la Figura 20 se puede ver el ciclo de vida de una actividad

[3] MVC: Modelo vista-controlador

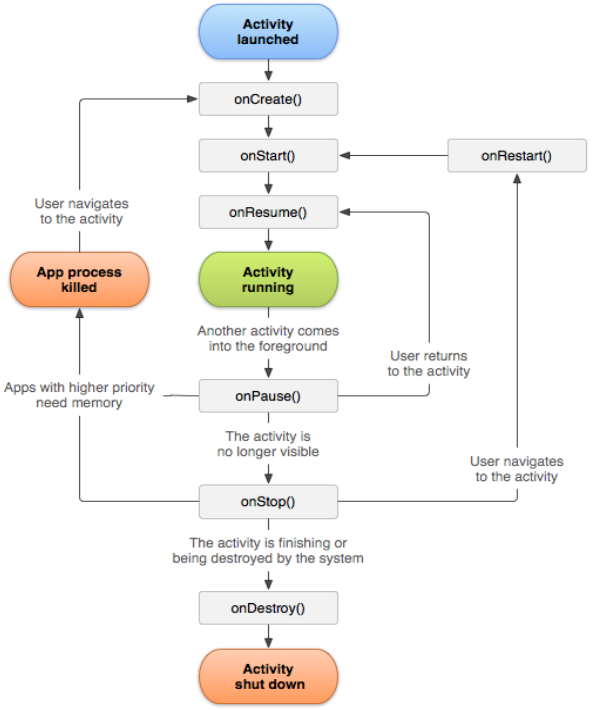


Figura 20.- Ciclo de vida de una Actividad [33]

El hecho de lanzar una actividad implica el consumo de una gran cantidad recursos, es por ello que se establece se ciclo de vida, mismo que es gobernado por el *Activity Manager*. A continuación se describen los métodos dentro del ciclo de una actividad [32] [29]:

- ***onCreate():*** es llamado cuando la aplicación es creada por primera vez, es aquí donde lanzan la interfaces de usuario, se cargan los datos etc. El parámetro para este método es el *Bundle*, que permite guardar el estado de la actividad para posteriormente recuperarlo.
- ***onStart():*** es llamado una vez que la actividad es visible al usuario.
- ***onRestart():*** es llamado después de que una actividad ha sido detenida y vuelta a iniciar.
- ***onResume():*** es llamado cuando la actividad comienza a interactuar con el usuario.
- ***onPause()****:* es llamado cuando la instancia de una nueva actividad comienza a hacerse visible y la actividad actual será detenida dejando de interactuar con el usuario. Este estado es usado para guardar los datos flotantes en datos permanentes , a fin de que puedan ser recuperados en el método *onResume()*;
- ***onStop():***es llamado cuando otra actividad es visible para el usuario y el usuario ya no puede interactuar con la anterior.
- ***onDestroy():*** este método es el último en el ciclo de la actividad y es llamado cuando la actividad va a ser destruida, liberando todos los recursos que se estén usando. Este puede ser invocado por la llamada *finish()*.

2.3.1.2 Services

Son procesos que corren en segundo plano a los cuales el usuario no posee acceso. Son utilizados para realizar tareas que se ejecutan fuera del hilo principal. La ventaja de que una tarea se ejecute en segundo plano es que no bloquean las tareas que se encuentran en el hilo principal, como suelen ser las interfaces de usuario.

Poseen un ciclo de vida más pequeño que el de una actividad, con lo cual se puede detener, pausar o reanudar el proceso según corresponda. Son un proceso que es más controlado por el desarrollador y menos por el sistema, por lo que es importante que el desarrollador implemente correctamente los servicios, controlando que no utilicen más recursos de los que necesita, puesto que esto implicaría un gran consumo en la memoria del CPU [26].

La mejor práctica es ejecutar los servicios de manera periódica, justo cuando son necesitados y de la misma manera terminar con ellos cuando han cumplido la tarea que se les haya sido asignada [30]. Un claro ejemplo de los servicios, puede ser la reproducción de música sin que esto influya mientras se navega dentro de otras aplicaciones [26].

Los servicios están clasificados en locales y remotos [31]:

- Servicios Locales: corren en el mismo proceso que el resto de la aplicación, permitiendo que sea fácil la implementación de tareas que se ejecutan en segundo plano.
- Servicios Remotos: corren en un proceso diferente, permitiendo la comunicación entre los distintos procesos.

2.3.1.3 Broadcast Receiver

El *Broadcast Reciver* nace de la necesidad de que las aplicaciones deben responder a eventos globales, como puede ser un mensaje o una llamada entrante [30]. De esta manera la aplicación debe responder a ciertos eventos que pueden ser originados por el sistema y realizar alguna acción, para ello primero es necesario estar registrado en el *Broadcast Reciver*. Este proceso funciona mediante *Intents* o intentos, que pueden ser disparados por algún elementó del sistema, será en aquel momento cuando el mismo comience a ejecutarse con el fin de lanzar algún servicio o actividad [29] [26].

2.3.1.4 Content Providers

El *Content Provider* es el elemento que permite acceder o compartir la información almacenada que posean otras aplicaciones. El *Content Provider* implementa los métodos estándares para que una aplicación acceda a los datos que posee otra, permitiéndole leer o escribir dicha información. Este proceso puede utilizar cualquier método para almacenar datos, ya sea en archivos del sistema o en una base de Datos SQL [26].

Por defecto Android ejecuta una aplicación en su propia caja de seguridad, lo que aísla totalmente los datos de la aplicación del resto de aplicaciones del sistema, pero existen pequeñas partes de información que pueden ser compartidas entre las mismas [30].

En la Tabla 4 se muestran los métodos y la operación que posee el *Content Provider*.

Operación	**Método**
Crear	insert()
Leer	query()
Actualizar	update()
Eliminar	delete()

Tabla 4.- Métodos de un ContetProvider [26]

Un ejemplo del uso del *Content Provider*, es leer los contactos que posea el dispositivo almacenado para luego ser usados en otra aplicación.

2.3.1.5 Intents

Los *intents* o intentos, lanzan una actividad, inician, terminan o enlazan servicios. Los *intents* son procesos asíncronos, que se inician en cualquier parte del código sin la necesidad de que este haya sido completado [26].

Básicamente son mensajes que describen una acción a realizar, pueden ser clasificados en implícitos y explícitos. Un *intent* explícito designa su componente objetivo por su nombre, ya que generalmente dicho nombre es conocido por el desarrollador. Estos Intentos son usados usualmente para enviar mensajes o realizar acciones dentro de una misma aplicación, como puede ser el lanzar una segunda actividad. En un *intent* implícito no se conoce el nombre del componente objetivo, por lo que es utilizado para iniciar componentes de otras aplicaciones, en este caso Android seleccionará el mejor componente o componentes para terminar la acción en respuesta a este intento [31].

Dentro de los *intents* tenemos los *Intents* e *ItentsFilter* [30]:

- *Intents:* es una declaración de necesidad, crea un número de piezas de información para decidir la acción o el servicio a ejecutar.
- *ItentsFilter:* es una declaración de capacidad e interés de ofrecer ayuda a quien la necesite, pudiendo ser genérica o especifica con los resultados que se quiere obtener con este intento.

2.3.2 Creación de una aplicación Android en Eclipse

Para la creación de un nueva aplicación en Eclipse se utiliza el *Wizard* que se instala con el ADT (Android Development Tools), para ello nos dirigimos a **File>New>Android Applicaction Project;** una vez desplegado el *wizard* presentará un dialogo como aprecia en la Figura 21.

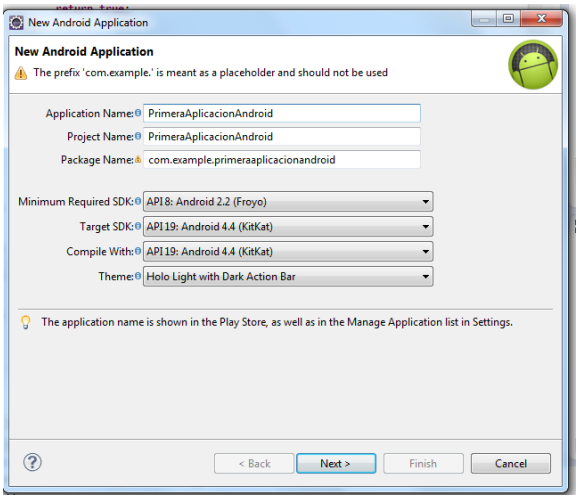


Figura 21.- Creación de un proyecto Android

En este punto será necesario asignar el nombre que tendrá la aplicación, el nombre del proyecto y el paquete del mismo, además se puede seleccionar la versión de Android que se desee utilizar, la versión de Compilación, la mínima versión que soportara la aplicación y si se requiere se puede escoger alguna apariencia para la interfaz de usuario.

Una vez seleccionada la mejor opción para le creación del proyecto, nos indicará que se creará un nueva actividad y el icono de lanzamiento (ver Figura 22).

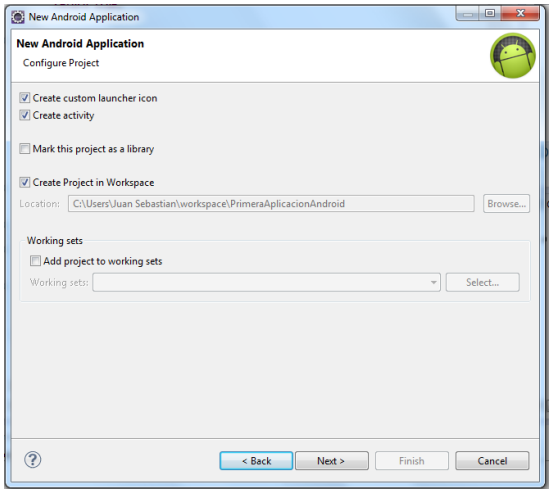


Figura 22.- Creación de un proyecto Android

Sin cambiar las opciones que nos da el *Wizard* se continúa a la siguiente ventana, que será la selección y configuración del icono de lanzamiento (ver Figura 23). En esta ventana se podrá seleccionar ya sea un Clipart de los que viene por defecto en el *wizard*, un texto, o una imagen propia desde los archivos del sistema, en cuyo caso el *wizard* se encargará de ajustar la imagen a la resolución adecuada para cada tipo de dispositivo.

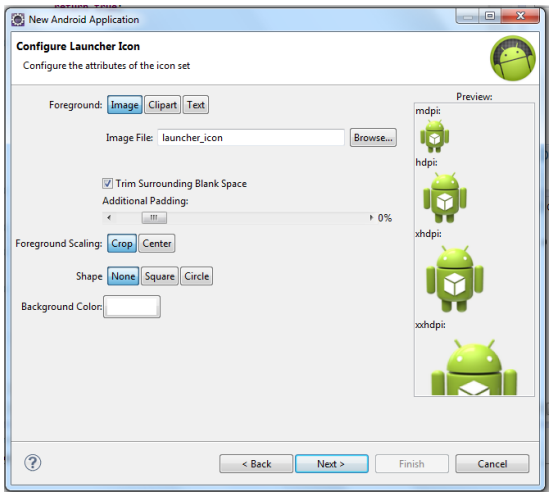


Figura 23.- Creación de un proyecto Android

Terminado este punto, se continuará con la creación de la nueva actividad (ver Figura 24), en ella se puede escoger entre una actividad en blanco (*Blank Activity)*, una actividad en pantalla completa (*FullScren Acitivity)* y finalmente una *Master*/*Detail Flow activity* que creará un proyecto con un flujo en forma de lista y cada elemento de lista con su componente descriptivo.

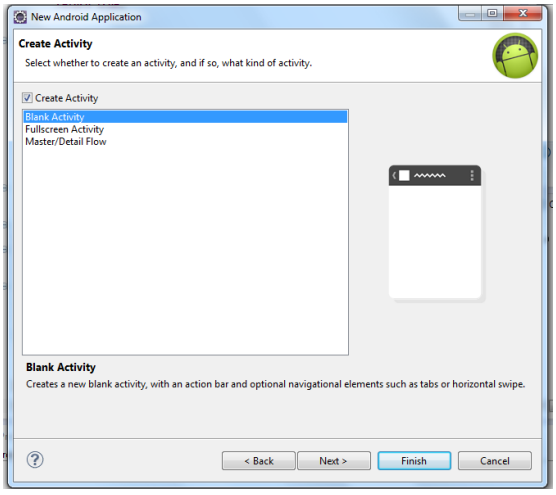


Figura 24.- Creación de un proyecto Android

El último paso para la creación del nuevo proyecto será asignar el nombre que tendrá la actividad a crearse y el nombre del archivo de los recursos gráficos o la interfaz de usuario (ver Figura 25).

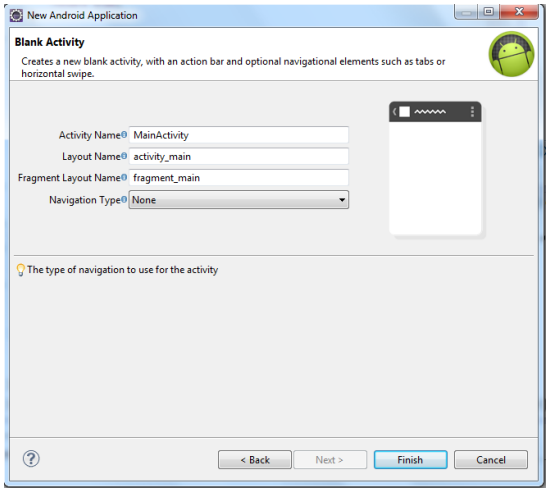


Figura 25.- Creación de un proyecto Android

Una vez finalizado el proceso, el *wizard* creara el nuevo proyecto Android, con todos los archivos, recursos y librerías para el correcto funcionamiento y compilación de la misma.

2.3.3 Estructura de una aplicación Android

2.3.3.1 Estructura de un proyecto Android

Los recursos que conforman la estructura de un proyecto Android están organizados como se puede apreciar indica en la Figura 26.

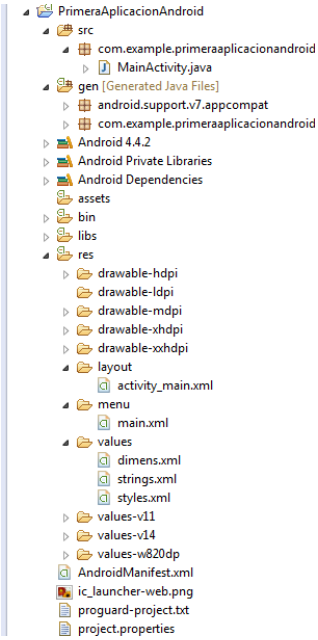


Figura 26.- Estructura de un proyecto Android

A continuación se describe los elementos del proyecto [34]:

- **src:** es el directorio que contiene el código fuente en Java para la aplicación.
- **gen:** el directorio en donde Android almacena el código autogenerado por las herramientas de compilación de Android.
- **assets:** contiene los archivos adicionales que se desean usar en el proyecto, como por ejemplo pueden ser imágenes, tipos de fuentes o archivos de audio.
- **bin:** es el directorio que contiene a la aplicación una vez que es compilada.
- **libs:** contiene las librerías de terceros que la aplicación puede necesitar.
- **res:** contiene los recursos del proyecto, como los iconos, textos, estilos, GUI Layout, etc.

Dentro del directorio ***res*** se encuentra algunos subdirectorios, cada uno con una funcionalidad específica [34]:

- **Drawable:** contiene las imágenes necesarias para el proyecto, ya sean JPEG o PNG. Posee carpetas individuales que guardan las imágenes para las distintas resoluciones de teléfonos existentes.
- **Layout:** contiene las interfaces de usuario escritas en código XML. Se pueden crear carpetas para cada una de las distintas resoluciones de teléfonos existentes
- **Menú:** contiene las especificaciones para menús escritos en código XML
- **Values:** contiene los valores de textos, dimensiones y estilos. Se pueden crear carpetas para brindar soporte en varios idiomas.

2.3.3.2 Android Manifest

El *Android Manifest* es el archivo principal que posee toda aplicación, dentro de él se presenta información esencial sobre la misma. Asimismo contiene la de los componentes de la aplicación, como son las actividades, *Content Providers*, *Services*, *Broadcast Reciver* e *Intents* [35].

Dentro del *Manifest* también se indica cual será la Actividad principal o la encargada de lanzar la aplicación, así como la versión de la aplicación, los permisos y características que requiere, la versión de Android para la que será diseñada y la mínima versión que será capaz de soportar [26].

En el siguiente ejemplo se puede apreciar el contenido del archivo *Manifest.*

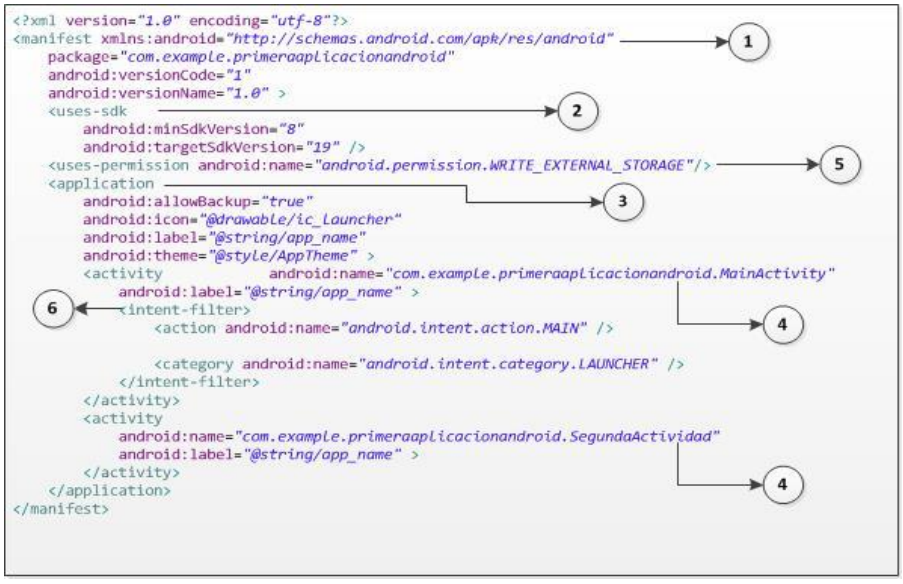


Figura 27.- Estructura del Android Manifest

A continuación se detallan los elementos básicos del *Android Manifest* [35]:

1. <manifest>: además de soportar la estructura de todo el *manifest* contiene:
   - versionCode: usada para indicar la versión de la aplicación, que pueden ser números enteros como 1, 2, 3…etc.
   - versionName: el nombre de la versión, utilizado al momento de la publicación de la aplicación; su propósito será de mostrar al usuario la versión de la aplicación.
2. <uses-sdk>: indica la versión de Android que usará, contiene:
   - minSdkVersion: versión mínima de Android que soportara.
   - tragetSdkVersion: es la versión de Android para la cual es desarrollada la aplicación.
3. <application>: Especifica todas las actividades que contendrá la aplicación, además de los *Services*, *Content Providers* y *Broadcast Reciver*.
4. <activity>: en esta sección se declaran las distintas actividades que contendrá la aplicación, así como los distintos atributos e *Intents*.
5. <usses-permisson>: declara los permisos que posee la aplicación para acceder a los recursos del dispositivo.
6. <intent-filter>: especifica el tipo de intentos con los cuales una actividad puede responder. Puede especificar elementos como:
   - action: agrega la acción al *Intent-Filter*.
   - category: define la categoría de la actividad.

Para más detalles acerca del uso y los elementos del *manifest* se puede encontrar información más detallada en la página para desarrolladores de Android.

2.3.3.3 El Archivo R.java

El archivo ***R.java*** es uno de los archivos más importantes dentro de un proyecto Android. Es la unión entre el código en Java y los recursos que posee el proyecto. Este recurso se genera automáticamente y por ello nunca debe ser modificado. Es recreado cada vez que surge alguna modificación dentro del directorio **res.** Este archivo contiene los identificadores mediante los cuales el código en Java puede localizar un recurso, como puede ser un texto, un *layout*, un *view* o las imágenes del directorio *drawable* [26].

2.3.3.4 Contenido de una Activity

Como se pudo apreciar en secciones anteriores toda actividad está asociada a un interfaz de usuario, así como también posee un ciclo de vida, cuyos métodos pueden ser redefinidos. Es necesario también recordar que cada actividad deberá estar declarada de manera adecuada dentro del *Android Manifest.* Para el análisis del contenido de la actividad se

usará como ejemplo el proyecto "PrimeraAplicacionAndroid" creado con anterioridad, El código de la actividad de dicho proyecto se puede apreciar en la Figura 28.

```
package com.example.primeraaplicacionandroid;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.linear);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

         getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

Figura 28.- Contenido de la actividad principal

Como vemos en ejemplo de la Figura 28, la clase lleva el nombre de *MainActivity* y extiende de *Activity.* Se puede apreciar que se redefine el método ***onCreate()***, que como se vio con anterioridad es llamado cuando la aplicación es creada. Además se le pasa el parámetro *Bundle* que es necesario cuando se requiere recuperar el estado de la actividad cuando la misma pasa a otro estado del ciclo de vida, luego es necesario invocar el constructor de la clase padre mediante la instrucción ***super.onCreate()***. Dentro del método ***onCreate()*** se indica la ubicación de la interfaz de usuario o su *layout* mediante el método ***setContentView()*** que es el encargado de inflar o mostrar la interfaz de usuario.

Finalmente se puede apreciar también que se ha sobrescrito el método ***onCreateOptionsMenu()*** que es implementado para crear el menú de opciones en caso de que sea necesario. El menú o menús se localizan dentro del directorio de recursos (**res/menú).**

Es importante recordar importar todos los paquetes necesarios que se vayan a usar.

2.3.4 Interfaces de usuario

Android organiza todas la interfaces de usuario en *views* y *layouts*, cada *layout* organiza las distintas *views* como pueden ser botones o cuadros de texto, se podría decir que los *Layouts* son los contenedores de los *view* [26].

2.3.4.1 Layouts

Los *layouts* pueden ser creados mediante código directamente en Java, o utilizando archivos en formatos XML. Los *layouts* en formato XML son considerados los recursos de la aplicación, por lo cual se ubicaran en el directorio **res/layout** del proyecto, en ellos se describe las propiedades así como el aspecto que tendrá interfaz de usuario. La combinación de distintos *layouts* anidados unos dentro de otros pueden crear interfaces muy complejas y atractivas para el usuario, pero es necesario tener cuidado con la cantidad de recursos de memoria que puede consumir el hecho de inflar dichas interfaces.

- **Linear layout.-** es uno de los más simple *layouts* que existen, únicamente apila horizontal o verticalmente todos los elementos (hijos) .Este elemento pregunta a sus hijos cuanto espacio necesitarán, para posteriormente asignarles dicho espacio [26].

Atributo	Método	Descripción
android:baselineAligned	setBaselineAligned(boolean)	Cuando es falso previene que el *layout* se alinee con la línea base de su hijo.
android:baselineAlignedChildIndex	setbaselineAlignedChildIndex(int)	Cuando un linear *layout* es parte de otro *layout* al cual está alineado, se puede especificar que se alinee a la línea base de su hijo.
android:divider	setDividerDrawable(Drawable)	*Drawable* usado como un divisor vertical entre dos botones.
android:Gravity	setGravity(int)	Especifica como un objeto debería posicionar su contenido.
android:measureWidthLargestChild	setmeasureWidthLargestChildEnable(boolen)	Cuando es verdadero, todos los hijos que tenga un ancho menor al hijo con ancho más largo, adoptaran el tamaño del más grande.
android:orientation	setOrientation(int)	Indica si el *Layout* será una columna o una fila, mediante el parámetro “Horizontal” o

		“Vertical”.
android:weightSum		Define la suma del ancho máximo

Tabla 5.- Atributos del archivo XML perteneciente al Linearlayout [36]

- **Frame Layout.-** está diseñado para bloquear un área en la pantalla para mostrar un único elemento. Por lo general es usado para contener un único hijo, por la dificultad que presenta en escalar y organizar los hijos para distintos tamaños de pantallas [37]

Atributo	Método	Descripción
android:foreground	setForeground(Drawable)	Define el dibujable que se dibujara sobre el contenido.
android:foregroundGarvity	setForeground(Gravity)	Define la gravedad que se aplicará al dibujable sobre el primer plano.
android:measureAllChildren	setMeasurAllChilldren(boolean)	Determina si se debe medir a todos los hijos o tan solo aquellos que se encuentren en el estado visible o invisible.

Tabla 6.- Atributos del archivo XML perteneciente al FrameLayout [37]

- **Relative Layout.-** este elemento alinea sus hijos de manera relativa unos entre otros. Este *layout* es bastante poderoso y permite ahorrar mucho tiempo en el diseño de una interfaz, ya que gracias a él es posible evitar la implementación de *layouts* anidados con el fin de obtener una interfaz de usuario atractiva que consuma una menor cantidad de recursos [26]. En la Tabla 7 se detalla los atributos XML para este *layout*.

Atributo	Método	Descripción
android:gravity	setGravity(int)	Especifica como un objeto debería posicionar su contenido.
android:IngoreGravity	setIgnoreGravity(int)	Indica que *view* es afectada por la gravedad.

Tabla 7.- Atributos del archivo XML perteneciente al RelativeLayout [38]

Los distintos *layouts* se pueden ir combinando a fin de obtener vistas más complejas, en la Tabla 8 se muestra un ejemplo de cada *layout*.

Layout	Código XML	
Linear Layout	<LinearLayout android:layout_width="match_parent" android:layout_height="match_parent" android:orientation="vertical" > </LinearLayout>	
Frame Layout	<FrameLayout android:layout_width="match_parent" android:layout_height="wrap_content"> </FrameLayout>	
Relative Layout	<RelativeLayout android:layout_width="match_parent" android:layout_height="wrap_content"> </RelativeLayout>	

Tabla 8.- Ejemplos de los distintos layouts

2.3.4.2 Views

Android posee una gran cantidad de elementos para ser usados en las distintas interfaces que se realicen. Cada actividad contiene *views*, y las clases *views* representan los elementos que se muestran en la pantalla de un dispositivo y son responsables de la interacción con el usuario mediante eventos [30].

En la Tabla 9 se detallan algunos de los *views* más utilizados así como el código XML empleado para la creación de los mismos

Elemento	Codigo Xml	Resultado
Text View	<TextView android:id="@+id/textView1" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="TextView" />	
Edit Text	<EditText android:id="@+id/editText1" android:layout_width="match_parent" android:layout_height="wrap_content" android:ems="10" > <requestFocus /> </EditText>	
Button	<Button android:id="@+id/button1" android:layout_width="wrap_content" android:layout_height="wrap_content"	

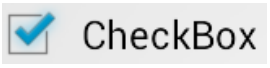
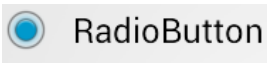
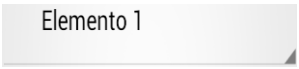
	android:text="Button" />	
Image Button	<ImageButton android:id="@+id/imageButton1" android:layout_width="wrap_content" android:layout_height="wrap_content" android:src="@drawable/ic_launcher"/>	
Check Box	<CheckBox android:id="@+id/checkBox1" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="CheckBox" />	
Radio Button	<RadioButton android:id="@+id/radioButton1" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="RadioButton" />	
Progress Bar	<ProgressBar android:id="@+id/progressBar1" android:layout_width="match_parent" android:layout_height="wrap_content" />	
Seek Bar	<SeekBar android:id="@+id/seekBar1" android:layout_width="match_parent" android:layout_height="wrap_content" />	
Spinner	<Spinner android:id="@+id/spinner1" android:layout_width="match_parent" android:layout_height="wrap_content" />	

Tabla 9.- Ejemplos de Views más comunes

Existen muchos más *views* que se pueden encontrar durante la creación del proyecto y pueden ser utilizados de manera similar a los que se muestran en la Tabla 10, obviamente con sus determinadas particularidades. Alguno de los atributos que la mayoría de *views* poseen se describe en la Tabla 10.

Atributo	Método	Descripción
Android:background	setBackgroundResorce(int)	Una imagen que utilizar como fondo.
Android:clickable	setClickable(boolean)	Describe si cierto *view* reaccionara cuando se haga clic sobre él.

Android:focusable	setFocusable(boolean)	Controla cuando se hace *focus* sobre una *view*
Android:id	setId(int)	Provee el nombre identificador que utilizara cierto *view*.
Android:padding	setpaddingRelative(int,int,int,in)	Aplica espaciamiento, en pixeles desde cada borde.
Android:text	setText(String)	Colocará un texto.

Tabla 10.- Atributos XML de los distintos Views [39]

Hay que tener en cuenta algunos aspectos importantes al manejar determinados atributos, como es el *android:text* especifica el texto por los *TextView o EditText*, ya que existen varias formas de hacerlo, la primera será colocar el texto directamente en el código XML como se muestra Figura 29.

```
android:text="TextView"
```

Figura 29.- Ejemplo del uso de un texto estático

La segunda opción y la más recomendada es crear una variable dentro del documento Strings.xml (ver Figura 30), misma que se encuentra en el directorio **res/values** y asociarla como en el ejemplo de la Figura 31.

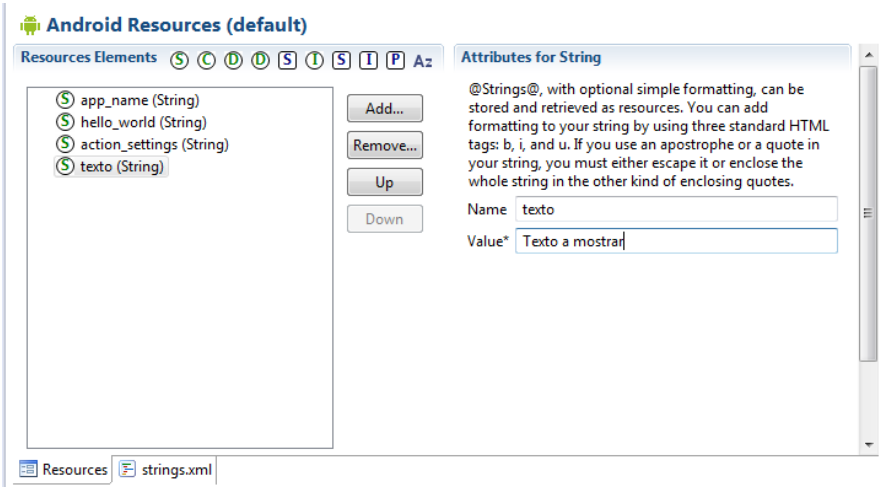


Figura 30.- Creación de una variable string dentro del directorio de recursos res/values

```
android:text="@string/texto"
```

Figura 31.- Asociación de una variable del archivo XML con un view

La tercera y última opción sería ingresar el texto desde el código en Java, mediante el método ***setText()*.**

Para el caso que existan *view*s que requieren imágenes (*ImageButton*, *ImageView*, etc.) se debe almacenar dichas imágenes dentro del directorio **res/drawable ,** de manera similar se puede indicar directamente la identificación de la imagen en el documento XML o también desde el código en Java con el método respectivo.

El atributo más importantes será el identificador (id), mismo que deberá ser asignado para que se pueda crear el identificador en el archivo *R.java* y de esta manera poder asociar un *view* al código en Java mediante el método ***findViewById()***, como se puede apreciar en el ejemplo de la Figura 32.

```
TextView mTextView=(TextView)this.findViewById(R.id.textView1);
EditText mEditText=(EditText)this.findViewById(R.id.editText1);
Button mButton=(Button)this.findViewById(R.id.button1);
ImageButton mImageButton=(ImageButton)this.findViewById(R.id.imageButton1);
CheckBox mCheckBox=(CheckBox)this.findViewById(R.id.checkBox1);
RadioButton mRadioButton=(RadioButton)this.findViewById(R.id.radioButton1);
ProgressBar mProgressBar=(ProgressBar)this.findViewById(R.id.progressBar1);
SeekBar mSeekBar=(SeekBar)this.findViewById(R.id.seekBar1);
Spinner mSpinner=(Spinner)this.findViewById(R.id.spinner1);
```

Figura 32.- Asociación de los distintos views con el código en Java

Una vez que se ha conectado Java con los elementos gráficos es posible hacer uso de las clases anidadas que posee cada *view* y sobrescribirlas, en la Tabla 11 se listan las clases que poseen las distintas *views.*

Clases e Interfaces anidadas
View.AccessibilityDelegate
View.BaseSavedState
View.DragShadowBuilder
View.MeasureSpec
View.OnAttachStateChangeListener
View.OnClickListener
View.OnCreateContextMenuListener
View.OnDragListener
View.OnFocusChangeListener
View.OnGenericMotionListener
View.OnHoverListener
View.OnKeyListener
View.OnLayoutChangeListener
View.OnLongClickListener
View.OnSystemUiVisibilityChangeListener
View.OnTouchListener

Tabla 11.- Clase e interfaces anidadas de la clase View [39]

La descripción de las clases e interfaces es algo extensa y se la puede encontrar directamente en la página para desarrolladores de Android.

2.3.5 Tareas asíncronas

Las tareas asíncronas son usadas para separar tareas del hilo principal, por ejemplo cuando se tiene tareas que requieren algún tiempo de procesamiento es necesario ponerlas en otro hilo para que la interfaz de usuario no se vea afectada cuando realiza dicha tarea. Android posee la Clase *AsyncTask* que a diferencia de los *threads* normales de Java posee métodos que permiten actualizar la interfaz de usuario antes, durante y después de que se ejecute el hilo secundario.

Para usar un AsyncTask es necesario seguir los estos pasos [34]:

- Crear la subclase del *AsyncTask.*
- Sobrescribir alguno o todos los métodos del *AsyncTask.*
- Cuando sea necesitado, crear la instancia del *AsyncTask* y invocarla con el método ***execute().***

Para la implementación de la subclase del *AsyncTask*, es necesario considerar tres tipos de datos [34]:

- El tipo de información que se requieren procesar.
- El tipo de información que se requiere que devuelva cuando se está ejecutando.
- El tipo de información que devolverá cuando termine la tarea.

Los métodos que posee el AsyncTask se describen en la Tabla 12.

Estado	Descripción
onPreExecute()	Es llamado antes de que la tarea sea ejecutada.
doInBackground(params…)	Es llamado después del método anterior, aquí será donde se ejecute la tarea en segundo plano.
onProgressUpdate(Progress…)	Es utilizado para mostrar o publicar cualquier progreso desde la tarea en segundo plano, cuando esta se encuentra en ejecución.
onPostExecute(Reult..)	Es llamado una vez que la tarea en segundo plano es terminada. El resultado de la tarea que se estaba ejecutando se pasara a este método.

Tabla 12.- Método de la clase AsyncTask [40]

En el estado ***doInBackround()*** se publicara el progreso de la tarea en ***onProgressUpdate()***, siempre y cuando se llame a ***onPublishProgress()***.

2.3.6 Manejo de los recursos del directorio Assets

El directorio **Assets** contiene los archivos adicionales que serán necesarios para la aplicación, como pueden ser sonidos, imágenes o estilos de fuentes, por dicha razón es necesario tener acceso a los mismos y, para ello Android posee la clase *AssetManager* que nos brida esta opción. Por lo tanto el primer paso será crear el constructor de la clase *AssetManager* como se muestra en la Figura 33.

```
AssetManager mAssetManger=getAssets();
```

Figura 33.- Constructor del AssetManager

Luego será necesario usar el método ***open()***, al cual se le pasa el nombre del archivo que se desea abrir y este nos devolverá un *stream* de datos que luego podrán ser convertidos.

Para el caso de una imagen se usa como se muestra en la Figura 34, decodificando el *InputStream* mediante el método ***decodeStream()*** propio de la clase *BitmapFactory*.

```
InputStream imagen=mAssetManager.open("MyImage.png");
Bitmap mImagen=BitmapFactory.decodeStream(imagen);
```

Figura 34.- Ejemplo del uso del AssetManager para Imágenes

En el caso de archivos de audio es distinto ya que para la reproducción es necesario un descriptor de archivo, para ello es necesario el *AssetFileDescriptor* y el método ***openFd(),*** el mismo devolverá el descriptor requerido. En el ejemplo de la Figura 35 se pude observar el uso *AssetManager* para archivos de audio.

```
AssetFileDescriptor sound = mAssetManager.openFd("alarm.ogg");
```

Figura 35.- Ejemplo del uso del AssetManager para un archivo de audio

Para cargar estilos de fuentes es necesario emplear el método ***createFromAssets()***, propio de la clase *TypeFace*, pasándole como parámetro el *AssetManager* y el nombre de la Fuente. En el ejemplo de la Figura 36 se muestra el uso del *AssetManager* para el caso de fuentes.

```
TypeFace font=Typeface.createFromAsset(mAssetManager, "accid.ttf");
```

Figura 36.- Ejemplo del uso del AssetManger para fuentes de texto

2.3.7 Generación de gráficos en 2D

2.3.7.1 SurfaceView

Para el dibujado o renderizado de recursos gráficos Android provee la clase *SurfaceView*. Esta clase posee una superficie de dibujo dedicado embebido dentro de la jerarquía *view*, encargándose de controlar los parámetros de la superficie. Además será responsable de ubicar la superficie en el lugar correcto de la pantalla [41].

Se puede crear una *SurfaceView* desde el código en java, o puede ser creada en un *layout*, pudiendo ser combinada con cualquier tipo de *view* sin ningún inconveniente.

El acceso a la superficie de dibujo o es controlador por la interfaz *SurfaceHolder* que puede ser obtenido llamando al método ***getHolder(),*** este método será usado para conocer en qué estado se encuentra la superficie. El objetivo principal de la clase *SurfaceView* es proveer de una superficie de dibujo con su propio hilo secundario que pueda realizar las renderizaciones sobre la pantalla [41] .

Par el correcto uso del *SurfaceView* será necesario tener en consideración los siguientes aspectos [41]:

- Todo *SurfaceView* y método *SurfaceHolder.Callback* deberán ser llamados desde el hilo que se está ejecutando.
- Que el hilo de dibujado esté renderizando sobre una superficie que exista, para ello es necesario manjar correctamente el *SurfaceHolder.Callback.surfaceCreated()* y el *SurfaceHolder.Callaback.surfaceDestroyed().*

Los métodos que deben ser sobrescritos se describen la Tabla 13.

Métodos	Descripción
SurfaceCreated()	Es usado cuando la superficie es construida o reconstruida.
SurfaceChanged()	Usado cuando la superficie sufre algún cambio, como puede ser el tamaño.
surfaceDestroyed()	Cuando la superficie es destruida y es necesario liberar recursos.

Tabla 13.- Métodos para la clase SurfaceView [41] [30]

Una vez que se ha implementado la superficie de dibujo correctamente, será necesario proceder a dibujar sobre ella, para lo cual se utiliza el método público ***Draw(Canvas canvas)*.**

2.3.7.2 Canvas

La clase *canvas* mantiene las imágenes que serán renderizadas sobre una superficie. Para dibujar es necesario cuatro elementos [42]:

- Un *Bitmap* para sostener los pixeles.
- Un *Canvas* para alojar el dibujo.
- Un objeto dibujable, como por ejemplo un texto o un *Bitmap.*
- Un elemento de la clase *paint*, que define las propiedades como el color o el estilo.

La clase *canvas* posee varios métodos, algunos de ellos se detallan en la Tabla 14.

Métodos	Descripción
drawARGB(inr a, int r, int g, int b)	Rellena el *canvas* con *bitmap* con los especificados colores ARGB.
drawBitmap(Bitmap bitmap, matrix matrix, Paint paint)	Dibuja un *bitmap* usando una matriz específica.
drawBitmap(Bitmap bitmap, float left, float top, Paint paint)	Dibuja un *bitmap* en las posiciones indicadas, usando un *paint* especifico.
drawCircle(float cx, float cy, float radius, Paint paint)	Dibuja un círculo en las coordenadas indicadas, con un determinado radio, usando un *paint* especifico.
drawText(String text, float x, float y, Paint paint)	Dibuja un texto en las coordenadas indica, con un *paint* especifico.

Tabla 14.- Métodos para la clase Canvas [42]

Finalmente para que se publique el *canvas* sobre la superficie será necesario llamar al método ***onDraw()*.**

2.3.7.3 BitmapFactoy

BitmaFactory es una clase importante cuando se desea dibujar recursos sobre una superficie. Esta clase crea objetos desde varias fuentes, incluyendo archivos, *streams* y arreglos de bytes. [43]

En la Tabla 15 se describen algunos de los métodos públicos más importantes para esta clase.

Métodos	Descripción
decodeByteArray(byte[] data, int offset, int lenght)	Decodifica un *bitmap* inmutable, desde un *byte array* específico.
decodeFile(String pathName, Option opts)	Decodifica un archivo desde un directorio en un *bitmap.*
decodeFileDescriptor(FileDescriptor fd)	Decodifica un *bitmap* desde un descriptor de archivo.
decodeStream(Inputstream is)	Decodifica un stream de datos en un *bitmap.*

Tabla 15.- Métodos para la clase BitmapFactory [43]

2.3.8 Reproducción de archivos de audio

Para la reproducción de archivos existen dos opciones el *MediaPlayer y* el *SoundPool*.

2.3.8.1 MediaPlayer

Android provee la clase *MediaPlayer* para la reproducción de los formatos más populares de audio y video, también posee la capacidad de grabar en ciertos formatos para luego almacenarlos en donde se crea más conveniente.

Para el uso del *MediaPlayer* es necesario seguir algunos pasos [29]:

- Crear la in instancia del *mediaPlayer* mediante el método ***create()*.**
- Inicializar el *mediaPlayer* con el archivo a reproducir.
- Prepara el *mediaPlayer* para la reproducción mediante el metodo ***prepare*()**.
- Iniciar la reproducción del *mediaPlayer* mediante el método ***start*()**.
- Durante la reproducción se puede ya sea pausar, detener o reiniciar la reproducción.
- Cuando la reproducción haya terminado es necesario liberar los recursos del *mediaPlayer* mediante el método ***release*().**

El directorio de los archivos a reproducir se puede situar ya sea dentro de los recursos del proyecto, desde la tarjeta SD o desde una página web. En la Figura 37 se muestra un ejemplo para cargar archivos desde distintas fuentes.

```
Para un Archivo Web:
        Uri mediaReference="http://someUriToaMediaFile.mp3";
        mediaplayer.setDataSource(this, mediaReference);

Para un directorio Local:
        mediaplayer.setDataSource("/sdcard/somefile.mp3");
```

Figura 37.- Ejemplo para cargar un archivo de audio desde una pagina web y un directorio local [29].

En la Tabla 16 se describen algunos de los métodos más utilizados en la clase *MediaPlayer*.

Método	Descripción
prepare()	Prepara el reproductor para la reproducción (Síncrono).
prepareAsync()	Prepara el reproductor para la reproducción (Asíncrono).
start()	Inicia o resume la reproducción.
stop()	Detiene la reproducción.
pause()	Pausa la reproducción.
reset()	Reinicia el reproductor el estado de no inicializado.
release()	Libera los recursos asociados con el objeto *MediaPlayer*.
isPlaying()	Comprueba si se está reproduciendo algún archivo.
getDuration()	Devuelve la duración del archivo.
getCurrentPosition()	Devuelve la posición en donde se encuentra la reproducción.

Tabla 16.- Métodos para la clase MediaPlayer [44].

2.3.8.2 SoundPool

La clase *SoundPool* permite manejar y reproducir archivos de audio desde los recursos del proyecto o desde un archivo del sistema. Esta clase utiliza el servicio *MediaPlayer* para decodificar los archivos de audio originales en archivos de audio PCM de 16 bits estéreo o mono, lo que permite reducir la carga del CPU y la latencia provocada por la descompresión al momento de la reproducción [45].

La clase *SoundPool* es capaz de soportar la reproducción de múltiples archivos simultáneamente, mediante el parámetro *maxStreams* en donde se puede definir la cantidad máxima de archivos que puede reproducir al mismo tiempo [45].

Algunas de las características más importantes de esta clase son [45]:

- Reproducción de archivos en bucles de repetición.
- Permite ajustar el volumen de la reproducción.
- Permite ajustar la velocidad de reproducción.
- Permite asignar prioridades para la reproducción a cada archivo de audio.

Cuando se vaya a usar la clase *SoundPool* es necesario implementar su constructor, al cual se le debe pasar tres parámetros:

- **maxStreams:** el número máximo de archivos a reproducir.
- **streamType:** el tipo de señal de audio.
- **srcQuality:** La calidad de conversor definida por la frecuencia de muestreo.

En la Tabla 17 se describe algunos de los métodos más importantes que pueden ser usados dentro de esta clase

Método	Descripción
autoPause()	Pausa todas las reproducciones.
autoResume()	Resume todas las previas reproducciones.
load(AssetFileDescriptor afd, int priority)	Carga un archivo de audio desde un descriptor de archivo *assets*.
pause(int StreamID)	Pausa el archivo en reproducción.
play(int soundID, float leftVolume, Float rightVolume, int Priority, int loop, float rate)	Reproduce un sonido mediante su identificador.
release()	Libera todo los recursos de *SoundPool*.
resume(int streamID)	Resume una reproducción.
setLoop(int streamID, int loop)	Activa la reproducción en bucle.
stop(int streamID)	Detiene una reproducción.

Tabla 17.- Métodos para la clase SoundPool [45].

2.3.9 Base de datos SQLite

Android utiliza el motor de base de datos *SQLite* que no requiere de un proceso de servidor separado. A diferencia de una base de datos para empresas que posee una gran cantidad de características relacionadas a la tolerancia a falla y al acceso simultánea a datos, para el caso de Android se ha eliminado algunas de las características que no son absolutamente necesarias para el almacenamiento a pequeña escala. Con *SQLite* la base de datos se vuelve un simple archivo dentro del sistema, que puede residir ya sea en la tarjeta interna externa de memoria [29]

Las instrucciones en *SQLite* residen en dos distintas categorías que serían crear y modificar Tablas. Los dos métodos principales para el manejo de Tablas son [29]:

- **CREATE TABLE**: este comando se encargara de la creación de una nueva tabla dentro la base de datos. En este método se deberá especificar un nombre único de tabla, además de las columnas y los tipos de datos que se escribirán en las mismas.

- **DROP TABLE**: remueve o elimina cualquier tabla con toda la información que esta contenga.

En la base de datos se pueden almacenar varios tipos de datos, para cada columna se podrá definir uno de los siguientes [29]:

- **TEXT:** un texto, almacenado usando la codificación de la base de datos.

- **REAL:** un valor en coma flotante.
- **BLOB:** datos binarios arbitrarios
- **INTEGER:** un entero con signo.

Una vez que la base de datos haya sido creada y posea tablas, se puede ingresar nueva información o actualizarla, para ello se cuenta con los siguientes métodos [29]:

- **SELECT:** esta instrucción permite realizar la consulta en la base de datos, posibilitando desplazarse a lo largo de las filas y columnas existentes en una tabla.
- **INSERT:** Esta sentencia permite añadir una nueva fila de datos a una Tabla de base de datos especificada, con el conjunto de valores respectivos para cada columna.
- **UPDATE:** esta sentencia permite modificar los datos de alguna fila con nuevos datos.

2.4 APIs para el reconocimiento de voz, procesamiento de imágenes y comunicaciones en Android.

2.4.1 API de reconocimiento de voz

Android provee acceso al reconocimiento voz por medio de un *Intent* el *RecognizerIntent*, con el uso de la constante *ACTION_RECOGNIZE_SPEECH*. Para iniciar la actividad de reconocimiento se debe usar el método ***startActivityForResult(Intent, int)*** que iniciara la actividad mostrando un mensaje a fin de indicar al usuario el momento en el que debe comenzar a hablar, una vez que termine de escuchar devolverá en resultado del reconocimiento mediante el método ***onActivityResult(int, int, Intent)*** [46].

Los parámetros extras que usa el *Intent* para el reconocimiento de voz son [46]:

- EXTRA_LANGUAGE_MODEL
- EXTRA_PROMPT
- EXTRA_LANGUAGE
- EXTRA_MAX_RESULTS
- EXTRA_RESULTS_PENDINGINTENT
- EXTRA_RESULTS_PENDINGINTENT_BUNDLE

Los resultados extra se devolverán con el parámetro [46]:

- EXTRA_RESULTS

2.4.2 API para detección de rostros

Para la detección de rostros Android provee la clase *Face*, que permite identificar la ubicación de un rostro dentro de un *Bitmap* [47].

Para detectar rostros utiliza el método ***findFaces*** de la clase ***FaceDetector***; dicho método *s* devolverá un arreglo de la clase *FaceDetector.faces[]* con el número total de rostros detectados [48].

Los métodos que se pueden utilizar para la clase *Face* se describen en la Tabla 18.

Método	Descripción
confidence()	Devuelve un factor de confianza en 0 y 1.
eyesDistance()	Devuelve la distancia entre los ojos.
getMidPoint(PointF point)	Coloca la posición del punto medio entre los ojos.
pose(int euler)	Devuelve la pose del rostro.

Tabla 18.- Método para la clase Face [47]

2.4.3. API de comunicación Bluetooth

Mediante esta API Android provee los recursos para soportar la creación de conexiones sobre el protocolo Bluetooth, permitiendo tanto la transmisión como recepción de datos. Las clases más importantes que se incluyen en esta API se describen en la Tabla 19.

Clase	Descripción
BluetoothAdapter	Representa el adaptador Bluetooth local, este será el punto de entrada para todas las interacciones Bluetooth.
BluetoothDevice	Representa el dispositivo Bluetooth remoto.
BluetoothSocket	Representa la interfaz del puerto de comunicación Bluetooth.
BluetoothClass	Describe las características y capacidades del dispositivo Bluetooth
BluetoothProfile	Una interfaz que representa el perfil Bluetooth. Un perfil Bluetooth es una especificación para la comunicación inalámbrica basada en la tecnología Bluetooth.
BluetoothHeadset	Provee soporte para el uso de auriculares Bluetooth.
BluetoothA2dp	Define la calidad de audio que se trasmitirá entre dispositivos por medio de una conexión Bluetooth.
BluetoothHealth	Representa la salud del dispositivo proxy que controla el estado de la conexión.

Tabla 19.- Métodos del Api Bluetooth [49]

Antes de hacer uso del adaptador Bluetooth es necesario otorgarle los permisos necesarios a la aplicación dentro del *Manifest* como se muestra en la Figura 38.

```
<manifest ... >
 <uses-permission android:name="android.permission.BLUETOOTH" />
</manifest>
```

Figura 38.- Permiso para el uso del Bluetooth

Una vez que se haya establecido el permiso requerido los pasos importantes a considerar son [49]:

- Obtener adaptador Bluetooth: para obtener el adaptador Bluetooth será necesario llamar al método ***getDefautlAdtapter*()** de la clase *BluetoothAdapter*.
- Habilitar adaptador Bluetooth: es necesario asegurarse que el dispositivo se encuentre encendido mediante la llamada ***isEnabled*()**, en el caso de que no sea así será necesario lanzar un *Intent* para el encendido del mismo y posteriormente llamar al método ***onActivityResult*()**.

Se puede obtener los dispositivos vinculados mediante el método ***getBondedDevices*()**. El protocolo Bluetooth es capaz de soportar la conexión tanto como Servidor o Cliente, para el caso de servidor serán necesarios los siguientes puntos [49]:

- Obtener un *BluetoothServerSocke*t mediante el llamado a ***listenigUsingRfcommWithServiceRecord(String, UUID),*** al cual se le pasa un *string* que será el nombre el servicio y el UUID que identifica de forma única y exclusiva el servicio.
- Empezar a escuchar las solicitudes de conexión llamando al método ***accept*().**
- Cuando sea necesario aceptar conexiones adicionales se llamara **a *close*().**

Para el caso de cliente será necesario [49]:

- Usar la clase *BluetoothDevise* a fin de obtener el puerto de conexión o *BluetoothSocket* llamando a ***createRfcommSocketToServiceRecord(UUID)***. Para este caso será necesario usar el mismo identificador de servicio o UUID que se usó en la creación del servidor.
- Establecer la conexión mediante la llamada *connect*().

Cuando la conexión ha tenido éxito será posible enviar o recibir datos mediante el uso de las clases *InputStream* y *OutputStream* respectivamente.

CAPITULO 3: VISIÓN POR COMPUTADOR EN DISPOSITIVOS MÓVILES

3.1 Introducción a la visión por computador

3.1.1 Introducción

La visión para el cerebro humano es relativamente fácil, lo cual nos determinar los objetos que nos rodean dentro de un ambiente tri-dimensional, posibilitando también distinguir entre formas, colores y contornos con relativa facilidad. Un gran porcentaje de psicólogos y neurólogos han tratado por décadas de determinar cómo es que nuestra visión funciona y como es que este percibe la ilusiones ópticas [50].

La visión por computador es en sí, la transformación de los datos capturados por una cámara en una nueva representación, como puede ser la transformación de la imagen a escala de grises. El cerebro humano toma las imágenes capturadas y las divide en varios canales para su interpretación, puesto que posee distintas partes dedicadas a interpretar segmentos de la información por separado, permitiéndole analizar únicamente las partes que considere importantes para su interpretación. El cerebro lleva a cabo tareas muy complejas para realizar asociaciones y realizar una interpretación correcta del medio en el que se encuentra [51].

En la visión artificial el computador recibe una matriz de datos desde la cámara o desde uno o más ficheros almacenados en discos, dicha matriz posee una gran cantidad de ruido por lo que otorga muy poca información útil, entonces el trabajo de la visión por computador convertir esa matriz de información ruidosa en datos útiles para su interpretación. Se debe considerar que los datos que representan la imagen pueden estar corruptos, conteniendo distorsión y/o ruido, que usualmente son agregados a la imagen por cambios que se originan en el ambiente tales como la iluminación, reflexión o movimiento, e incluso por los elementos mismos de captura como pueden ser defectos en los sensores o ruido eléctrico [51].

El problema más grande que presenta la visión por computador es que se posee una imagen en dos dimensiones como representación de un mundo en tres dimensiones, la misma imagen 2D puede representar una infinita cantidad de escenas tridimensionales [51].

Por ello en la visión por computador se trata de describir el mundo que vemos en una o más imágenes tratando de reconstruir las propiedades del objeto, tales como el contorno, iluminación y la distribución del color [50].

3.1.2 Aplicaciones de la visión por computador

Aunque el análisis de imágenes en la visión por computador es muy complejo y requiere grandes recursos de procesamientos, en la actualidad ya es usada en aplicaciones tales como [50]:

- Reconocimiento de caracteres
- Inspección de maquinaria
- Reconstrucción tridimensional de objetos
- Conteo de objetos
- Imágenes medicas
- Seguridad automotriz
- Detección de movimiento
- Captura de movimiento
- Vigilancia
- Reconocimiento de huellas dactilares y datos biométricos

Además de las ya listadas se siguen realizando estudios para extender la capacidad de la visión artificial, existiendo un sin número de aplicaciones en las cuales se puede aplicar la visión por computador.

3.1.3 Historia de la visión por Computador

La visión artificial se puede decir que nace en la antigüedad, con la intención de representar objetos tridimensionales sobre una superficie plana como una pintura. Los griegos alcanzaron grandes conocimientos en lo que a proyección geométrica se refería, tal es el caso de Tales de Mileto, quien a partir de sus conocimientos en geometría fue capaz de realizar la predicción de un eclipse solar, o calcular la altura de un pirámide simplemente basándose en la sombra que proyectaba [52].

Serían los pintores Italianos durante el renacimiento quienes llegarían a comprender de mejor manera la formación de imágenes, así como los primeros en estudiar la geometría requerida para reproducir correctamente los efectos del mundo que observaban [52].

La perspectiva en sí fue inventada por Fillippo Brunelleschi (1377-1446) quien fue un gran arquitecto del renacimiento temprano, entre las principales obras de él se encuentra la catedral Santa Maria de Fiore [52].

Artistas como Piero della Francesa (1415-1492), Leonardo da Vinci (1452-1519) y Albercht Dürer(1471-1528) serían los que realizarían estudios más profundos en la geometría de la proyección, estudios que durarían hasta el día de hoy [52].

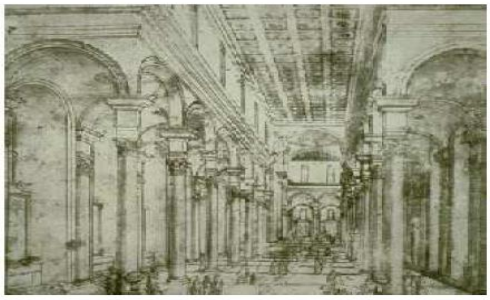


Figura 39.- Iglesia del Espíritu Santo, Bruneleschi. [52]

En el siglo XVI se desarrollaría la teoría de la perspectiva, creándose la máquina de perspectiva con el fin de ayudar a los pintores a reproducir exactamente la misma sin la necesidad de realizar cálculos matemáticos. En la Figura 40 podemos apreciar la máquina de perspectiva que se la puede considerar como el primer intento de crear una cámara [52]

Figura 40.-Máquina de perspectiva por Albrecht Dürer [52]

Sería en el año de 1545 cuando el astrónomo Germina Frisius publica un estudio en donde presentaría la cámara oscura, que mediante un orificio pequeño en una pared permitía pasar la luz externa, la misma que era proyectada sobre una de las paredes interiores de la cámara oscura, dando como resultado una imagen invertida del mundo exterior que brindaba la posibilidad a los pintores de realizar representaciones más precisas de la realidad [52].

Luego se introduciría el plano cartesiano por Descartes (1596-1650) quien concibió la geometría desde un punto de vista algebraico, logrando deducir que cualquier elemento se podía definir en el espacio mediante las coordenadas X,Y,Z [52].

En el año de 1826 el químico francés Niepce (1765-1833) llevó a cabo la primera fotografía, colocando una superficie fotosensible dentro de una cámara oscura a fin de conservar la imagen en ella. Luego en 1833 el químico francés Daguerre (1787-1851) hizo el primer proceso fotográfico práctico [52].

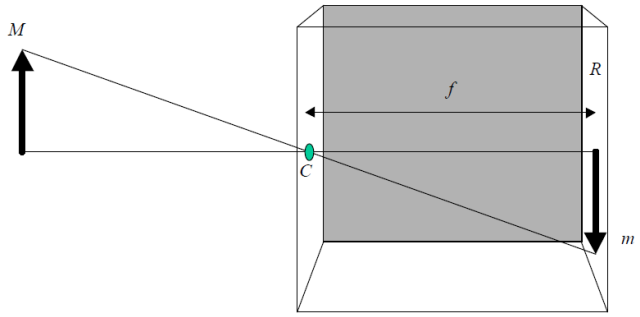


Figura 41.-Cámara Oscura [52]

La visión por computador como tal, nace en el año de 1970, cuando es considerada como el componente visual para imitar la inteligencia humana y dotar de este sentido a los robots. Los primeros pioneros en inteligencia artificial surgirían en el MIT, Standford y CMU [50].

Lo que distinguía a la visión artificial del procesamiento de imágenes ya existente en aquella época era el deseo de recuperar la representación tridimensional del mundo de las imágenes en dos dimensiones. En la Figura 42 se pueden apreciar los hitos más representativos durante la historia la visión artificial.

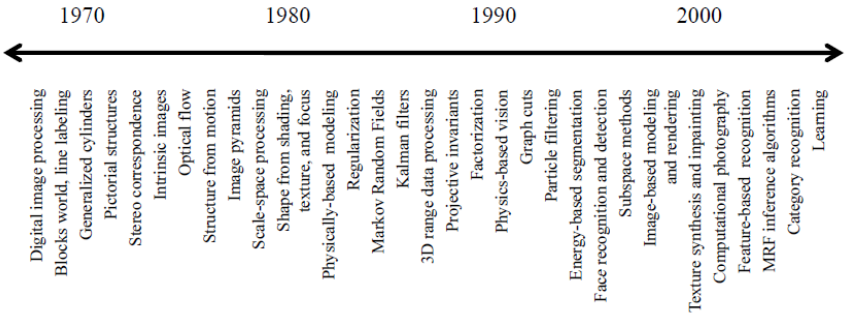


Figura 42.-Cronología de los eventos más importantes en la visión por computador [50]

3.1.4 Sistema de visión Humano

Para comprender de mejor manera la visión artificial, es necesario tener un concepto elemental cómo funciona nuestro sistema de visión y los elementos más importantes que lo conforman.

El ojo humano es uno de los órganos más complejos del cuerpo. Está conformado por una pared exterior y un contenido interno [53].

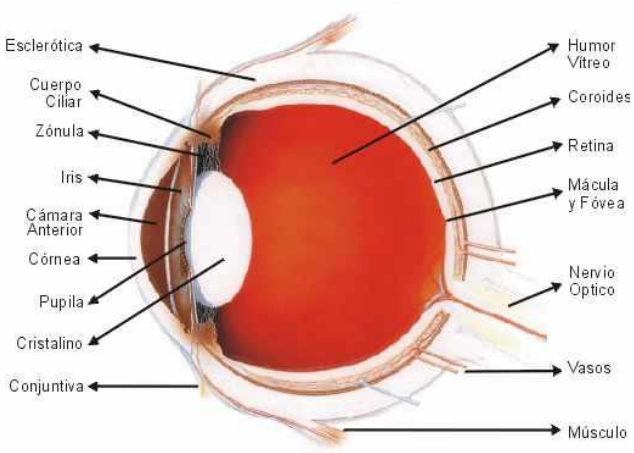


Figura 43.- Componentes del ojo humano [53]

Dentro de los elementos más importantes que conforman el ojo se tienen [54]:

- La córnea: es la parte anterior transparente de la capa del ojo, también es conocida como cristal.
- La esclerótica: la porción blanca posterior de la parte externa del ojo, que junto con la córnea forma la capa que protege al ojo.
- El iris: es el tejido pigmentado que se sitúa detrás de la córnea y delante del cristalino. Es el encargado de enfocar la luz en la retina.
- El lente cristalino: es un cuerpo que se encuentra suspendido detrás del iris.
- La cámara anterior: es el espacio existente entre la córnea y el iris, conteniendo en el humor acuoso que es un fluido transparente.
- La pupila: constituye la apertura que existe en el centro del iris en la parte transparente, está formada por un material blando y gelatinoso que cubre el ojo por la parte posterior del cristalino.
- La retina: es la capa interna del ojo, que posee las células foto sensibles, así como las fibras que se unen para formar el nervio óptico.
- El cuerpo ciliar: es la parte de la capa vascular ubicada entre el iris y la membrana coroides, además contiene los músculos de sostén y también se encarga de segregar el fluido acuoso.
- La membrana coroides: es la capa vascular intermedia, ubicada entre la retina y la esclerótica.
- La mácula: es una zona reducida de la retina encargada de proporcionar la visión clara central.
- El nervio óptico: está conformado por múltiples fibras nerviosas, siendo el encargado de transmitir las imágenes capturadas por la retina hacia el cerebro.

El ojo por sí solo no es capaz de interpretar lo que ve y, por ello es necesario que el cerebro procese la luz que ingresa al ojo. Cuando la luz golpea algún objeto, este refleja parte de la luz, la cual incide en el ojo y una vez allí pasa primero por la córnea, luego por el humor acuoso, la pupila, el cristalino, el gel vítreo y finalmente la retina [54].

La cornea y el cristalino son los encargados de refractar la luz y dirigirla hacia el la retina, la cual posee aproximadamente 127 millones de células foto sensibles llamadas conos y bastones [54].

Los conos y bastones absorben la luz y la convierten en una señal electroquímica que es transmitida al nervio óptico, para posteriormente enviarse hasta los lóbulos occipitales del cerebro, en donde se produce el proceso de visión como tal [54].

En el lóbulo occipital la imagen es procesada por el Cortex Visual Primario, que gracias a dos tipos de nervios es capaz de detectar mayor o menor movimiento. Para la interpretación de las imágenes el cerebro utiliza varias zonas, especializadas en extraer características de las mismas, dentro de las zonas destacan:

- Parietales: responden mejor a las modificaciones espaciales que pueda sufrir el objeto, tales como el tamaño, la forma o la rotación.
- Lóbulo Inferior temporal: responde a los colores, texturas y formas, además posee una zona dedicada al reconocimiento de rostros.
- Lóbulo Occipital: el encargado de separar los objetos del fondo.

El cerebro conjuntamente con el ojo realiza un trabajo impresionante, y con relativa facilidad al momento de comprender el mundo que lo rodea, siendo capaz de distinguir:

- Objetos similares a distancias distintas.
- Objetos a distancias similares pero de tamaños distintos.
- Objetos en distintas posiciones.
- Objetos parcialmente ocultos.

3.2 Librerías para implementación de algoritmos de visión por computador en Android.

Dentro de las librerías disponibles para la visón por computador en Android se encuentran ZXing y OpenCV, la primera es una librería de código abierto implementada en Java y utilizada para el procesamiento y detección de códigos de barras, en cambio OpenCV es una librería muy completa que contempla la mayoría de algoritmos conocidos para el procesamiento de imágenes. Por esa razón, esta sección tratará una introducción sobre OpenCV.

3.2.1 Introducción a OpenCV

OpenCV comenzó como una iniciativa de Intel para aplicaciones que hacían un uso intensivo de la CPU, la idea principal era crear un sistema de visión artificial universal, el cual estuviese disponible para todo mundo, de esta manera permitiría a los usuarios desarrollar nuevos algoritmos en base a los algoritmos ya existentes en el campo de la visión artificial [51].

La librería de OpenCV fue creada a fin de proporcionar una infraestructura única para aplicaciones de visión por computador, permitiendo una aceleración en el desarrollo de aplicaciones comerciales en lo que al aprendizaje máquina concierne [55].

La librería posee más de 2500 algoritmos optimizados, dichos algoritmos pueden ser utilizados para la detección y reconocimiento facial, identificación de objetos, detección de movimientos humanos en video, seguimiento de objetos, extracción de modelos 3D de objetos, seguimiento del movimiento de los ojos, etc. [55].

OpenCV es una librería que crece día a día con el aporte de miles usuarios dentro de la comunidad, además empresas bien consolidas como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda y Toyota hoy en día hacen uso de la misma para muchas aplicaciones prácticas [55]. OpenCV es compatible con interfaces como C, C++, Python, Java y Matlab, además puede ser ejecutada en con sistemas operativos como Windows, GNU/Linux, Android y Mac OS [55].

3.2.2 Integración de OpenCV en Android

OpenCV puede ser incluido dentro de un proyecto Android descargando la librería desde la página oficial y agregándola al proyecto como cualquier otra librería simplemente. Para ello, se debe descargarla, descomprimirla e importarla, en la Figura 44 se puede apreciar la importación de la librería a nuestro entorno de trabajo.

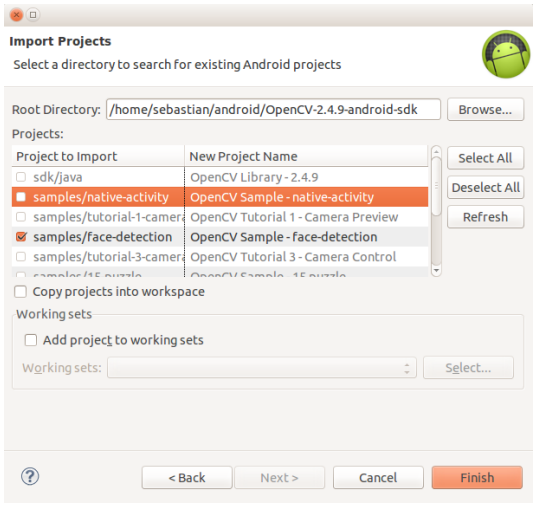


Figura 44.- Importación de la librería de OpenCV a Eclipse

Una vez que la librería ha sido importada a nuestro entorno de trabajo, podrá ser agregada a cualquier aplicación que requiera usarla como se muestra en la Figura 45.

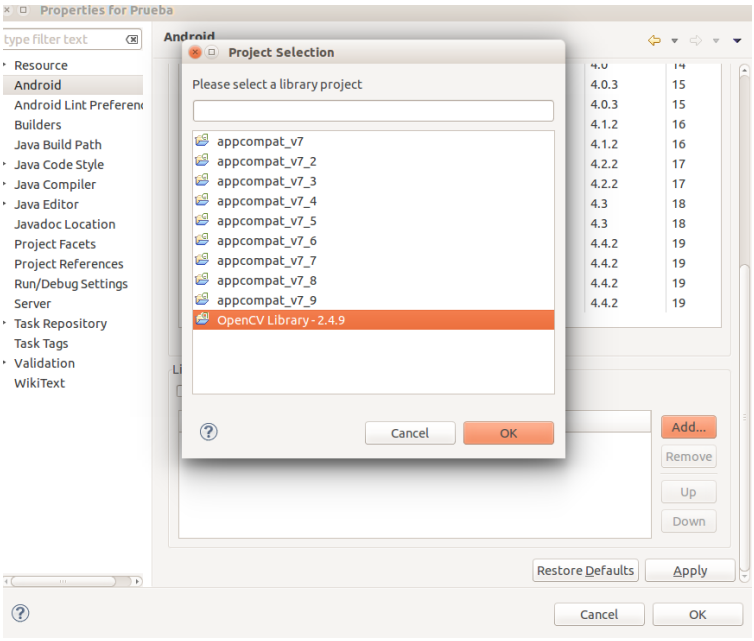


Figura 45.- Inclusión de la librería de OpenCV a un proyecto Android

Dentro de la carpeta de OpenCV no solo se incluye la librería, sino también ejemplos que pueden ser ejecutados desde Eclipse con la ayuda de algunos complementos que se detallaran más adelante, así como archivos de instalación (APK) listos para agregarse a cualquier dispositivo Android que posea una cámara.

3.2.3 NDK de Android

EL NDK es un compendio de librerías para programar y compilar aplicaciones en código nativo tales como C o C++, usado también para crear aplicaciones nativas en Android que usen OpenCV. En esta sección se tratará sobre la instalación del NDK en Eclipse usando como sistema operativo base Ubuntu 14.04 LTS.

Antes del uso de la librería de OpenCV es necesario tener instalado los siguientes paquetes [56]:

- JDK
- SDK de Android
- IDE Eclipse
- ADT Pluging
- NDK Pluging
- CDT

- GCC(GNU/Linux)
- Cygwin (Windows)

El CDT es el complemento para Eclipse necesario para desarrollar proyectos escritos parcial o totalmente en C/C++, este se instala automáticamente con el NDK Plugin, sino es así puede ser instalado de manera similar que el plugin ADT, agregando el repositorio del mismo en Eclipse. Para ello se accede a la opción "**Install new Software"**. La dirección del repositorio del CDT es http://download.eclipse.org/tools/cdt/releases/kepler.

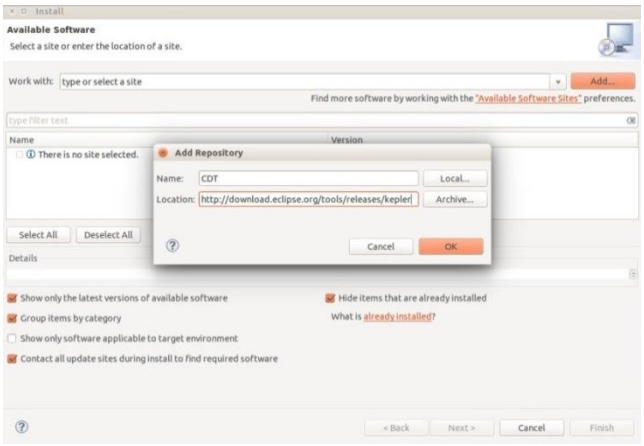


Figura 46.- Instalación del Plugin CDT en Eclipse

Una vez que se tengan todos los paquetes necesarios instalados se puede continuar con la instalación del NDK. Para ello será necesario descargar la última versión del mismo desde http://developer.android.com/tools/sdk/ndk/index.html, una vez descargado será extraído en la carpeta que se crea conveniente; para este caso se extrajo en la carpeta de Android dentro del directorio raíz como se muestra en la Figura 47.

```
HP-G42-Notebook-PC:~$ cd ~/android
HP-G42-Notebook-PC:~/android$ tar jxvf ~/Descargas/android-ndk-r9d-linux-x86.tar.bz2
```

Figura 47.- Extracción del NDK en el directorio raíz del sistema

Luego será necesario agregarlo al directorio ejecutable del sistema tal como se muestra en la Figura 48 [57].

```
HP-G42-Notebook-PC:~/android$ echo export ANDROID_NDK_HOME=~/android/android-ndk-r9d >>~/.bashrc
HP-G42-Notebook-PC:~/android$ echo export PATH=\$ANDROID_NDK_HOME:\$PATH >>~/.bashrc
```

Figura 48.- Comandos para agregar el NDK al directorio ejecutable del sistema

Si se desea, se puede comprobar que la instalación haya sido exitosa ejecutando el comando "ndk-build" desde el terminal, lo que debe dar de resultado lo que se muestra en la Figura 49 [57].

Figura 49.- Comprobar la instalación del NDK

Finalmente será necesario integrar el NDK al entorno de desarrollo Eclipse, para ello se deberá definir la variable de entorno NDKROOT que contendrá la ubicación del NDK dentro de nuestro sistema. Para ello, en Eclipse nos dirigimos **Window/Preferences** y seleccionamos **Android** y posteriormente **NDK**, en donde se indicará la ubicación del NDK dentro de nuestro sistema [58].

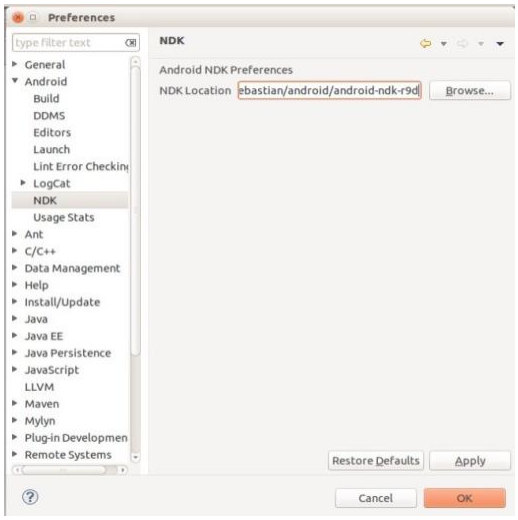


Figura 50.- Inclusión del directorio del NDK en Eclipse

El Android NDK contiene múltiples herramientas como compiladores, ensambladores, documentación, algunos de los componentes que este posee son [57]:

- ARM, x86, and MIPS cross-compilers
- Build system
- Java Native Interface headers
- C library
- Math library
- POSIX threads
- Minimal C++ library
- ZLib compression library
- Dynamic linker library
- Android logging library
- Android pixel buffer library

- Android native application APIs
- OpenGL ES 3D graphics library
- OpenSL ES native audio library
- OpenMAX AL minimal support

La estructura del NDK se organiza de la siguiente forma [57]:

- **Ndk-build:** es el punto inicial en el cual Android NDK compilará el sistema.
- **Ndk-gdb:** permite depurar componentes nativos utilizando el depurador GNU
- **Ndk-stack:** ayuda al análisis de los registros de pila y a descubrir cuando estos entran en conflicto.
- **Build:** es el directorio que contiene todos los módulos de entrada de compilación del sistema del Android NDK.
- **Plataforms:** es el directorio que contiene las cabeceras y archivos necesarios para soportar las distintas versiones de Android.
- **Samples:** es el directorio que contienes los ejemplos que demuestran la funcionalidad del Android NDK.
- **Sources:** es el directorio que contiene los módulos adicionales que cada desarrollador puede importar en su proyecto.
- **Toolchains:** este directorio contiene los compiladores cruzados (cross -compilers) para las distintas arquitecturas de dispositivos que el Android NDK es capaz de soportar.

3.2.4 Agregando el soporte Nativo a una aplicación Android

Para agregar el soporte nativo a una aplicación Android se puede utilizar el *wizard* que se instala en conjunto con el NDK, para ello se hará clic derecho sobre la aplicación a la que se desea agregar el soporte nativo y se navega hasta la pestaña **Tools,** para posteriormente seleccionar la herramienta **Add Native Support** en donde se ingresará el nombre de la librería dinámica que contendrá nuestro código nativo.

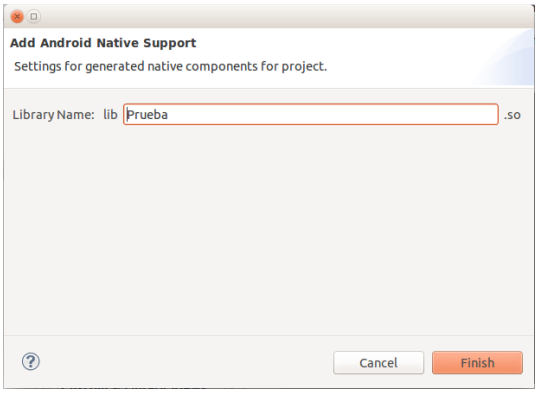


Figura 51.- Agregando soporte nativo a una aplicación Android

Una vez terminado el paso anterior, se agregarán a nuestro proyecto de manera automática los archivos necesarios para el soporte nativo, entre ellos destaca la interfaz JNI (Java Native Interface), necesario para la comunicación entre el código escrito en Java y el código nativo.

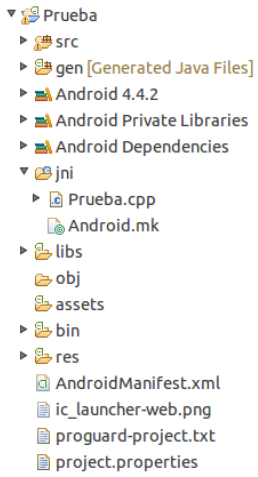


Figura 52.- Directorio del proyecto Android después de agregar el soporte nativo

Dentro del proyecto veremos los nuevos directorios [57]:

- *jni*: en esta carpeta contiene el código nativo de los componentes que la aplicación usará, así como el archivo **Android.mk** que describe cómo los componentes nativos serán compilados.
- *libs:* es el directorio creado por el proceso de compilación ejecutado por el Android NDK, dentro de él se contendrán los directorios para cada arquitectura de procesador especificado en el archivo de compilación.
- *obj:* este directorio contiene los archivos resultantes de la compilación del código fuente. Los archivos de este directorio no debe ser modificado por el desarrollador.

Para asegurarse que la compilación del proyecto se realice de manera adecuada es necesario asegurarse de agregar la ubicación del NDK dentro de las propiedades del proyecto, así como también de que comando de compilación sea el adecuado. El directorio del NDK es agregado a las variables del entorno ingresando a las propiedades del proyecto y navegando a la pestaña **C/C++ Build/Enviroment,** como se muestra en la Figura 53.

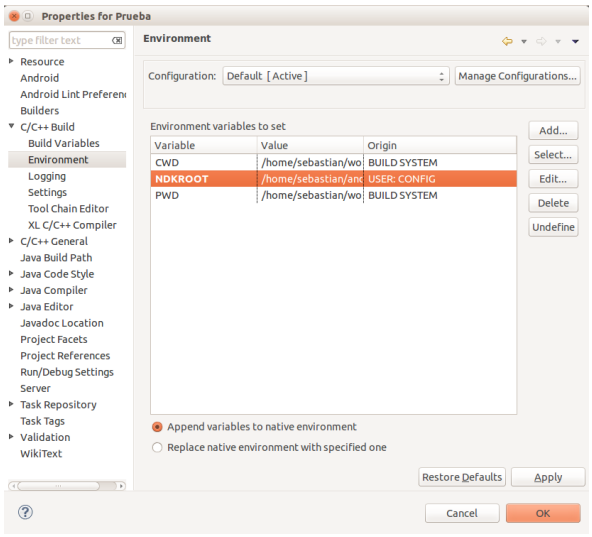


Figura 53.- Agregando la ubicación del NDK a las variables del entorno de Eclipse

Luego será necesario ingresar el comando de compilación navegando a **C/C++ Build** como se muestra en la Figura 54.

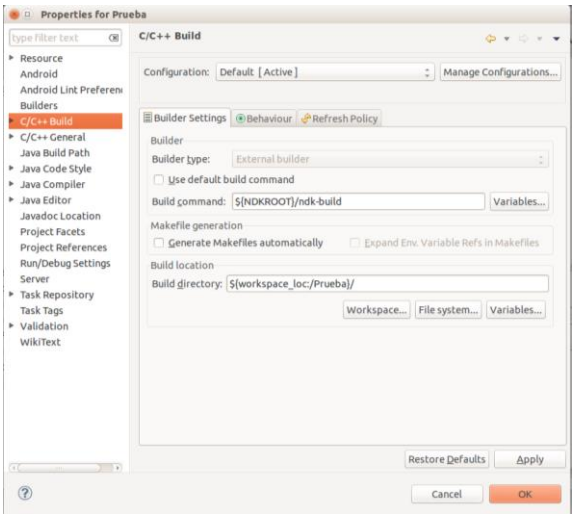


Figura 54.- Configuración de la variable de compilación del NDK en Eclipse

Finalmente se debe comprobar que las rutas y símbolos se encuentren incluidas navegando a la pestaña **C/C++ General /Paths and Symbols,** y luego en la pestaña **Includes** tal como se muestra en la Figura 55, si no es así será necesario agregar las rutas manualmente.

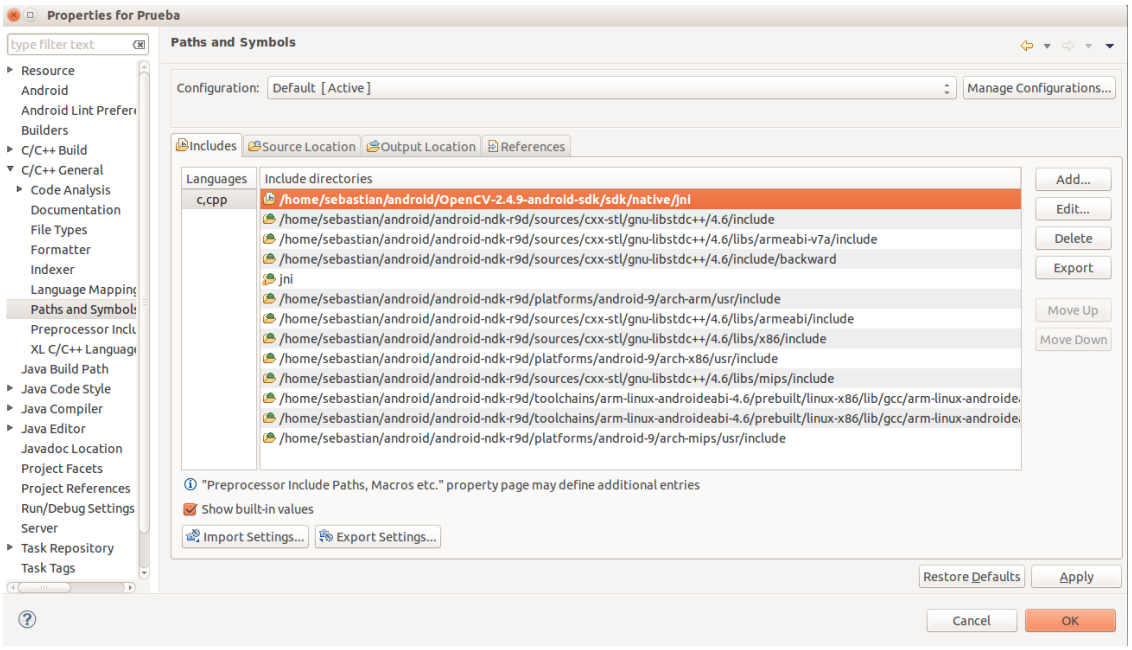


Figura 55.- Comprobación de la rutas y símbolos del NDK en Eclipse

Una vez que el entorno de trabajo esté configurado adecuadamente podremos empezar a agregar archivos en código nativo a la aplicación. Dentro de la carpeta JNI se podrán agregar o crear los archivos escritos en C o C++. Asimismo, dentro de esta carpeta encontraremos el archivo de compilación **Android.mk** y se podrá también agregar el archivo **Aplication.mk** que será en el cual se describirá las plataformas o arquitecturas de procesadores como ARM, MIPS o Intel ATOM para las cuales nuestro proyecto será compilado, así como la versión mínima de Android sé que soportará. En la Figura 56 se puede apreciar el contenido del archivo **Aplication.mk.**

```
APP_STL := gnustl_static
APP_CPPFLAGS := -frtti -fexceptions
APP_ABI := all
APP_PLATFORM := android-10
```

Figura 56.- Contenido del archivo Aplication.mk

El archivo **Aplication.mk** es un archivo de compilación opcional que es usado por el NDK, su propósito es describir qué módulos serán necesarios para la aplicación. Las variables que se pueden encontrar en el son [57]:

- APP_MODULES
- APP_OPTIM
- APP_CFLAGS

- APP_CPPFLAGS
- APP_BUILD_SCRIPT
- APP_ABI
- APP_STL
- APP_GNUSTL_FORCE_CPP_FEATURES
- APP_SHORT_COMANDS

El **Android.mk** es el archivo de compilación GNU que describe como el NDK compilará el proyecto para ser incluido como un archivo del sistema [57], un ejemplo del contenido de este documento se muestra en la Figura 57.

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)
LOCAL_SRC_FILES := Prueba.cpp
LOCAL_MODULE    := prueba_lib


include $(BUILD_SHARED_LIBRARY)
```

Figura 57.- Contenido del archivo Android.mk

La primera variable LOCAL_PATH es necesaria dentro del archivo Android.mk y es usada por el sistema de compilación de Android para localizar los archivos fuente, mediante *call my-dir* devolverá el directorio actual [57].

La variable CLEAR_VARS es usada por el sistema de compilación de Android para localizar el fragmento *clear_vars.mk*, que es necesario puesto que múltiples archivos de compilación pueden ser pasados por el sistema de compilación Android en una sola ejecución, por lo cual se debe siempre limpiar las variables para evitar conflictos [57].

La variable LOCAL_MODULE es usada para nombrar los módulos con un nombre único, este indicara el nombre de salida de la librería compilada, el sistema de compilación agregara el prefijo adecuado al archivo según corresponda [57].

La variable LOCAL_SRC_FILES incluirá todos los archivos fuente que serán compilados y ensamblados para generar el módulo final [57].

Dependiendo de la arquitectura de la aplicación múltiples librerías compartidas pueden ser generadas por un solo archivo **Android.mk** y para ello se usará BUILS_SHARED_LIBRARY. También es posible soportar librerías estáticas mediante BULD_STATIC_LIBRARY y cargarlas mediante LOCAL_STATIC_LIBRARIES. Asimismo puede ser usada con librerías compartidas, así como agregar librerías de terceros

mediante LOCAL_STATIC_LIBRARIES; la ventaja de usar librerías estáticas es que mantiene nuestro código de manera modular. [57]

Otras variables de compilación usadas por este archivo son [57]:

- *TARGET_ARCH*
- *TARGET_PLATAFORM*
- *TARGET_ARCH_ABI*
- *TARGET_ABI*
- *LOCAL_MODULE_FILENAME*
- *LOCAL_CPP_EXTENSION*
- *LOCAL_CPP_FEATURES*
- *LOCAL_C_INCLUDES*
- *LOCAL_CFLAGS*
- *LOCAL_CPP_FLAGS*
- *LOCAL_WHOLES_STATIC_LIBRARIES*
- *LOCAL_LDLIBS*
- *LOCAL_ALLOW_UNDEFINED_SYMBOLS*
- *LOCAL_ARM_MODE*
- *LOCAL_ARM_NEON*
- *LOCAL_DISABLE_NO_EXECUTE*
- *LOCAL_EXPORT_CFLAGS*
- *LOCAL_EXPORT_CPPFLAGS*
- *LOCAL_EXPORT_LDFLAGS*
- *LOCAL_EXPORT_C_INCLUDES*
- *LOCAL_SHORT_COMMANDS*
- *LOCAL_FILTER_ASM*

En este punto ya será posible agregar nuestros archivos escritos en C o C++ y compilarlos dentro de nuestro proyecto Android sin ningún problema, de esta manera podemos brindarle compatibilidad completa con la librería de OpenCV escrita en código nativo.

3.3 Reconocimiento de objetos y patrones en Android usando OpenCV.

3.3.1 Etapas para el reconocimiento de objetos

El reconocimiento de objetos y patrones es un proceso muy complejo dentro de la visión artificial, e implica el consumo de muchos recursos informáticos, así como el uso de varios algoritmos y etapas a fin de conseguir este objetivo. Dentro de las etapas a seguir para el reconocimiento de objetos están:

1. Adquisición: cualquier medio que nos permita adquirir las imágenes del mundo que nos rodea, como pueden ser las cámaras digitales.
2. Pre procesamiento: es el proceso posterior que se aplica a la imagen adquirida, con el fin de reducir el ruido debido al ambiente o factores ajenos a nuestro sistema durante el momento de la captura de la imagen.
3. Segmentación. Separar el objeto de interés del fondo en el que se encuentre.
4. Extracción de Características: consiste en el proceso por el cual, extraemos las características más relevantes del objeto, que pueden ayudar posteriormente a su identificación.
5. Reconocimiento: la etapa en la cual comparamos las características extraídas del objeto con características previas, a fin de confirmar que es o no es el objeto deseado.

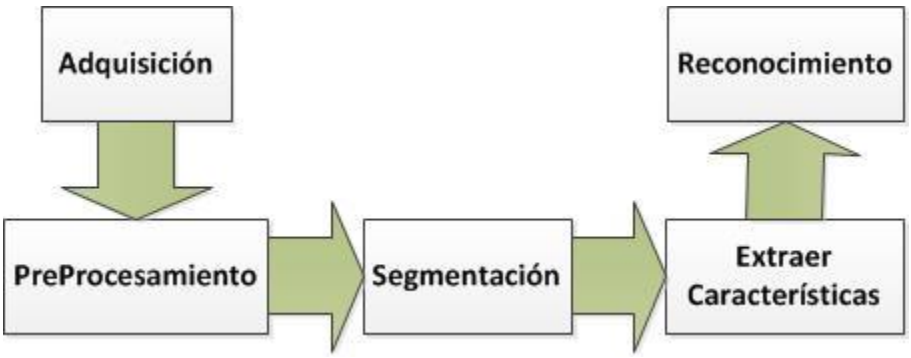


Figura 58.- Diagrama de bloque para el Reconocimiento de Objetos

Cabe mencionar que en esta sección se tratarán los métodos que se pueden utilizar para el reconocimiento de objetos en OpenCV utilizando C++, ya que los archivos escritos en este lenguaje pueden ser incluidos en un proyecto Android mediante la interfaz JNI. También es posible utilizar la API de OpenCV escrita en Java cuyos métodos no presentan una gran vaiación respecto a los escritos en C++. La documentación de esta API se encuentra disponible en: http://docs.opencv.org/Java/.

3.3.2 El espacio de Color

Un espacio de color se puede definir como la manera en la cual son representados los colores dentro de un sistema informático. Dentro de OpenCV se puede utilizar la función *cvtColor()* que nos permite cambiar de un espacio de color a otro, incluido entre ellos la escala de grises [59]. OpenCV cuenta con más de 150 conversiones para los distintos espacios de color [60].

Algunas de las conversiones que OpenCV permite son:

- **RGB:** El espacio de color RGB, es el más usado y está definido por tres canales Rojo(R), Verde(G), Azul(B); la combinación de los tres canales conforman los colores necesarios para representar una imagen

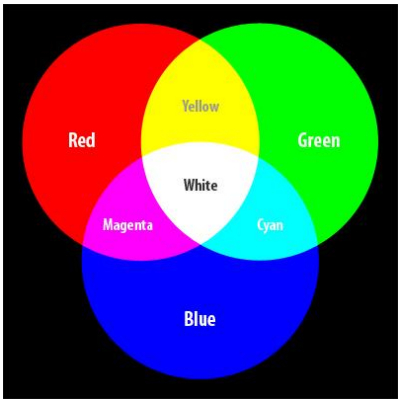


Figura 59.- Espacio de Color RGB [61]

- **HSV:** El espacio de color HSV (Hue Saturation Value) es una representación de los colores en tres canales, en donde *H* está definido como el matiz o color, *S* como la saturación y *V* es el brillo o intensidad.

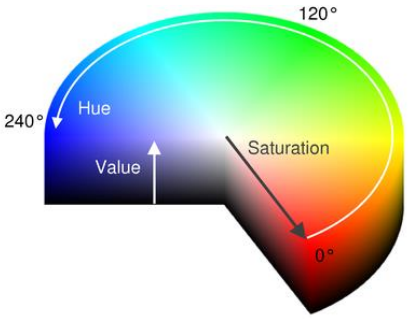


Figura 60.- Espacio de color HSV [62]

- **Lab:** o más conocido como CIELAB se basa en tres canales en donde *L* define la iluminación y, *a* y *b* definen el color. Este espacio de color es usado para describir todos los colores que puede percibir el ojo humano; la finalidad de este modelo es presentar una perspectiva lineal, es decir, ante un cambio de la misma cantidad en

un valor de color este debe producir un cambio casi de la misma importancia visual [63].

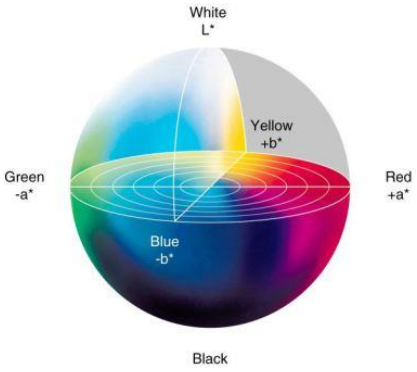


Figura 61.- Espacio de color LAB [63]

Dentro de los espacios de color se puede seleccionar los que más se adecúen al procesamiento posterior que se le vaya a realizar a la imagen.

3.3.3 Suavizado de la Imagen

El suavizado o desenfoque, es una técnica frecuente de procesamiento de imágenes, usualmente se emplea para la reducción del ruido dentro de la imagen. OpenCV provee varias técnicas de desenfoque todas ellas se encuentran al alcance mediante el método ***cvSmooth()***, los tipos de desenfoque o suavizado disponibles son [51]:

- Simple blur
- Simple blur no scaling
- Median blur
- Gaussian blur
- Bilateral filter

3.3.4 Morfología de imagen

OpenCV provee métodos muy rápidos para la transformación morfológica dentro de una imagen, las operaciones básicas son llamadas erosión y dilatación, que son métodos utilizados generalmente para [64]:

- Eliminación del ruido.
- Aislamiento de elementos individuales.
- Encontrará cambios abruptos de intensidad o agujeros en la imagen.

3.3.5 Dilatación y erosión

La dilatación es la convolución de la imagen con un kernel o mascara, que kernel puede ser de cualquier forma y tamaño. Un kernel B estará definido por un punto de anclaje (*anchor*

point) que usualmente es el centro del kernel. Cuando el kernel B es escaneado sobre la imagen, se calcula el valor máximo de todos los pixeles sobrepuestos por B, para luego reemplazar el pixel bajo el punto de anclaje con el máximo valor [64].

Figura 62.- Proceso de dilatación en una imagen [64]

La erosión calcula el mínimo local sobre el kernel, suponiendo que el kernel B es escaneado sobre la imagen, se calculará el valor mínimo del pixel sobrepuesto por B, siendo remplazado el pixel de la imagen bajo el punto de anclaje con el valor mínimo [64].

Figura 63.- Proceso de erosión en una imagen [64]

Los tipos de kernel que se pueden utilizar se muestran en la Tabla 20.

Kernel	Tipo de Kernel
CV_SHAPE_RECT	Rectangular
CV_SHAPE_CROSS	Transversal
CV_SHAPE_ELLIPSE	Elipse
CV_SHAPE_CUSTOM	Definido por el usuario

Tabla 20.- Tipos de Kernels usados para la erosión y dilatación de una imagen [64]

La erosión y dilatación podrán ser usadas en OpenCV mediante los métodos ***cvErode()*** y ***cvDilate()*** [51].

3.3.6 Segmentación

La segmentación es un método simple utilizado para la segmentación de objetos, que consiste en separar la imagen en dos regiones, el fondo y el objeto que se desea analizar. La separación está basada en la variación que existe entre los pixeles del objeto y los pixeles del fondo; una vez que se han separado los pixeles importantes, se les puede asignar un valor constante para luego identificarlos [65]

Figura 64.- Proceso de segmentacion en una imagen [65]

OpenCV dispone de la función ***cvThreshold ()*** que permite realizar la segmentación. Los tipos de segmentación que se disponen se muestran en la Tabla 21:

Tipo	Operación
CV_THRESH_BINARY	$dst_i = (src_i > T)?\, M : 0$
CV_THRESH_BINARY_INV	$dst_i = (src_i > T)?\, 0 : M$
CV_THRESH_TRUNC	$dst_i = (src_i > T)?\, M : src_i$
CV_THRESH_TOZERO_INV	$dst_i = (src_i > T)?\, 0 : src_i$
CV_THRESH_TOZERO	$dst_i = (src_i > T)?\, src_i : 0$

Tabla 21.- Tipos y Operaciones de segmentación disponibles en OpenCV [51]

En donde *dst*, es la matriz de origen, *src* la matriz de destino, *T* el umbral designado y *M* el nuevo valor de pixel que se le asignará.

3.3.7 Detección de bordes

Un detector de bordes eficiente fue desarrollado por John F. Canny en el año de 1986, también conocido como detector óptimo. Este algoritmo satisface tres criterios [66]:

- Baja tasa de error: buena detección de los bordes existentes.
- Buena ubicación: la menor distancia entre los pixeles del borde detectado y el pixel del borde real.
- Respuesta mínima: un solo detector responde por borde.

Los contornos son determinados aplicando *hysterisis threshold* a los pixeles, esto significa que existen dos umbrales, uno superior y otro inferior, si el pixel posee una gradiente más grande que el umbral superior es aceptado como un pixel que forma parte del borde, si el

pixel está por debajo del umbral menor este es rechazado [51]. El método que da acceso a la detección de bordes es ***cvCanny()***.

3.3.8 Aproximación poligonal

En el análisis de formas de una figura o de un contorno es común realizar una aproximación que represente el contorno mediante polígonos, de tal manera que se reduzca en gran medida el número de vértices que el contorno posee [51].

OpenCV permite realizar una aproximación poligonal de un contorno mediante la función ***cvApproxPoly()***, a este método se le puede pasar una lista o un árbol de contornos sobre los cuales se requiera realizar la aproximación [51].

Al método ***cvApproxpoly()*** se le debe pasar un argumento en donde se almacenarán los resultados; este método también acepta otros argumentos como CV_POLY_APPROX_DP que es la manera que efectuará la aproximación. También se le puede indicar la precisión que emplea el algoritmo [51].

En la Figura 65 se puede apreciar cómo trabaja el algoritmo para la aproximación poligonal.

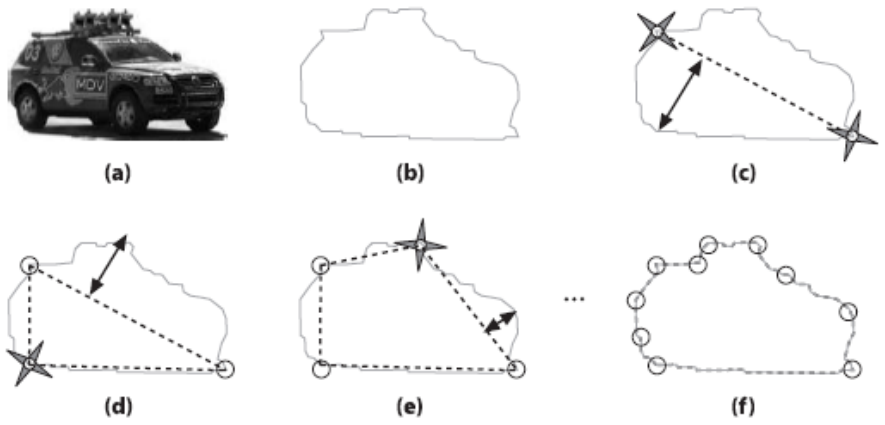


Figura 65.- Proceso para la aproximación poligonal [51]

El proceso funciona tomando dos puntos externos y conectándolos con una línea, luego se busca el punto más lejano a la línea, agregándolo a la aproximación para luego reajustar el polígono con el nuevo punto encontrado [51].

De forma similar a la aproximación poligonal OpenCV proporciona el método ***cvFindDominantPoints()***, que se encarga de determinar los puntos dominantes dentro de

un contorno, se considera un punto dominante como aquel que represente mayor información de una curva que los demás puntos.

3.3.9 Defectos de convexidad

Otra forma de comprender la forma de un objeto o un contorno es calcular los defectos de convexidad que estos posean. En la Figura 66 se puede apreciar el concepto de los defectos de convexidad usando una mano humana como ejemplo.

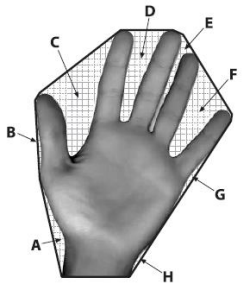


Figura 66.- Defectos de convexidad aplicado a una mano Humana [51]

OpenCV posee tres métodos para determinar la convexidad de un objeto. El primero nos permite determinar el defecto de un contorno que se le haya pasado, el segundo permitirá identificar si este defecto es convexo y el tercero calcula los puntos de convexidad en un contorno para el que se conoce la envolvente convexa [51].

Para el cálculo de los defectos de convexidad OpenCV implementa el método ***cvConvexHull()***, que toma un arreglo de puntos o un contorno como parámetro de entrada y el siguiente parámetro será el argumento en donde se almacenará el resultado [51].

Una vez se haya encontrado el contorno devuelto por el método ***cvConvexHull()***, se puede comprobar que sea convexo mediante el método ***cvCheckContourConvexity()*** [51].

Finalmente el método ***cvConvexityDefects()*** buscará los defectos , devolviendo una secuencia de los defectos encontrados. Para esto este método requiere el contorno determinado por el método ***cvConvexHull()*** y un parámetro donde se almacenarán los resultados [51].

3.3.10 Momentos de una imagen

Una de las maneras más rápidas para comparar dos imágenes o contornos es usar sus momentos, dichos momentos son calculados sumando todos los pixeles contenidos dentro de un contorno [51]. Los momentos se calculan como se muestran en la ecuación (1).

$$m_{p,q} = \sum_{i=1}^{n} I(x,y) x^p y^q \qquad (1)$$

En donde *p* es el orden de *x*, y *q* es el orden de *y*, se entiende como orden la potencia a la que es elevada el elemento correspondiente en la suma mostrada en la ecuación (1). Cuando *p* y *q* son cero se obtiene el momento m_{00} que es solamente la longitud en pixeles de contorno [51].

OpenCV nos permite calcular los momentos mediante la función ***cvMoments()***, la cual nos devolverá los momentos correspondientes al contorno analizado. De esta manera tendríamos los siguientes momentos [51]:

- Momentos espaciales:

$$m_{00}, m_{10}, m_{01}, m_{20}, m_{11}, m_{02}, m_{30}, m_{21}, m_{12}, m_{03}$$

- Momentos centrales:

$$mu_{20}, mu_{11}, mu_{02}, mu_{30}, mu_{21}, mu_{12}, mu_{03}$$

Los momentos describen características rudimentarias de un contorno que no son parámetros usados para comparación en casos prácticos. Para un análisis adecuado es necesario el uso de los momentos normalizados o llamados también "momentos invariantes de Hu" que como su nombre lo dice son invariantes a la traslación rotación y escalado [51].

Las funciones que tenemos a disposición en OpenCV son [51]:

- *CvMoments()*
- *cvGetCentralMoments()*
- *cvGetNormalizedMoments()*
- *cvGethuMoments()*

Los momentos centrales son calculados como se muestra en la ecuación (2).

$$\mu_{p,q} = \sum_{i=0}^{n} I(x,y)\left(x - x_{avg}\right)^p \left(y - y_{avg}\right)^q \qquad (2)$$

En donde $x_{avg} = m_{10}/m_{00}$ y $y_{avg} = m_{01}/m_{00}$.

Los momentos normalizados son calculados como se muestra en la ecuación (3).

$$\eta_{p,q} = \frac{\mu_{p,q}}{m_{00}^{(p+q)/2+1}} \qquad (3)$$

Finalmente, los momentos invariantes de Hu son una combinación lineal de los momentos centrales. La idea principal es que combinando los diferentes momentos normalizados y centrales es posible crear funciones invariantes, que representen los distintos aspectos de una imagen de tal manera que sea invariante al escalado, traslación y rotación [51]:

$$h_1 = \eta_{20} + \eta_{02}$$

$$h_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$h_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$h_4 = (\eta_{30} - \eta_{12})^2 + (\eta_{21} - \eta_{03})^2$$

$$h_5 = (\eta_{30} + 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) + (3\eta_{21} + \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)$$

$$h_6 = (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$h_7 = (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})((3\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)$$

3.3.11 La distancia Euclidiana

La distancia Euclidiana es usada generalmente como método para la comparación de características extraídas de un objeto. Se emplea para calcular la distancia entre dos puntos, primero en el plano y luego en el espacio, así como también para definir la longitud de la distancia entre puntos de otros tipos de espacios con más dimensiones [67]. Está basada en el teorema de Pitágoras sobre triángulos rectángulos, en donde la distancia es equivalente a la hipotenusa [67].

En el plano cartesiano la distancia euclidiana entre dos puntos $A = (x_a, y_a)$ y $B = (x_b, y_b)$, está definida como [67]:

$$d(A, B) = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \qquad (1)$$

En el espacio tridimensional la distancia euclidiana entre dos puntos $A = (x_a, y_a, z_a)$ y $A = (x_b, y_b, z_b)$ estará definida por [67]:

$$d(A,B) = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2} \qquad (2)$$

En un espacio de ***N*** dimensiones la distancia euclidiana entre puntos $A = (a_1; a_2; \ldots . a_n)$ y $B = (a_1; a_2; \ldots . a_n)$ será definida como [67]:

$$d(A,B) = \sqrt{\sum_{i=1}^{N} (b_i - a_i)^2} \qquad (3)$$

3.4 Reconocimiento facial en Android usando OpenCV.

3.4.1 Etapas para el Reconocimiento facial

En la actualidad existe una gran cantidad de algoritmos usados para el reconocimiento facial, cuyas etapas principales son:

1. Detección: es la primera etapa antes del reconocimiento; consiste en detectar si dentro de la imagen analizada existe un rostro humano.
2. Reconocimiento: una vez que se detecten los rostros dentro de una escena, el siguiente paso será compararlo con una base de datos de los rostros de personas "conocidas", a fin de determinar si existe alguna coincidencia.

Al igual que en la detección de objetos los temas tratados en esta sección se refieren a los métodos de OpenCV escritos totalmente para C++ que pueden ser implementados en la interfaz nativa JNI de Android o usando la API de Java.

3.4.2 Detección de rostros usando Haar Cascades

La detección de rostros u objetos utilizando clasificadores cascada *Haar* es un método efectivo propuesto por Paul Viola y Michael Jones en su publicación *"Rapid Object Detection using a Boosted Cascade of Simple Features"* (Detección rápida de objetos mediante usos de filtro cascada optimizado) en 2001. Es un aprendizaje máquina, en donde una función cascada es entrenada con una gran cantidad de imágenes positivas y negativas, para luego ser usada para identificar un objeto dentro de una imagen [68].

Inicialmente el algoritmo necesita un gran número de imágenes positivas (imágenes de rostros), e imágenes negativas (muestras que no contengan rostros) para entrenar el clasificador. Entonces será necesario extraer determinadas características, para esto se

utilizan las que se muestran en la Figura 67, dichas imágenes son el kernel de convolución usado para extraer las características. Cada característica obtenida es un valor único determinado a partir de la substracción de la suma de los pixeles bajo el rectángulo blanco y de la suma de pixeles bajo en rectángulo negro [68].

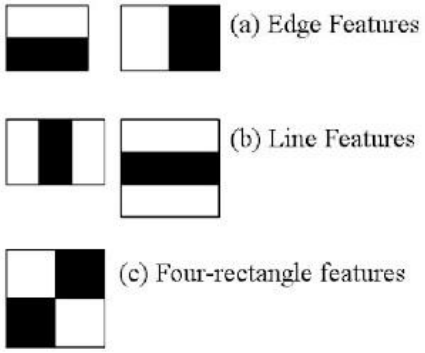


Figura 67.- Kernels de convolución para la extracción de características [68]

Ahora todos los posibles tamaños y ubicaciones de cada kernel son usados para calcular las características, esto requiere una gran cantidad de cálculos computacionales puesto que para cada cálculo será necesario encontrar la suma de los pixeles bajo el rectángulo negro y blanco. A fin de resolver este problema se introdujo las imágenes integradas, que simplifican el cálculo de la suma de pixeles [68].

Es importante considerar que no todas las características extraídas de un objeto son relevantes, así que para el caso del reconocimiento facial las más relevantes se encuentran archivadas en el repositorio "Adaboost" [68].

Para cada característica se busca el mejor umbral, de tal manera que se pueda clasificar un rostro como positivo o negativo, obviamente pueden existir errores en la clasificación, por lo cual es necesario extraer las características que posean un error de menor rango, de tal manera que permitan clasificar de forma más eficiente en donde existe y en donde no existe un rostro. El proceso es muy complicado puesto que al inicio cada imagen tendrá el mismo peso o valor, después de cada clasificación el peso de las imágenes mal clasificadas se incrementa, el proceso se vuelve a repetir calculando una nueva tasa de error, hasta que se obtiene la tasa de error y la precisión requerida [68].

El clasificador total será la suma ponderada de los clasificadores más débiles, ya que por sí solos no son capaces de clasificar una imagen. La configuración final tenía alrededor de 6000 características que podían identificar un rostro [68].

El proceso de clasificación por ejemplo toma una imagen con una ventana de 24X24 y aplica las 6000 características para buscar si existe un rostro o no, es un proceso lento que consume una gran cantidad de recursos y al final puede resultar que la mayor parte de una

imagen no incluya un rostro, por lo cual es mejor poseer un método sencillo para verificar si una ventana no contiene un rostro, lo descarte totalmente y no lo vuelva a procesar, permitiendo obtener más tiempo para procesar una región que si contenga un posible rostro [68].

Por la razón anterior nacen los clasificadores cascada, que en lugar de aplicar todas las 6000 características al principio en una ventana, aplican distintos números de características en cada etapa de la clasificación, de tal manera que en las primeras etapas se aplicarán muchas menos características. Si una ventana por ejemplo no supera la primera etapa no se la volverá a procesar [68].

El detector creado por los autores originales tenía más de 6000 características con 38 etapas, con 1, 10, 25 y 50 características en las primeras cinco etapas, según los autores en promedio 1 característica de las más de 6000 son evaluadas por sub-ventana [68].

3.4.3 Reconocimiento Facial

OpenCV provee varios métodos y algoritmos para el reconocimiento facial, esta sección se trataran de los más relevantes como son *Eigenfaces, FisherFaces y LBPH*.

3.4.3.1 EigenFaces

El problema de la representación de la imagen es su alta dimensionalidad, imágenes en escala de grises de dos dimensiones *m*x*n* , generan un vector de *d=mn* dimensiones, de ahí que no todas las dimensiones son igualmente útiles. Con ello, se realiza una selección si es que existe alguna variación significante en los datos a fin de buscar los componentes que brinden mayor cantidad de información. El Análisis de componentes Principales (PCA) fue propuesto independientemente por Karl Pearson y Harold Hotelling para identificar un conjunto de posibles variables correlacionadas dentro de un menor conjunto de variables no correlacionadas; la idea se basa en que un conjunto de datos de alta dimensionalidad está a menudo descrito por variables que están correlacionados, así que algunas dimensiones son significativas y representan la mayor parte de información [69].

Fundamento Matemático

Dado un conjunto de *N* muestras de imágenes $\{x_1, x_2 \ldots\ldots x_N\}$, tomando los valores en un espacio de imagen de n-dimensiones, se asumirá que cada imagen pertenece a una de las *c* clases $\{X_1, X_2 \ldots\ldots X_C\}$, ahora se debe considerar una transformación lineal, mapeando la imagen original del espacio de n-dimensiones en un nuevo espacio de m-dimensiones, en donde $m < n$. El nuevo vector de características $y_k \in R^d$ estará definido por la siguiente transformación lineal [70]:

$$y_k - W^T x_k \qquad k = 1,2, \ldots \ldots, N \qquad (1)$$

En donde $W \in R^{n \times m}$ es una matriz con columnas orto normales y la matriz de distribución S_T está definida por [70]:

$$S_T = \sum_{k=1}^{N} (x_k - u)(x_k - u)^T \qquad (2)$$

En donde c es el número de clases y $u \in R^n$ es la imagen media de todas las muestras, entonces después de aplicar la transformación lineal W^T, la dispersión de los vectores de características transformadas $\{y_1, y_2, \ldots . . y_N\}$ es $W^T S_T W$ [70]. En el análisis PCA, la proyección W_{opt} es elegida para maximizar el determinante de la matriz de dispersión total de las muestras proyectadas de la siguiente manera [70]:

$$W_{opt} = \arg max_w |W^T S_T W| = [w_1, w_2 \ldots \ldots w_m] \qquad (3)$$

En donde $\{w_i | i = 1,2, \ldots, m\}$ es el conjunto de vectores propios de n-dimensiones de S_T correspondiente a los valores propios más largos de m [70]. De esta manera los vectores propios tendrá la misma dimensión que las imágenes originales [70].

El método EigenFaces realiza el reconocimiento facial mediante [69]:

- Proyección de todas las muestras entrenadas dentro del sub espacio PCA.
- Proyección de la imagen a buscar en el sub espacio PCA.
- Buscando al vecino más cercano que se encuentre entre la imagen de entrenamiento proyectada y las posibles imágenes proyectadas

El uso de 10 vectores propios no es suficiente para una buena reconstrucción de la imagen, así que unos 50 vectores propios puede ser suficiente para codificar las características principales de un rostro [69].

3.4.3.2 FisherFaces

El problema con EigenFaces es que no considera factores variables como la iluminación [69].

El análisis discriminante lineal lleva acabo una reducción dimensional de una clase específica y fue inventado por el gran estadístico Sir R. A. Fisher, científico que logró con éxito la clasificación de flores en su publicación de 1936 [69]. A fin de encontrar la combinación de características que se separan mejor entre las clases, el análisis discriminante lineal maximiza la relación entre clases dentro de la dispersión de clases, en lugar de maximizar la dispersión global [69]. La idea básica es que las clases similares se deben agrupar juntas, mientras que las clases diferentes deben ser agrupadas lo más lejos

posible [69]. Esto también fue reconocido por Bellhumeur, Hesphana y Kriegman por lo que se aplicó este análisis discrimínate para el reconocimiento facial [69].

Fundamento Matemático

Siendo X un vector aleatorio con muestras tomadas de c clases [69]:

$$X = \{X_1, X_2, \dots\dots, X_c\} \qquad (1)$$

$$X_i = \{x_1, x_2, \dots\dots, x_n\} \qquad (2)$$

La matriz de dispersión S_By S_w se calcula mediante [69]:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T \qquad (3)$$

$$S_w = \sum_{i=1}^{c} \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T \qquad (4)$$

En donde u_i es la imagen media de la clase X_i y N_i es el número de la muestras en la clase X_i [70].

La media total de las imágenes μ estará definida como [69]:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i \qquad (5)$$

Y $\mu_i$es la media de la clase $i \in \{1, \dots\dots, c\}$ [69]:

$$u_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j \qquad (6)$$

El algoritmo clásico Fisher busca una proyección W, que maximiza la separación entre clases [69]:

$$W_{opt} = arg\ max_w \frac{|W^T S_B W|}{|W^T S_w W|} = [w_1, w_2 \dots\dots w_m] \qquad (7)$$

En donde $\{w_i | i = 1,2, \dots, m\}$ es el conjunto de vectores propios generalizados de S_B y S_w correspondiente al valor propio más largo generalizado de m $\{\lambda_i | 1,2, \dots m\}$, de esta manera tenemos [70]:

$$S_B v_i = \lambda_i S_w v_i \quad i = 1,2, \dots\dots, m \qquad (8)$$

Dentro del reconocimiento de rostros existe un problema con la matriz de distribución $S_w \in R^{n \times n}$ que resulta ser siempre singular [70]. Esto se deriva del hecho de que el rango de S_w es al menos igual a $N - c$, y el número de imágenes en el conjunto de entrenamiento N es mucho más pequeño que el número de pixeles en cada imagen n [70]. Esto significa que es posible escoger una matriz W de tal manera que la dispersión dentro de la clase de las muestras proyectadas pueda ser cero [70]. Partiendo de que S_w es singular se plantea el

método FisherFaces, que permite evitar el problema de la proyección del conjunto de imágenes en un espacio de menor dimensionalidad, dando como resultado que la matriz de dispersión dentro de la clase ya no es singular [70]. Esto se logra mediante el uso de la PCA con el fin de reducir el espacio de características a $N - c$, para luego aplicando la FLD definida por (7) a fin de reducir la dimensión a $c - 1$, la optimización W_{opt} está dada por [70]:

$$W_{opt}^T = W_{fld}^T W_{pca}^T \qquad (9)$$

En donde [70]:

$$W_{pca} = \arg max_W |W^T S_T W| \qquad (10)$$

$$W_{fld} = \arg max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|} \qquad (11)$$

Se puede notar que la optimización para W_{pca} se lleva a cabo con $n \times (N - c)$ matrices con columnas ortonormales, mientras que la optimización para W_{fld} se lleva a cabo con $(N - c) \times m$ matrices con columnas orto normales [70].

El análisis discriminante encuentra las características faciales para discriminar entre las personas [69]. El desempeño de FisherFaces dependerá de los datos de entrada, por ejemplo si aprendió imágenes con alta iluminación no podrá reconocer un rostro en una imagen de baja iluminación [69].

3.4.3.3 Local Binary Patterns Histograms

EigenFaces y FisherFaces utilizan un enfoque algo holístico[4] para el reconocimiento facial, en donde se trata la información como un vector en algún lugar de un espacio de alta dimensionalidad, pero se sabe que una alta dimensionalidad es mala, así que un sub espacio de menor dimensionalidad es identificado, en donde probablemente se conserva la información útil [69]. El método EigenFaces maximiza la dispersión total, ya que no todas las componentes con máxima variación sobre todas las clases no son necesariamente útiles para la clasificación, así que para conservar algo de la información útil se aplica un Análisis Discriminante lineal optimizado como se describe en el método FisherFaces [69].

En la vida real no se puede garantizar que la iluminación siempre será la misma, si suponemos que existe una sola imagen para cada persona las estimaciones de covarianza para el sub-espacio pueden calcularse mal y por lo tanto también la detección [69]. El

[4] Holístico.- Una forma de analizar la realidad como un todo

método EigenFaces posee una tasa de reconocimiento del 96% en la base de datos de rostros de AT&T [69]. En la Figura 68 se puede ver la comparación entre el método EigenFaces y FisherFaces para la base de datos de AT&T.

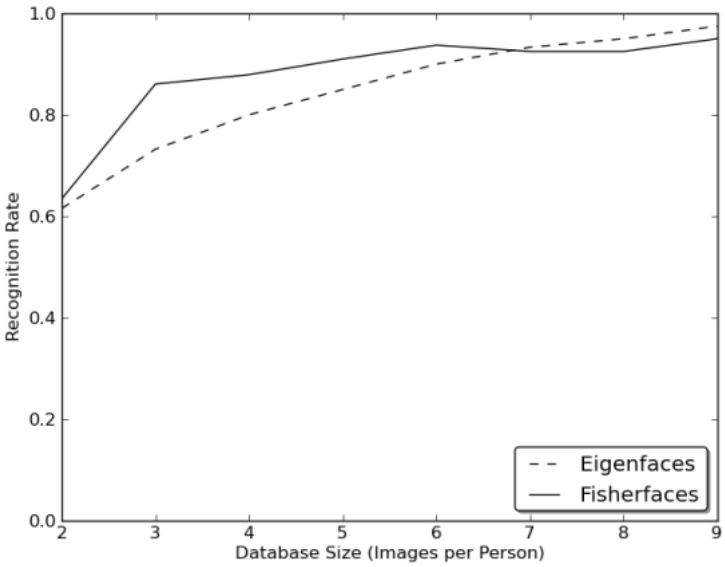


Figura 68.- Comparación entre la tasas de reconocimiento entre FisherFace y EigenFace [69]

Así que podemos ver que para un buen reconocimiento se necesita al menos 8(+-1) imágenes [69].

La investigación se centró en extraer las características locales de una imagen, la idea era no mirar toda la imagen como un vector de alta dimensionalidad, sino describir solamente la las características locales y representativas de un objeto [69]. Hay que considerar que la representación de la imagen no solo sufre cambios en iluminación sino también en escalado o en rotación, así que de manera similar a técnicas como SIFT, LBPH se basa en el análisis de texturas 2D [69]. La idea básica de LBPH es resumir la estructura local de una imagen comparando cada pixel con su pixel vecino, si la intensidad del pixel central es mayor o igual a su vecino se denotará con 1 caso contrario con un 0, de tal manera que se obtiene con un numero binario por cada pixel, por ejemplo 1100111, así que teniendo 8 pixeles alrededor existirán 2^8 combinaciones posibles, a las cuales se llamarán patrones binarios locales [69]. El primer operador LBP utiliza un barrido de 3X3 como se muestra en la Figura 69.

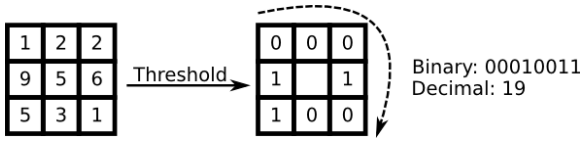


Figura 69.- Calculo de operador LBP para un barrido de 3X3 [69]

Fundamento Matemático

La descripción del operador LBP está dado por [69]:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c) \qquad (1)$$

En donde (x_c, y_c) es la posición del pixel central con intensidad i_c, e i_n la intensidad del pixel vecino. S es la función signo y se define como [69]:

$$s(x) = \begin{cases} 1 \; if \geq 0 \\ 0 \; else \end{cases} \qquad (2)$$

Esta descripción permite captura detalles muy pequeños dentro de las imágenes [69]. Se observó que el operador con un barrido fijo falla para decodificar detalles que difieren en escala, de tal manera que se extendió a usar un barrido variable. La idea es alinear un número variable de vecinos en un círculo con radio variable, que permite capturar los vecinos mostrados en la Figura 70 [69].

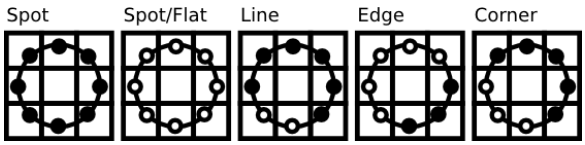


Figura 70.-Tipo de vecinos que se pueden determinar usando LBP [69]

Para un punto dado en la posición (x_c, y_c), la posición del vecino (x_p, y_p), en donde $p \in P$ puede ser calculado por [69]:

$$x_p = x_c + R\cos\left(\frac{2\pi p}{p}\right) \qquad (3)$$

$$y_p = y_c - R\sin\left(\frac{2\pi p}{p}\right) \qquad (4)$$

En dónde R es el radio del círculo y p es el número de puntos de muestra [69].

Este operador es una extensión de los códigos LBP, así que es llamado extended LBP [69]. Si los puntos de coordenadas en el círculo no corresponde a las coordenadas de la imagen, el punto puede ser interpolado, OpenCV utiliza una interpolación bilineal [69]:

$$f(x, y) \approx [1 - x \quad x] \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1 - y \\ y \end{bmatrix} \qquad (5)$$

Por definición el operador LBP es robusto en transformaciones monótonas que se encuentren en escala de grises [69].

El último paso es dividir la imagen en m regiones LBP y obtener un histograma, de ahí que los vectores de características mejorados son obtenidos mediante la concatenación de los histogramas locales, dichos histogramas se llaman *Local Binary Patterns Histograms* [69].

3.4.4 API de reconocimiento facial en OpenCV

Antes de empezar es importante considerar que el entrenamiento para el reconocimiento facial solo se puede realizar en un ordenador y no en un dispositivo móvil debido a los pocos recursos que este posee. Lo que sí se puede hacer en un dispositivo móvil es cargar un archivo de reconocimiento facial previamente entrenado.

Todos los modelos de reconocimiento facial están derivados de una clase abstracta basada en *FaceRecognizer*, que provee el acceso a todos los métodos de reconocimiento facial [71].

```
class FaceRecognizer : public Algorithm
{
public:
    //! virtual destructor
    virtual ~FaceRecognizer() {}

    // Trains a FaceRecognizer.
    virtual void train(InputArray src, InputArray labels) = 0;

    // Updates a FaceRecognizer.
    virtual void update(InputArrayOfArrays src, InputArray labels);

    // Gets a prediction from a FaceRecognizer.
    virtual int predict(InputArray src) const = 0;

    // Predicts the label and confidence for a given sample.
    virtual void predict(InputArray src, int &label, double &confidence) const = 0;

    // Serializes this object to a given filename.
    virtual void save(const String& filename) const;

    // Deserializes this object from a given filename.
    virtual void load(const String& filename);

    // Serializes this object to a given cv::FileStorage.
    virtual void save(FileStorage& fs) const = 0;

    // Deserializes this object from a given cv::FileStorage.
    virtual void load(const FileStorage& fs) = 0;
};
```

Figura 71.- Api de reconocimiento facial de OpenCV [71]

Descripción General

Dentro del algoritmo se provee varias clases que brindan una gran funcionalidad [71]:

- Se puede tener acceso a los métodos creando una instancia mediante ***Algorithm::create().***
- Se puede agregar u obtener parámetros al algoritmo por su nombre.

- Lectura y escritura de los parámetros en archivos en XML o YAML.
- El entrenamiento para un conjunto de imágenes mediante ***FaceRecognizer::train().***
- Predicción de una imagen de muestra, que puede contener un rostro.
- Cargar o guardar un modelo en archivos XML o YAML.

Ajuste de los umbrales

Existen situaciones en las que es necesario añadir un nivel de sensibilidad o confianza al momento de reconocer un rostro, para ello OpenCV no provee una clase directa para configurar este parámetro, que debe ser ingresado dentro del constructor ***FaceRecognizer()*** [71].

```
Ptr <FaceRecognizer> model = createEigenFaceRecognizer (num_components, umbral);
```

Figura 72.- Método para el ajustes de umbrales durante el entrenamiento para el Reconocimiento Facial [71]

Entrenamiento para el Reconocimiento Facial

El entrenamiento está dado mediante el método mostrado en la Figura 73.

```
void FaceRecognizer::train(InputArrayOfArrays src, InputArray labels)=0
```

Figura 73.-- Método para el entrenamiento del archivo para el Reconocimiento Facial [71]

En donde [71]:

- *Src*: son las imágenes de entrenamiento, es decir el rostro que se quiere aprender.
- *Labels:* la etiqueta que corresponde a cada grupo de imágenes.

Predicción de un Rostro

Los métodos están dados como se muestra en la Figura 74:

```
int FaceRecognizer::predict(InputArray src) const=0
void FaceRecognizer:predict(InputArray src, int& label, double& confidence) const=0
```

Figura 74.- Método para la predicción de un rostro [71]

En donde [71]:

- *Src:* la imagen en donde se realizara la predicción.
- *Label:* la etiqueta para la predicción de la imagen.
- *Cofidence:* el nivel de certeza de la predicción.

Guardando un archivo entrenado

El método está dado como se muestra en la Figura 75.

```
void FaceRecognizer:save(const String& Filename)
```

Figura 75.- Método para almacenar el archivo entrenado durante el Reconocimiento Facial [71].

En donde [69]:

- *Filename:* es el nombre de archivo para el reconocimiento facial ya sea en XML o YAML

Cagando un archivo entrenando

El método estará dado como se muestra en la Figura 76.

```
void FaceREcognizer::load(const String& fileneame)
void FaceRecognizer::load(const FileStorage fs)=0
```

Figura 76.- Métodos para cargar una archivo previamente entrenado durante el Reconocimiento Facial [71].

Este método cargará un archivo que haya sido almacenado en formato XML o YAML.

Crear un EigenFace Recognizer

El método estará dado como se muestra en la Figura 77.

```
Ptr<FaceRecognizer> createEigenFaceReognizer(int num_componets=0, double treshold=DBL_MAX)
```

Figura 77.- Método para crear un EigenFace Recognizer [71]

En donde:

- *Num_components:* el número de componentes que ha ser usados en el PCA.
- *Treshold:* el umbral aplicado a la predicción.

Es importante mencionar que el entrenamiento y la predicción se llevarán a cabo en escala de grises [71].

Crear un FisherFaces Recognizer

El método está dado como se muestra en la Figura 78.

```
Ptr<FaceRecognizer> createFisherFaceRecognizer(int num_components=0, ouble threshold=DBL_MAX)
```

Figura 78.- Método para crear un FisherFace Recognizer [71].

En donde:

- *Num_components:* el número de componentes utilizados para el Análisis Discriminante Linear.
- *Treshold:* el umbral aplicado a la predicción. Si la distancia al vecino más cercano es más grande que el umbral este método retorna -1.

Crear LBPH Face Recognizer

El método estará dado como se muestra en la Figura 79.

```
Ptr<FaceRecognizer> createLBPHFaceRecognizer(int radius=1, int neighbors=8, int grid_x=8, int grid_y=8, double threshold=DBL_MAX)
```

Figura 79.- Método para crear un LBPH Face Recognizer [71].

En donde [71]:

- *Radius:* el radio utilizado para crear el Patrón Binario Local Circular.
- *Neighbors:* número de puntos de muestras usados para crear el Patrón Binario Local Circular.
- *Grid_x:* el número de celdas en dirección horizontal.
- *Grid_y:* el número de celdas en dirección vertical.
- *Threshold:* el umbral de sensibilidad aplicado a la predicción. Si la distancia al vecino más cercano es más grande que el umbral este método retorna -1.

3.4.4 Cascade Classifier Training

OpenCV posee dos aplicaciones para entrenar filtros que pueden ser utilizados posteriormente para identificar cualquier objeto. Para el entrenamiento OpenCV posee los métodos *opencv_haartraining* y *opencv_traincascade*, la diferencia es que el segundo es una nueva versión que soporta entrenamientos mediante HAAR y LBP. LBP es mucho más rápido extrayendo las características que HAAR [72].

Algunas de las herramientas útiles para el entrenamiento son [72]:

- Opencv_createsamples: es usado para preparar el conjunto de imágenes positivas o muestras de entrenamiento en un formato soportado por *opencv_haartraining* y *opencv_traincascade*
- Opencv_perfomance: es utilizado para evaluar la calidad del clasificador.

Preparando la información de entrenamiento

Para el entrenamiento es necesario poseer dos tipos de datos, las imágenes positivas y las negativas. Las muestras negativas son imágenes aleatorias que no contendrán el objeto a identificar en ninguna parte de la imagen [72].

Las muestras positivas contendrá el objeto que se desea entrenar para la detección, se debe considerar que para el entrenamiento es necesario poseer un gran número de muestras positivas [72].

La creación de las muestras positivas se realiza mediante *opencv_create_samples* cuyos argumentos se muestran en la Tabla 22.

Parámetro	Descripción
-vec <vec_file_name>	Nombre del archivo de salida que contendrá las muestras positivas a entrenar.
-img <image_file_name>	Objeto Fuente de la Imagen.
-bg <background_file_name>	Contiene una lista de imagen que usará como fondo para distorsionar el objeto.
-num <number_of_samples>	Número de muestras positivas a generar.
-bgcolor <background_color>	El color del fondo; el color de fondo denotará un color transparente.
-bgthresh<background_color_threshold>	
-inv	Si se especifica el color será invertido.
-randiv	Se especifica si los colores son invertidos aleatoriamente.
-maxidev <max_intensity_deviation>	La desviación máxima en la intensidad de los pixeles de las muestras en primer plano.

-maxxangle <max_x_rotation_angle> **-maxyangle <max_y_rotation_angle>** **-maxzangle <max_z_rotation_angle>**	Los ángulos máximos de rotación que deben estar en radianes.
-show	Usada como opción de debug. Si es especificada mostrará cada muestra.
-w <sample_width>	El Ancho en pixeles de las muestras de salida
-h <sample_height>	El Alto en pixeles de las muestras de salida

Tabla 22.- Parámetros y Descripción para el método opencv_create_samples [72]

El entrenamiento Cascada

Una vez preparados los datos, el siguiente paso será entrenar el clasificador ya sea usando las aplicaciones *opencv_trainingcascade* u *opencv_haartraining.*

Los parámetros y argumentos para la aplicación *opencv_traincascade* se describen en la Tabla 23, 24, 25 y 26.

Argumentos comunes	Descripción
-data <cascade_dir_name>	Aquí será donde los clasificadores se almacenaran.
-vec <vec_file_name>	El archivo **vec** con las muestras positivas.
-bg <background_file_name>	Descripción de los archivos de fondo.
-numPos <number_of_positive_samples> **-numNeg <number_of_negative_samples>**	El número de muestras positivas y negativas a usar para cada etapa del entrenamiento.
-numStages <number_of_stages>	Número de estados cascada a ser entrenados
-precalcValBufSize <precalculated_vals_buffer_size_in_Mb>	El tamaño del buffer para los índices de función pre-calculados.
-baseFormatSave	Este argumento es usado actualmente para activar las características especiales Haar.

Tabla 23.- Argumentos Comunes y su descripción para el método opencv_traincascade [72].

Parámetros Cascada	Descripción
-stageType <BOOST(default)>	Tipos de estados.
-featureType<{HAAR(default), LBP}>	Tipo de características HAAR o LBP
-w <sampleWidth> **-h <sampleHeight>**	El tamaño en pixeles de las muestras de entrenamiento. Deben ser del mismo valor que los que se usó al crear las muestras para el entrenamiento.

Tabla 24.- Parámetros Cascada [72].

Parámetros del clasificador Potenciados	Descripción
-bt <{DAB, RAB, LB, GAB(default)}>	Tipo de optimizador de clasificador: DAB, RAB, LB, GAB.
-minHitRate <min_hit_rate>	La tasa mínima de retorno deseado para cada etapa del clasificador.
-maxFalseAlarmRate <max_false_alarm_rate>	La tasa máxima de falsa alarma deseada para cada etapa del clasificador.
-weightTrimRate <weight_trim_rate>	Especifica si se debe usar un recorte y su peso. Una elección adecuada es 0.95.
-maxDepth <max_depth_of_weak_tree>	La máxima profundidad en un árbol débil. Una elección adecuada es 1.
-maxWeakCount <max_weak_tree_count>	El conteo máximo de árboles débiles para cada una de las etapas de entrenamiento cascada.

Tabla 25.- Parámetros del Clasificador Potenciados [72].

Parámetros de características especiales Haar	Descripción
-mode <BASIC (default) | CORE | ALL>	Seleccionar el tipo de características HAAR que serán usadas en el entrenamiento. BASIC usará solo características verticales, y ALL empleará todas las características entre un ángulo vertical y un ángulo de rotación de 45 grados

Tabla 26.- Parámetros de caracteristicas especiales Haar [72].

LBP no posee parámetros [72].

CAPITULO 4: DISEÑO E IMPLEMENTACIÓN DE PLATAFORMA DE TRANSPORTE

4.1 Análisis de requerimientos funcionales

En esta sección se analizarán los requerimientos que debe poseer la plataforma, tanto en sus componentes mecánicos, como en los elementos de hardware y software necesarios para que cumpla con los objetivos planteados en este proyecto. Esta sección es de suma importancia puesto que en ella se basa el desarrollo del resto del proyecto y las funcionalidades finales que el autómata terminado podrá implementar.

4.1.1 Requisitos mecánicos

La plataforma que soportará los dispositivos móviles inteligentes debe poseer varios elementos que le brinden la movilidad y recursos necesarios para que realice las tareas previstas. Dentro de los elementos más importantes se puede mencionar:

- Un soporte para teléfonos inteligentes, capaz de acoplarse a todos los tamaños existentes de los mismos. Este elemento debe poseer el suficiente torque para levantar los dispositivos inteligentes más pesados existentes en el mercado actual (2014).
- Una pinza que sea capaz de manipular un número determinado de objetos, con varios grados de libertad.
- Una base que soportará los elementos como la pinza, la cabeza, baterías, placa de control, sensores, servomotores y demás componentes de ensamblaje.
- Los elementos como ruedas o llantas que conjuntamente con los motores le brindarán la movilidad total a la plataforma sobre superficies lisas.

4.1.2 Requisitos de hardware

En los requisitos de hardware destacan todos los componentes electrónicos que debe poseer la plataforma, dentro de ellos tenemos:

- La placa de control encargada del protocolo de comunicación, así como del control de servomotores, motores y adquisición de datos desde los sensores.
- Servomotores necesarios y adecuados de acuerdo a las necesidades mecánicas.
- El hardware debe estar construido con las normas eléctricas adecuadas, de tal manera que sea capaz de trabajar en ambientes ruidosos.
- Poseer sensores de distancia sobre la plataforma para permitirle una reconstrucción bidimensional del ambiente en el que se encuentre.
- Un sensor de fuerza o presión para determinar la fuerza que aplica la pinza sobre el objeto que se sujete.

4.1.3 Requisitos de software

En este punto se debe considerar el software que se desarrollara para el micro controlador y que formará parte de la placa de control de la plataforma. Las características que debe poseer el software de microcontrolador son:

- Un protocolo de comunicación robusto, con la capacidad de detectar y evitar fallas, que permita el control total de todos los elementos de la plataforma de manera sencilla y rápida.
- La capacidad de manejar los datos mostrados en un pantalla LCD
- La capacidad de manejar adecuadamente los datos recibidos del dispositivo móvil y reaccionar de acuerdo a lo previsto.
- Controlar en tiempo real los elementos como servomotores y servomotores de rotación continua.
- Realizar las adquisiciones adecuadas de todos los sensores que posea la plataforma.

4.1.4 Dispositivos seleccionados para el desarrollo y pruebas de aplicaciones Android

En esta sección se detallan las características de los smarthphones que se utilizaron para el desarrollo y prueba de las distintas aplicaciones que se realizaron durante el desarrollo de esta tesis. Por ello, es importante mencionar que es de gran importancia en el desarrollo de aplicaciones móviles comprobar su funcionamiento en diversos dispositivos móviles, a fin de comprobar que las mismas sean compatibles y puedan ejecutarse en una amplia gama de dispositivos y versiones Android existentes en el mercado.

- **Huawei U8180**

Figura 80.- Smartphone Huawei U8180 [73]

Red	GSM 850 / 900 / 1800 / 1900 - HSDPA 900 / 2100
Navegación	EDGE 3G HSDPA 7.2Mbps
Pantalla	240 x 230 pixeles, 2.8 pulgadas
Dimensiones	104x56x13mm
Peso	100 g
Sensores	Acelerómetro para auto rotación Controles sensibles al tacto Sensor de proximidad para auto apagado
Memoria RAM	256 MB
Memoria ROM	512 MB
Procesador	Qualcomm MSM7225
Velocidad de Procesador	528 MHz
Sistema Operativo	Android, v2.2 Froyo
Cámara	Cámara trasera de 3.15 MP
Wi-fi	Wi-Fi 802.11 b/g/n
Bluetooth	Bluetooth v2.1 A2DP, EDR
GPS	GPS con soporte A-GPS

Tabla 27.- Características Huawei U8180 [73]

- **Samsung Galaxy S3 mini**

Figura 81.- Smartphone Samsung Galaxy S3 mini [74]

Red	GSM 850 / 900 / 1800 / 1900 - HSDPA 900 / 1900 / 2100
Navegación	EDGE 3G HSDPA, 21Mbps HSUPA, 5.76Mbps
Pantalla	480 x 800 pixels, 4.0 pulgadas
Dimensiones	121.6 x 63 x 9.9 mm
Peso	112 g
Sensores	Sensor acelerómetro para auto rotación Controles sensibles al tacto
Memoria RAM	1 GB
Memoria ROM	8 GB
Procesador	Dual-Core ARM Cortex-A9
Velocidad de Procesador	1.2 GHz
Sistema Operativo	Android OS, v4.2.2 Jelly Bean
Cámara	Cámara trasera 5 MP, 2592x1944 pixels, autofocus, flash LED, geo-tagging, detección de rostro, foco táctil, video 720p a 30fps, cámara frontal VGA.
Wi-fi	Wi-Fi 802.11 a/b/g/n; DLNA; Wi-Fi Direct; banda dual
Bluetooth	Bluetooth v4.0 A2DP, LE, EDR
GPS	GPS con soporte A-GPS

Tabla 28.- Características samnsung Galaxy S3 mini [74]

- **Samsung Galaxy S3**

Figura 82.- Smartphone Samsung Galaxy S3 [75]

Red	GSM 850 / 900 / 1800 / 1900 - HSDPA 850 / 900 / 1900 / 2100 - LTE
Navegación	EDGE Clase 12 3G HSDPA 21Mbps HSUPA 5.76Mbps 4G HSPA+ LTE
Pantalla	720 x 1280 pixels, 4.8 pulgadas
Dimensiones	136.6 x 70.6 x 8.6 mm
Peso	133 g
Sensores	Sensor acelerómetro para auto rotación Barómetro Controles sensibles al tacto Sensor de proximidad para auto apagado Sensor giroscópico
Memoria RAM	1 GB
Memoria ROM	16GB/32GB/64GB
Procesador	Procesador Exynos 4 Quad quad-core 1.4 GHz, GPU Mali 400MP
Velocidad de Procesador	1.4 GHz
Sistema Operativo	Android OS, v4.0.4 Ice Cream Sandwich
Cámara	Trasera de 8 MP, 3264x2448 pixels, autofocus, flash LED, geo-tagging, detección de rostro y sonrisa, foco táctil, estabilizador de imagen, video 1080p a 30fps, cámara frontal 1.9MP 720p a 30fps
Wi-fi	Wi-Fi 802.11 a/b/g/n; Wi-Fi Direct; DLNA; Wi-Fi Direct
Bluetooth	Bluetooth v4.0 A2DP, EDR
GPS	GPS con soporte A-GPS

Tabla 29.- Características Samsung Galaxy S3 [75]

- **Samsung Galaxy Note 3**

Figura 83.- Smartphone Samsung Galaxy Note 3 [76]

Red	GSM 850 / 900 / 1800 / 1900 - HSDPA 850 / 900 / 1900 / 2100 - LTE Cat. 4
Navegación	EDGE 3G HSDPA/HSUPA 4G LTE
Pantalla	1920 x 1080 pixels, 5.7 pulgadas
Dimensiones	151.2 x 79.2 x 8.3mm
Peso	168 g
Sensores	Sensor acelerómetro para auto rotación Controles sensibles al tacto Sensor de proximidad para auto apagado Sensor giroscópico Barómetro Sensor geomagnético Sensor de temperatura y humedad
Memoria RAM	3 GB
Memoria ROM	32GB/64GB memoria interna,
Procesador	Procesador Exynos octa-core 1.9 GHz(versión 3G), Qualcomm Snapdragon 800 quad-core 2.3GHz(versión LTE)
Velocidad de Procesador	1.9 GHz, 2.3 GHz
Sistema Operativo	Android OS, v4.3 Jelly Bean
Cámara	Frontal de 13 MP, autofocus, flash LED, BSI, geo-tagging, detección de rostro y sonrisa, foco táctil, estabilizador de imagen, video 1080p, cámara frontal 2MP
Wi-fi	Wi-Fi 802.11 a/b/g/n/ac; DLNA; Wi-Fi Direct; banda dual
Bluetooth	Bluetooth v4.0 LE
GPS	GPS con soporte A-GPS; GLONASS

Tabla 30.- Características Samsung Galaxy Note 3 [76]

- **Samsung Galaxy Tab 2, 7.1 pulgadas**

Figura 84.- Tablet Samsung galaxy Tab 2 7.1 [77]

Red	GSM 850 / 900 / 1800 / 1900 - HSDPA 900 / 1900 / 2100
Navegación	EDGE 3G HSDPA 21Mbps/ HSUPA 5.76Mbps
Pantalla	600 x 1024 pixels, 7.0 pulgadas
Dimensiones	193.7 x 122.4 x 10.5 mm
Peso	344 g
Sensores	Sensor acelerómetro para auto rotación Sensor giroscópico de tres ejes Sensor de proximidad para auto apagado
Memoria RAM	1 GB
Memoria ROM	8GB/16GB/32GB memoria interna
Procesador	ARM Cortex A8 dual core
Velocidad de Procesador	1 GHz
Sistema Operativo	Android OS, v4.0 Ice Cream Sandwich
Cámara	Trasera de 3.15 MP, 2048x1536 pixels, autofocus, geo-tagging, detección de sonrisa, video 1080p a 30fps, cámara frontal VGA
Wi-fi	Wi-Fi 802.11 a/b/g/n, DLNA, Wi-Fi Direct, banda dual
Bluetooth	Bluetooth v3.0 A2DP, HS
GPS	GPS con soporte A-GPS; GLONASS

Tabla 31.- Características Samnsung Galaxy tab 2 7.1 [77]

- **Samsung Galaxy Tab 2, 10.1 pulgadas**

Figura 85.- Tablet Samnsung Galaxy tab 2 10.1 [78]

Red	GSM 850 / 900 / 1800 / 1900 - HSDPA 850 / 900 / 1900 / 2100
Navegación	EDGE Clase 12 3G HSDPA 21Mbps/ HSUPA 5.76Mbps
Pantalla	800 x 1280 pixels, 10.1 pulgadas
Dimensiones	256.6 x 175.3 x 9.7 mm
Peso	588 g
Sensores	Sensor acelerómetro para auto rotación Sensor giroscópico de tres ejes Controles sensibles al tacto
Memoria RAM	1 GB
Memoria ROM	16GB/32GB memoria interna
Procesador	Procesador Cortex-A9 dual core
Velocidad de Procesador	1 GHz
Sistema Operativo	Android OS, v4.0 Ice Cream Sandwich
Cámara	Cámara trasera de 3.15 MP, 2048x1536 pixels, autofocus, geo-tagging, video 1080p, cámara secundaria VGA
Wi-fi	Wi-Fi 802.11 a/b/g/n, DLNA, Wi-Fi Direct, banda dual
Bluetooth	Bluetooth v3.0 A2DP, EDR
GPS	GPS con soporte A-GPS

Tabla 32.- Características Samsung Galaxy Tab 2 10.1 [78]

- **Samsung Galaxy Tab Pro 10.1 pulgadas**

Figura 86.- Samsung Galaxy Tab pro 10.1 [79]

Red	GSM 850 / 900 / 1800 / 1900 - HSDPA 850 / 900 / 1900 / 2100 - LTE 800 / 850 / 900 / 1800 / 2100 / 2600 (opcional)
Navegación	3G HSDPA 42Mbps / HSUPA 5.76Mbps 4G LTE (opcional)
Pantalla	2560 x 1600 pixels, 10.1 pulgadas
Dimensiones	243.1 x 171.4 x 7.3 mm
Peso	477 g
Sensores	Sensor acelerómetro para auto rotación Sensor giroscópico de tres ejes Sensor geomagnético
Memoria RAM	2 GB
Memoria ROM	16GB/32GB
Procesador	Procesador octa core Exynos 5420 1.9GHz x 4 + 1.3GHz x 4 (versión 3G), Qualcomm Snapdragon 800 quad-core 2.3GHz (versión LTE)
Velocidad de Procesador	1.9-1.3-2.3 GHz
Sistema Operativo	Android OS, v4.4 KitKat
Cámara	Cámara trasera de 8MP, 3264x2448 pixels, autofocus, flash LED, geo-tagging, video 1080p@60fps. Cámara frontal 2MP 1080p
Wi-fi	Wi-Fi 802.11 a/b/g/n/ac, Wi-Fi Direct, banda dual
Bluetooth	Bluetooth v4.0 A2DP
GPS	GPS con soporte A-GPS y GLONASS

Tabla 33.-Características Samsung Galaxy Tab pro 10.1 [79]

- **LG Nexus 4**

Figura 87.- LG Nexus 4 [80]

Red	GSM 850 / 900 / 1800 / 1900 - HSDPA 850 / 900 / 1700 / 1900 / 2100
Navegación	EDGE 3G HSPA+ 42Mbps / HSDPA 21Mbps / HSUPA 5.76Mbps
Pantalla	768 x 1280 pixels, 4.7 pulgadas
Dimensiones	133.9 x 68.7 x 9.1 mm
Peso	139 g
Sensores	Sensor acelerómetro para auto rotación Controles sensibles al tacto Sensor de proximidad para auto apagado Sensor giroscópico
Memoria RAM	2 GB
Memoria ROM	8GB/16GB memoria interna
Procesador	Procesador quad-core Qualcomm Snapdragon APQ8064 1.5 GHz, GPU Adreno 320
Velocidad de Procesador	1.5 GHz
Sistema Operativo	Android OS, v4.2 Jelly Bean
Cámara	Cámara trasera de 8 MP, 3264 x 2448 pixels, autofocus, flash LED, foco táctil, detección de rostro y sonrisa, estabilizador de imagen, Photo Sphere, geo-tagging, video 1080p@30fps Cámara frontal 1.3MP 720p
Wi-fi	Wi-Fi 802.11 a/b/g/n; DLNA; Wi-Fi Direct; banda dual
Bluetooth	Bluetooth v4.0 A2DP
GPS	GPS con soporte A-GPS y GLONASS

Tabla 34.- Características LG Nexus 4 [80]

4.2 Selección e implementación del protocolo de comunicación con el dispositivo móvil

Para la selección del protocolo de comunicación se consideraron algunos aspectos, como el desgaste físico en el caso de utilizar el protocolo a través de cable USB, la facilidad de implementación, la velocidad de transmisión, entre otros. Por ello se escogió un protocolo de comunicación inalámbrica basado en tecnología Bluetooth, que ya es un protocolo estandarizado y que al mismo tiempo brinda excelentes alternativas para la implementación dentro de dispositivos Android, además se evita el desgaste físico que puede surgir por usar medios como el USB.

4.2.1 Selección del módulo bluetooth

Para brindarle a la plataforma la capacidad de comunicarse inalámbricamente por medio de protocolo Bluetooth se seleccionó un módulo Bluetooth-serial HC-06 que puede ser acoplado a cualquier micro controlador, Arduino, u otro dispositivo con puerto serial con niveles lógicos TTL/CMOS.

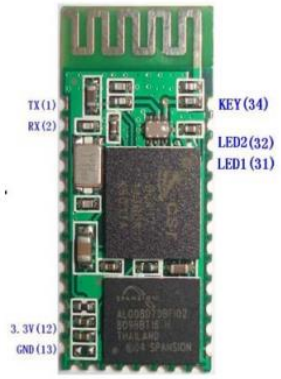


Figura 88.- Módulo Bluetooth HC-06

Las propiedades del módulo Bluetooth son las que se muestran a continuación [81]:

- El modo entre maestro y esclavo no puede ser cambiado.
- Nombre del dispositivo por defecto: linvor.
- Password por defecto: 1234
- En el modo maestro: almacena en su memoria el último esclavo con el que se emparejó.
- Emparejamiento o Vinculación: El maestro buscará y se emparejará con el esclavo automáticamente.
- Comunicación multi-dispositivo: solo se permite comunicación punto a punto para los módulos, pero el adaptador puede comunicarse con varios módulos.
- Modo AT: después de vincularse puede pasar al modo AT. Después de estar vinculado la comunicación es transparente.
- Durante el modo de comunicación, el módulo no podrá entrar en el modo AT.

- La tasa de transmisión por defecto es de 9600 baudios.
- KEY, PIN26 en el alto el maestro vacía la memoria de emparejamiento o vinculación.
- LED: la frecuencia de oscilación del dispositivo en modo esclavo es de 102ms. Si el dispositivo se encuentra en modo maestro, el parpadeo durante la vinculación serán de 110ms/s. Si en modo maestro no contiene nada en memoria la frecuencia de parpadeo será de 750ms. Cuando el dispositivo está conectado el led pasa a alto.
- El consumo durante la vinculación es de 30 a 40 mA, durante el resto del tiempo exista o no comunicación el consumo es de 8mA.

La distribución y descripción de los pines del módulo HC-06 se muestra en la Tabla 35.

PIN1	UART_TXD, niveles TTL/CMOS
PIN2	UART_RXD, niveles TTL/CMOS
PIN11	RESET, el pin de reset, con una entrada en bajo se reiniciara el módulo, cuando el módulo está en funcionamiento el pin puede estar al aire
PIN12	VCC, voltaje de alimentación. El voltaje nominal es 3.3V pero puede soportar entre 3.0-4.2V
PIN13	GND
PIN22	GND
PIN24	LED indicador del modo de funcionamiento
PIN26	Para el modo maestro, este PIN es usado para eliminar la información de vinculación o emparejamiento

Tabla 35.- Distribución de Pines del módulo HC-06 [81] [82]

La configuración serial por defecto del módulo es 9600 8 N 1 [81] [82], que significa que tenemos una velocidad de transmisión de 9600 baudios, 8 bits de datos, sin paridad y 1 bit de parada. El módulo puede ser configurando enviando comandos AT a través del puerto serial, algunos de los comandos AT que se pueden utilizar para configurar el módulo se muestran en la Tabla 36.

Comando	Acción
AT	Inicia la configuración por comandos AT
AT+BAUD1	Configura la velocidad de transmisión a 1200 baudios
AT+NAME*name*	Se cambia el nombre el módulo a *"name"*
AT+PIN*1234*	Se cambia la contraseña del módulo a "1234"
AT+PAPEL=M	Cambia al modo maestro
AT+PAPEL=S	Cambia al modo esclavo
AT+PN	Sin paridad
AT+PO	Paridad impar
AT+PE	Paridad par

Tabla 36.- Comandos AT para configuración del módulo HC-06 [81] [82]

El módulo puede soportar velocidades entre 1200 a 1382400 baudios, aunque es recomendado no superar una tasa de transmisión de 115200 baudios por el ruido generado, ya que se puede perder la comunicación con el mismo [82]. Los comandos AT para configurar el módulo a las distintas velocidades se muestran en la Tabla 37.

Comando	Velocidad(baudios)
AT+BAUD1	1200
AT+BAUD2	2400
AT+BAUD3	4800
AT+BAUD4	9600
AT+BAUD5	19200
AT+BAUD6	38400
AT+BAUD7	57600
AT+BAUD8	115200
AT+BAUD9	230400
AT+BAUDA	460800
AT+BAUDB	921600
AT+BAUDC	1382400

Tabla 37.- Comandos AT para la configuración de la tasa de transmisión del módulo HC-06 [82]

En el mercado actual (2014) se puede adquirir este módulo a un costo muy bajo y en la mayoría de los casos viene integrado en una placa que contiene los elementos necesarios para su correcto funcionamiento, incluyendo los leds indicadores del modo de funcionamiento, la compatibilidad con niveles de alimentación de 5V y un conector reducido que permite acceder a los pines principales del módulo.

Las configuraciones de módulos pueden variar dependiendo del proveedor, el módulo adquirido para desarrollo de esta tesis posee un conector de seis pines, entre ellos tenemos:

- *STATE:* indica el modo de funcionamiento
- *RXD:* pin para la recepción Serial
- *TXD:* pin para la Transmisión serial
- *VCC:* alimentación a 5V
- *GND:* conexión a tierra
- *KEY:* vaciar la memoria de vinculación cuando está en modo maestro

En la Figura 89 se puede apreciar una fotografía del dispositivo Bluetooth utilizado para este proyecto.

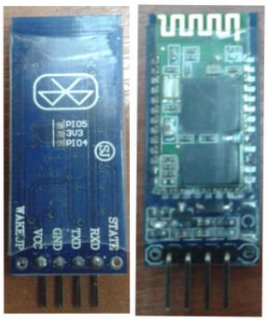


Figura 89.- Módulo HC-06 adquirido para el desarrollo de este proyecto

Para nuestro caso se usaron los primeros 5 pines, ya que solo se trabajará en modo esclavo, además se mantuvo la velocidad de transmisión a 9600 baudios, con 8 bits de información, sin paridad y un bit de parada (9600 8N1). Solamente se cambió el nombre del módulo a "ROBODROID" para hacer más fácil su identificación al momento de vincular el módulo con el teléfono inteligente. El pin *STATE* es de gran importancia ya que nos servirá para detectar cuando el módulo esté conectado o cuando este haya perdido la conexión con el maestro (teléfono inteligente).

4.2.2 Comunicación bluetooth en Android

Para el manejo del adaptador Bluetooth dentro del smartphone será necesario primero darle los permisos para acceder a dicho hardware (archivo *Android Manifest)*. Para llevar a cabo esta tarea, se deben incluir las instrucciones que se muestran en la Figura 90.

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

Figura 90.- Permisos de uso Bluetooth, declarados dentro del *Android Manifest*

Es importante también destacar que para manejar la comunicación Bluetooth se deben considerar tres elementos:

- *BluetootSocket:* socket o puerto de comunicación Bluetooth
- *BluetoothDevice:* dispositivo Bluetooth remoto o vinculado
- *BluetoothAdapter:* el adaptador Bluetooth del dispositivo que estamos usando

Una vez establecidos los permisos necesarios dentro del *Manifest* y tomando en consideración los conceptos anteriores, será necesario obtener el adaptador Bluetooth del Smartphone que se esté utilizando mediante el método ***getDefaultAdapter()***, de esta manera en la clase principal tendremos el fragmento de código que se ilustra Figura 91.

```
BluetoothAdapter mBluetoothAdapter=BluetoothAdapter.getDefaultAdapter();
```

Figura 91.- Obtención del adaptador Bluetooth del Smartphone

Una vez que se ha obtenido el adaptador Bluetooth será necesario verificar si el mismo se encuentra encendido, caso contrario se lanzará un nuevo Intento (*Intent*) a fin de encender el mismo; para este punto se puede utilizar el método ***startActivityForResult()***, que devolverá el resultado del intento previamente lanzado en el método ***onActivityResult()***. Este procedimiento nos permitirá determinar si el adaptador Bluetooth se encendió correctamente o si la acción fue cancelada por el usuario, ver Figura 92.

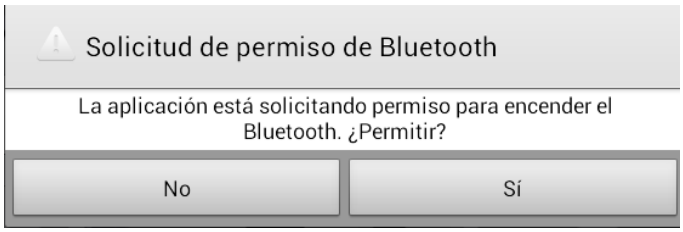


Figura 92.- Solicitud de encendido del Adaptador Bluetooth

Cuando se haya verificado que el adaptador Bluetooth está encendido, se puede obtener la información del mismo, como puede ser la lista de dispositivos vinculados así como la *mac-address* correspondiente, para hacer con ellos lo que se crea conveniente.

El siguiente punto será conectarse al dispositivo remoto o esclavo de nuestro interés, por lo cual se debe considerar que este tipo de acciones deben ser ejecutadas siempre en segundo plano, a fin de evitar que el hilo principal o la interfaz gráfica se congele o que la aplicación finalice inesperadamente. Por ello, es necesario el uso de una tarea asíncrona (*AsyncTask)* que lo llamaremos *"Conectar"*, al cual se le pasa el dispositivo bluetooth (*BluetoothDevice)* remoto al cual se desea conectar. En el método ***onProgresupdate()*** perteneciente al *AsyncTask* ser donde se proceda abrir el socket de comunicación (Figura 93).

```
BluetothSocket socket=dispositivo.createInsecureRfcommSocketToServiceRecord(
UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"));
```

Figura 93.- Creación del socket Bluetooth

Como se aprecia en la figura anterior, se debe pasar como parámetro el UUID[5] o identificador único de proceso, que es requerido para la creación del socket o puerto de comunicación. Una vez se crea el socket se puede conectar al mismo mediante el método ***connect()***; además hay que crear dos buffers tanto para escritura como para lectura de la información transmitida y recibida, respectivamente.

```
socket.connect();

InputStream ins=socket.getInputStream();

OutpuStream ons=socket.getOutputStream();
```

Figura 94.- Conexión del Socket Bluetooth y creación de canales de lectura y escritura

En este punto si la conexión ha sido exitosa será posible enviar los datos que se requieran por parte del dispositivo esclavo, aunque aún no será posible recibir los datos provenientes del dispositivo esclavo, ya que no se posee ningún método de escucha constante que notifique la llegada de información nueva. En tal virtud, en el método ***onPostExecute()*** de la tarea asíncrona de conexión "*Conectar*" se lanzará una segunda tarea asíncrona de recepción que estará ejecutándose siempre como método de escucha continua, a fin de capturar los datos transmitidos por el dispositivo Bluetooth remoto (plataforma). Es importante considerar que se deben tener presentes los ciclos de vida de la aplicación, puesto que se debe cerrar el socket de comunicación y detener las tareas en segundo plano cuando la aplicación es detenida o destruida.

Los diagramas de flujo de conexión y comunicación mediante tecnología Bluetooth se muestran en las Figuras 95, 96 y son válidos para cualquier aplicación basada en Android que re-quiera manejar la comunicación serial sobre tecnología Bluetooth.

Para las pruebas iniciales de conectividad, entre el dispositivo inteligente y el módulo bluetooth se utilizó la aplicación "Bluetooth Terminal" [83].

[5] UUID.- Universally unique indentifier

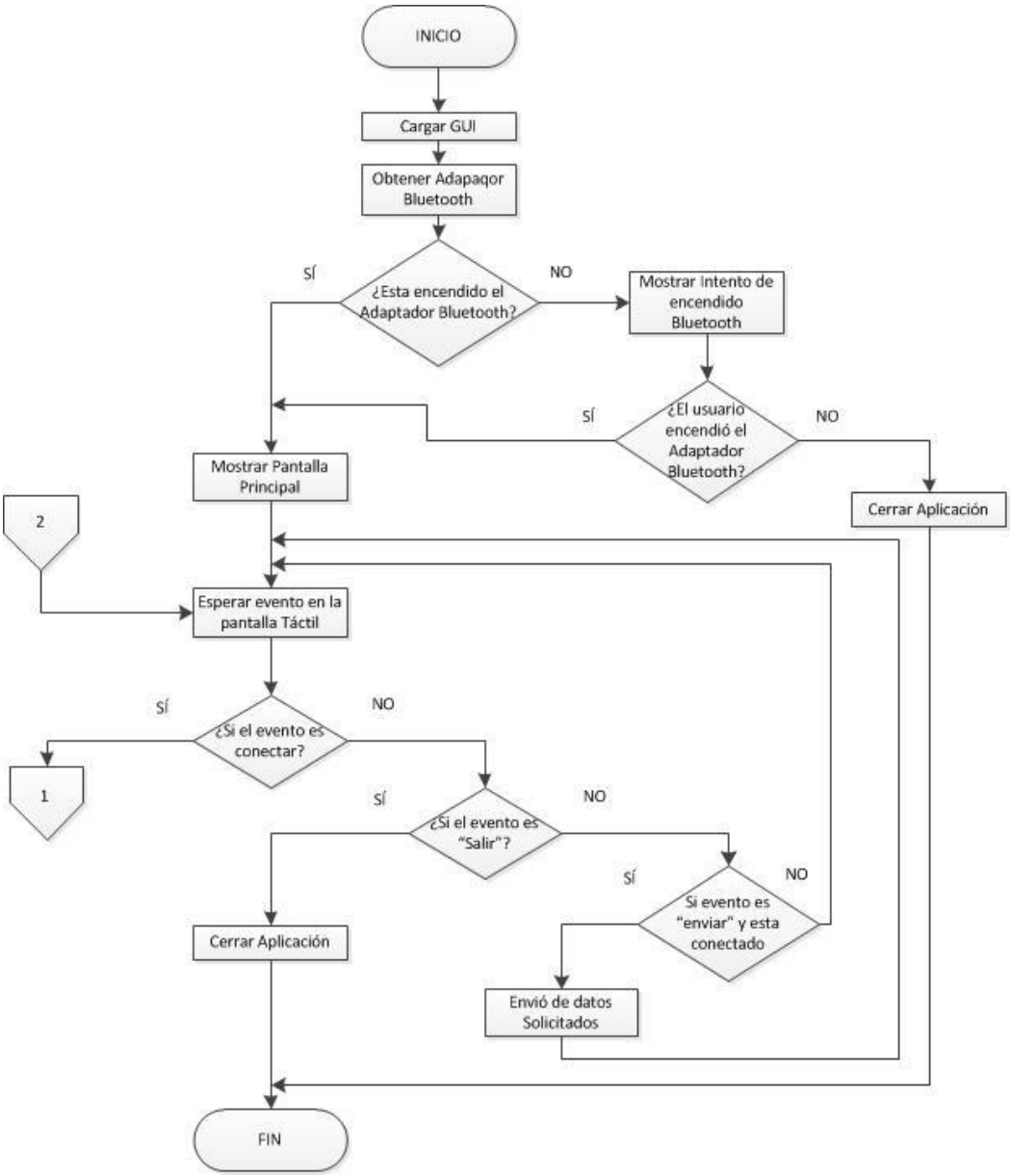


Figura 95.- Diagrama de Flujo de conexión y comunicación Bluetooth para dispositivos Android

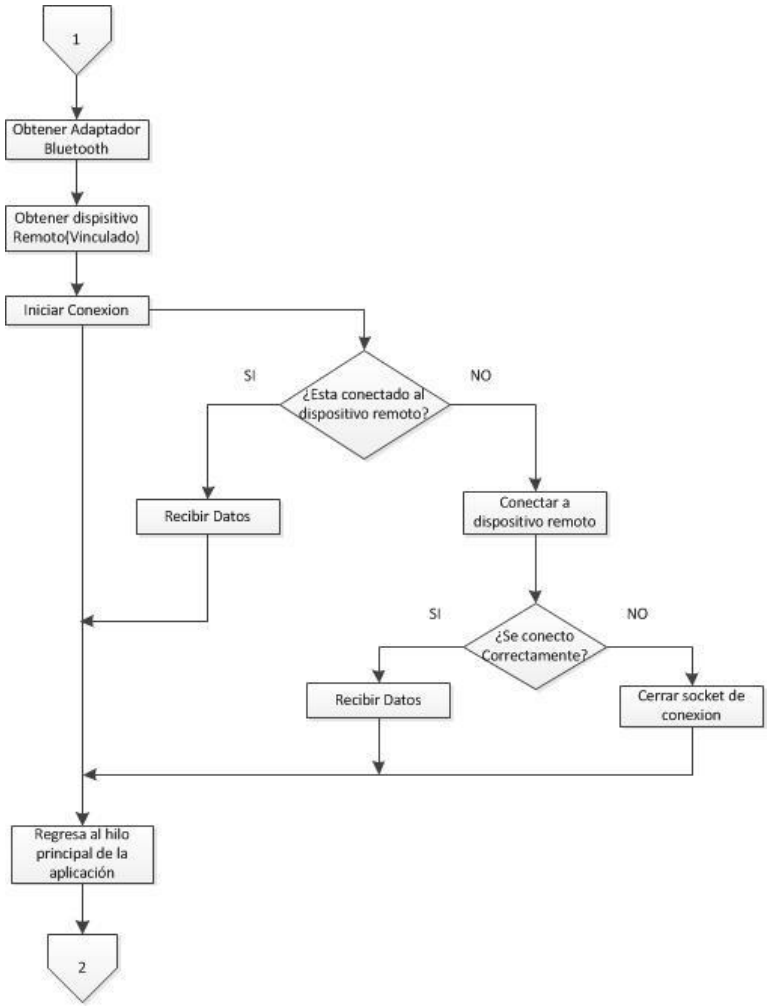


Figura 96.- Diagrama de Flujo de conexión y comunicación Bluetooth para dispositivos Android

Para el desarrollo de la tesis se crearon dos aplicaciones, basadas en los diagramas de flujo mostrados en las Figuras 95 y 96, ya que el único elemento que cambia es la interfaz gráfica. En la Figura 97 se aprecia una captura de pantalla de una primera aplicación Android desarrollada para verificar el correcto funcionamiento del protocolo de comunicación. La descripción de cada elemento numerado en la Figura 97 se detalla en la Tabla 38.

Figura 97.- Aplicación "TestProtocolo" desarrollada para las pruebas del protocolo de comunicación

Elemento	Tipo	Descripción
1	Spinner	Permite visualizar y seleccionar los dispositivos Bluetooth vinculados a los que se puede realizar un intento de conexión
2	Button	Este elemento permite conectarse o desconectarse del dispositivo Bluetooth remoto. El texto que muestra cambia entre "Conectar" o "Desconectar" indicando el estado de conexión en el que se encuentra.
3	EditText	Un total de 10 campos de texto numéricos que permiten ingresar los bytes que conformarán la trama de datos que se desea enviar al dispositivo Bluetooth remoto.
4	Spinner	Permite seleccionar el tamaño en bytes de la trama a enviar.
5	Button	Envía la trama configurada al dispositivo Bluetooth remoto.
6	Button	Cada vez que es presionado limpia los datos recibidos desde el dispositivo Bluetooth remoto y que son mostrados en el elemento número 7.
7	TextView	Muestra los datos que recibe desde el dispositivo Bluetooth remoto, así como también indica cuando se ha perdido la comunicación con el dispositivo Bluetooth remoto.

Tabla 38.- Descripción de los elementos de la aplicacion "TestProtocolo"

La segunda aplicación que se desarrolló en Android posee más elementos distribuidos a lo largo de cuatro pestañas, permitiendo un manejo mucho más ágil de todos los componentes de la plataforma. En las Figuras 98, 99, 100 y 101 se muestran las diferentes pestañas que conforman la segunda aplicación y en la Tabla 39 se encuentra la descripción detallada de cada componente de la aplicación.

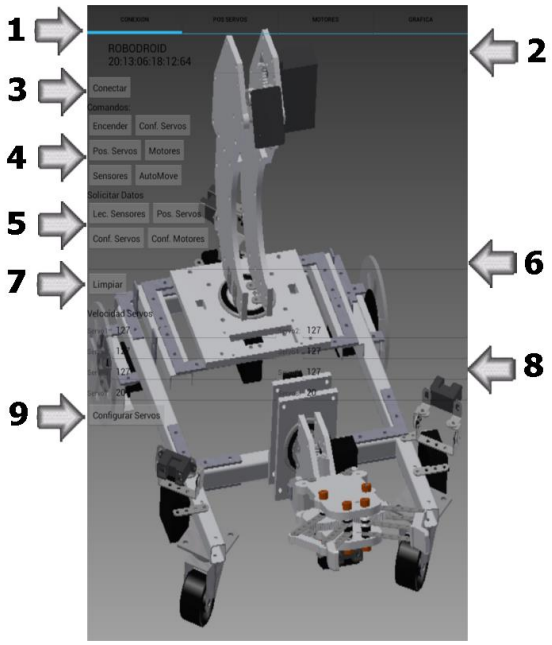


Figura 98.- Primera pestaña de la aplicación "Robodroid"

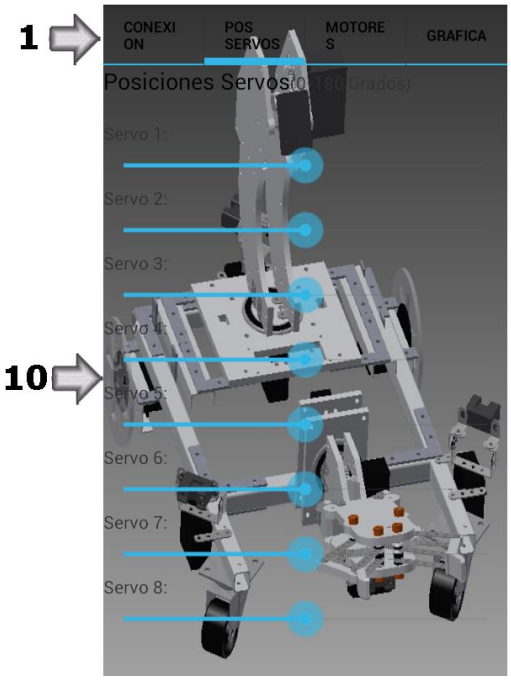


Figura 99.- Segunda pestaña de la aplicación "Robodroid"

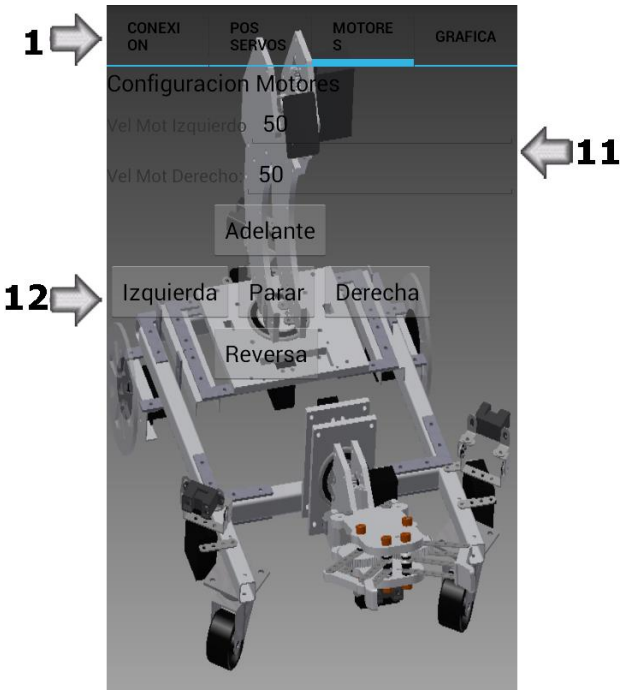


Figura 100.-Tercera pestaña de la aplicación "Robodroid"

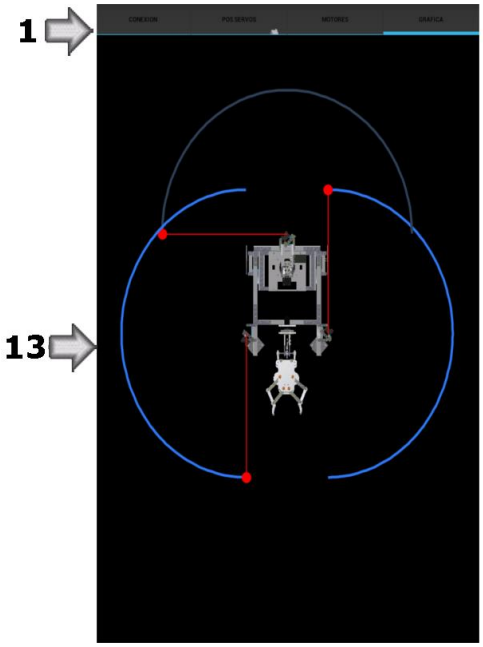


Figura 101.-Cuarta pestaña de la aplicación "Robodroid"

Elemento	Tipo	Descripción
1	TabWidget	Contenedor de pestañas: permite seleccionar entre las cuatro pestañas disponibles “CONEXION”, “POS-SERVOS”, “MOTORES” y “GRÁFICA”.
2	Spinner	Contenido dentro de la pestaña “CONEXIÓN”: permite visualizar y seleccionar los dispositivos Bluetooth vinculados a los que se puede realizar un intento de conexión.
3	Button	Contenido dentro de la pestaña “CONEXIÓN”: permite conectarse o desconectarse del dispositivo Bluetooth remoto. El texto que muestra cambia entre “Conectar” o “Desconectar” indicando el estado de conexión en el que se encuentra.
4	Button	Los seis botones contenidos dentro de la pestaña “CONEXIÓN”: permiten configurar la LCD que esta posee. Se puede establecer si la LCD está encendida o apagada, así como los datos que mostrará en la misma. El botón “Automove” activa la secuencia de movimiento de los servomotores que sostiene a los 3 sensores de distancia a fin de realizar un barrido alrededor del robot.
5	Button	Los cuatro botones contenidos dentro de la pestaña “CONEXIÓN” son utilizados para solicitar las lecturas desde la plataforma.
6	TextView	El editor de texto contenido en la ventana “CONEXIÓN” muestra los datos que recibe desde el dispositivo Bluetooth remoto, e indica cuando se ha perdido la comunicación con el mismo.
7	Button	Al presionar el visor de texto contenido en la pestaña “CONEXIÓN” limpia los datos recibidos desde el dispositivo Bluetooth remoto que son mostrados en el elemento número 6.
8	EditText	Los ocho campos de texto numéricos contenidos en la pestaña “CONEXIÓN” son utilizados para configurar la velocidad de los ocho servomotores de la plataforma, y admite valores de 0 a 127.
9	Button	El botón contenido en la pestaña “CONEXIÓN” envía las configuraciones de la velocidad de los servomotores a la plataforma.
10	Seekbar	Las ocho barras de búsqueda contenidas en la pestaña “POS_SERVOS” permiten seleccionar los ángulos de cada uno de los servomotores de la misma. Admite valores de 0 a 180.
11	EditText	Los dos campos de texto numéricos contenidos en la pestaña “MOTORES” permiten configurar la velocidad de los dos motores de la plataforma y por lo tanto la velocidad de desplazamiento de la misma. Admites valores de 0 a 100.
12	Button	Los cinco botones contenidos en la pestaña “MOTORES” permiten configurar la dirección de desplazamiento de la plataforma.
13	SurfaceView	Superficie de dibujo contenida en la ventana “GRÁFICA”

		realiza una gráfica de las lecturas de los sensores de distancia que posee la misma.

Tabla 39.- Descripción de los elementos de la aplicación "Robodroid"

Un aspecto importante a considerar en la segunda aplicación es el proceso que se efectúa a fin de graficar las mediciones que realizan los sensores de distancia; tal como se aprecia en la Figura 125, la curva del sensor no es lineal, por lo cual es necesario tomar muestras de la parte más importante de la curva. Esto se hace a fin de determinar una ecuación que la pueda representar y de esta manera tener el valor real en centímetros de la medición de distancia efectuada por los sensores.

Las muestras se toman solo de la sección de la curva que representa la información real de la distancia, de tal manera en la Tabla 40 tenemos los valores de las muestras de voltaje y distancia tomadas.

Voltaje(V)	Distancia(ctms)
2,76	15
2,55	20
2,25	25
2	30
1,75	35
1,54	40
1,39	45
1,24	50
1,15	55
1,04	60
0,97	65
0,9	70
0,86	75
0,83	80
0,76	85
0,71	90
0,68	95
0,65	100
0,61	105
0,59	110
0,57	115
0,53	120
0,51	125
0,49	130
0,47	135

0,45	140
0,43	145
0,42	150
0,2	200

Tabla 40.- Muestras tomadas del Sensor Infrarrojo de distancia

La gráfica de los valores que nos da la curva que se muestran en la Figura 102, en donde los valores del eje Y son la distancia en centímetros y los valores en el eje X representan el voltaje que devuelve el sensor.

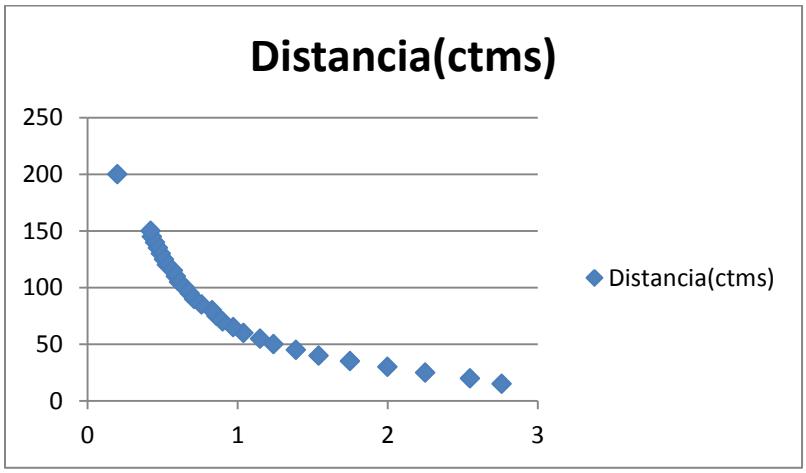


Figura 102.- Muestras de Distancia versus Voltaje del sensor de distancia

Con la ayuda de un software de computadora se puede estimar una línea de tendencia que represente de mejor manera y con un menor rango de error la curva del sensor. De esta manera se determinó que la mejor ecuación que representaba la curva es una polinómica de orden 3. La ecuación queda como se muestra a continuación:

$$y = -30{,}456x^3 + 157{,}59x^2 - 342{,}78x + 259.65$$

En donde **X** equivale al voltaje en voltios e **Y** equivale a la distancia en centímetros. En la Figura 103, se puede observar la curva del sensor conjuntamente con la ecuación que se determinó como la mejor aproximación.

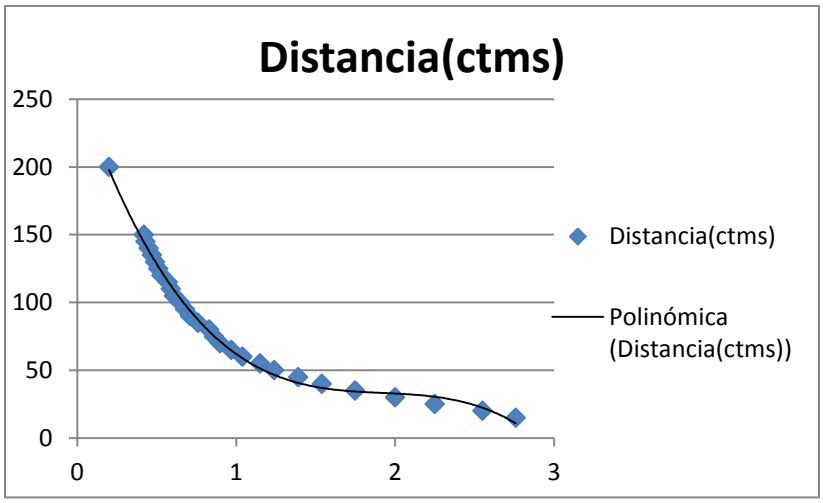


Figura 103.- Muestras y Ecuación de aproximación de la Curva del sensor de distancia

4.2.3 Protocolo de comunicación de la plataforma

El protocolo de comunicación fue escrito en C utilizando el programa CCS C, que es un compilador de C optimizado para micros controladores PIC. Debido a que en C para micro controladores no existe gestión de buffer y la comunicación debe ser lo suficientemente robusta, se diseñó un protocolo para manejar los datos a medida que son recibidos. De esta forma es posible liberar recursos lo más rápido, detectar errores y prevenir fallas que puedan ocasionar el mal funcionamiento de la plataforma. El protocolo fue diseñado de tal manera que sea fácil de usar en el caso de que otro desarrollador quiera hacer uso de la misma.

El micro controlador usado para el protocolo de comunicación de la plataforma fue el PIC16f877a, en la sección posterior se describe la distribución de pines, el esquema de conexión con los diferentes componentes y algunos detalles importantes sobre la configuración del mismo. La recepción de datos se hace por una interrupción en hardware del puerto serial del PIC antes mencionado, lo que permite optimizar dicha recepción, permitiendo que esta operación se realice en cualquier instante de ejecución del programa y no exista ninguna pausa.

Con el protocolo de comunicación diseñado se posee una tiene gran libertad para el control de cada elemento que conforma la plataforma que soportará los dispositivos inteligentes, dentro de las características más destacadas del protocolo de comunicación están:

- Es robusto, ya que es capaz de detectar información repetida o errores en la información que contiene en la trama.
- Se pueden configurar los parámetros que se muestran en la LCD de la plataforma.
- Permite configurar la velocidad individual de todos los servomotores que posee la plataforma.

- Permite configurar la posición individual en grados de los ocho servomotores que posee la plataforma.
- Permite configurar la velocidad así como la dirección de los dos servomotores de rotación continua que proveen el desplazamiento de la plataforma.
- Permite activar el barrido de los sensores de distancia, a fin de brindar soporte a operaciones de reconstrucción en dos o tres dimensiones del ambiente.
- Es posible solicitar las configuraciones de velocidad y posición de todos los servomotores de la plataforma
- Permite solicitar la dirección y velocidad de los dos servomotores de rotación continua que proveen el desplazamiento a la plataforma.
- Permite solicitar las lecturas de los sensores de distancia, así como la del sensor de fuerza.
- Permite activar la función especial "Automove" que realiza un barrido del entorno en donde se encuentre la plataforma con los tres sensores de distancia.

En la Figuras 104, 105 y 106 se muestra el diagrama de flujo del protocolo de comunicación que será programado en el micro controlador PIC16f877a.

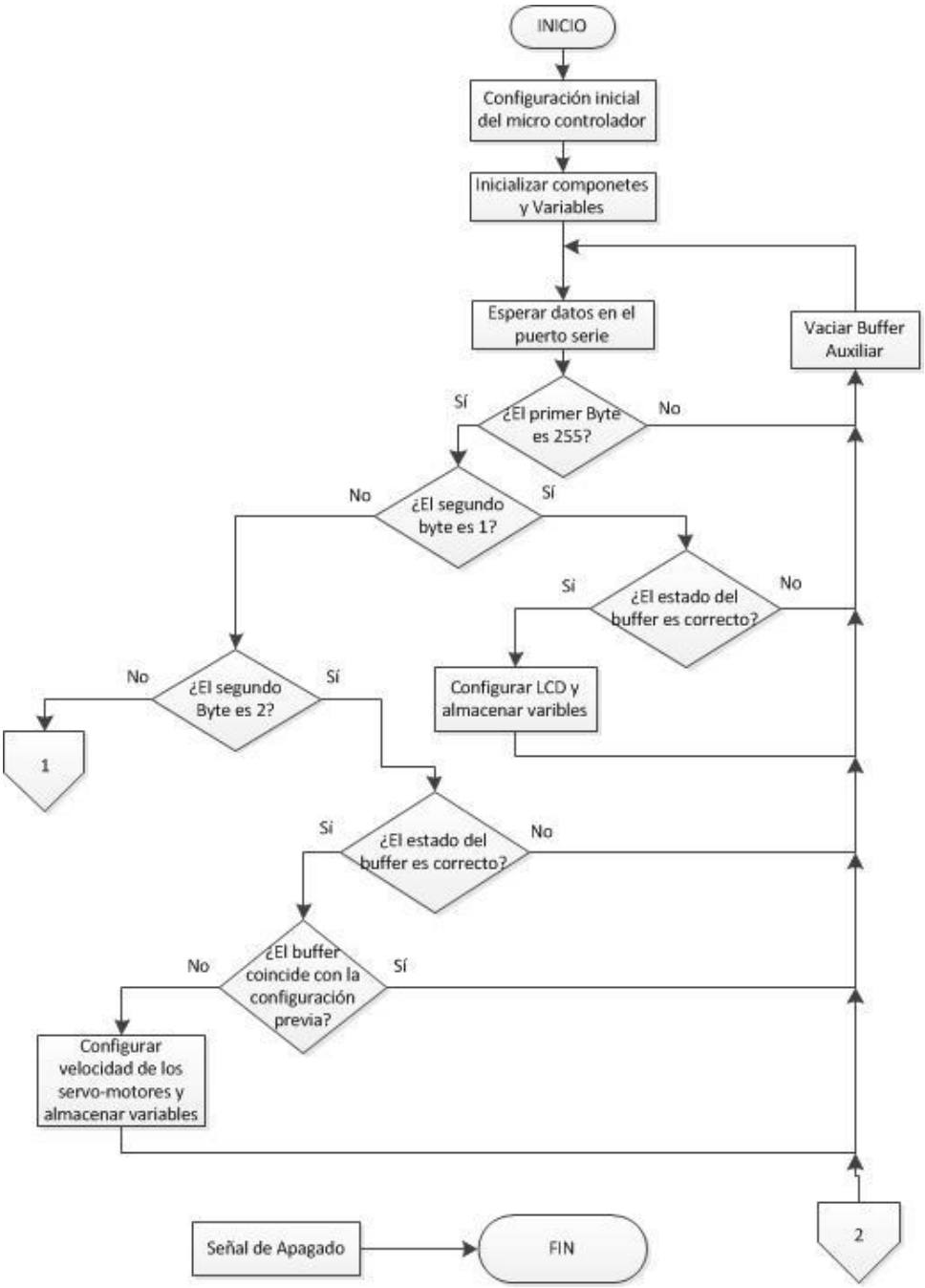


Figura 104.- Diagrama de flujo del Protocolo de Comunicación

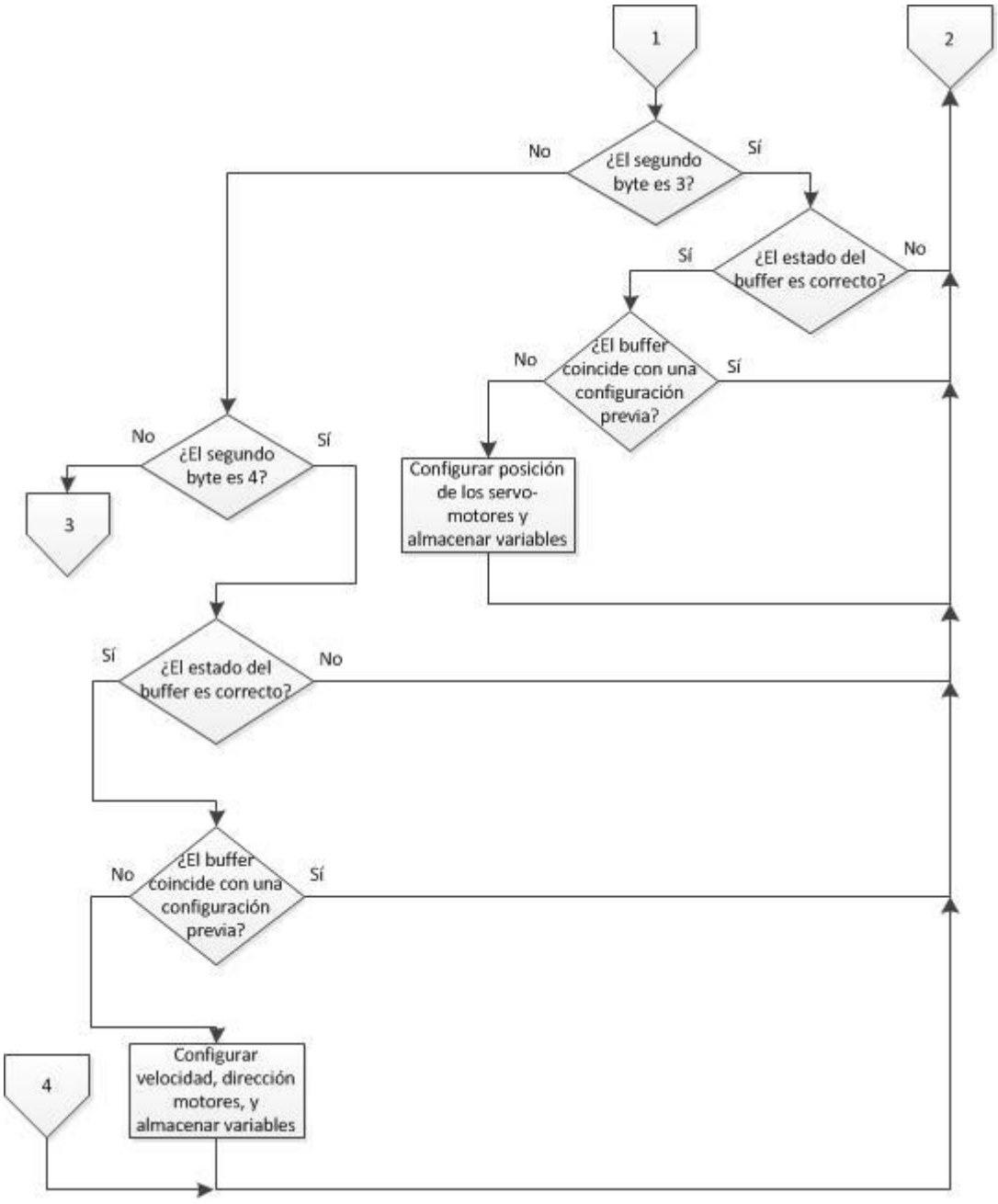


Figura 105.-Diagrama de flujo del Protocolo de Comunicación

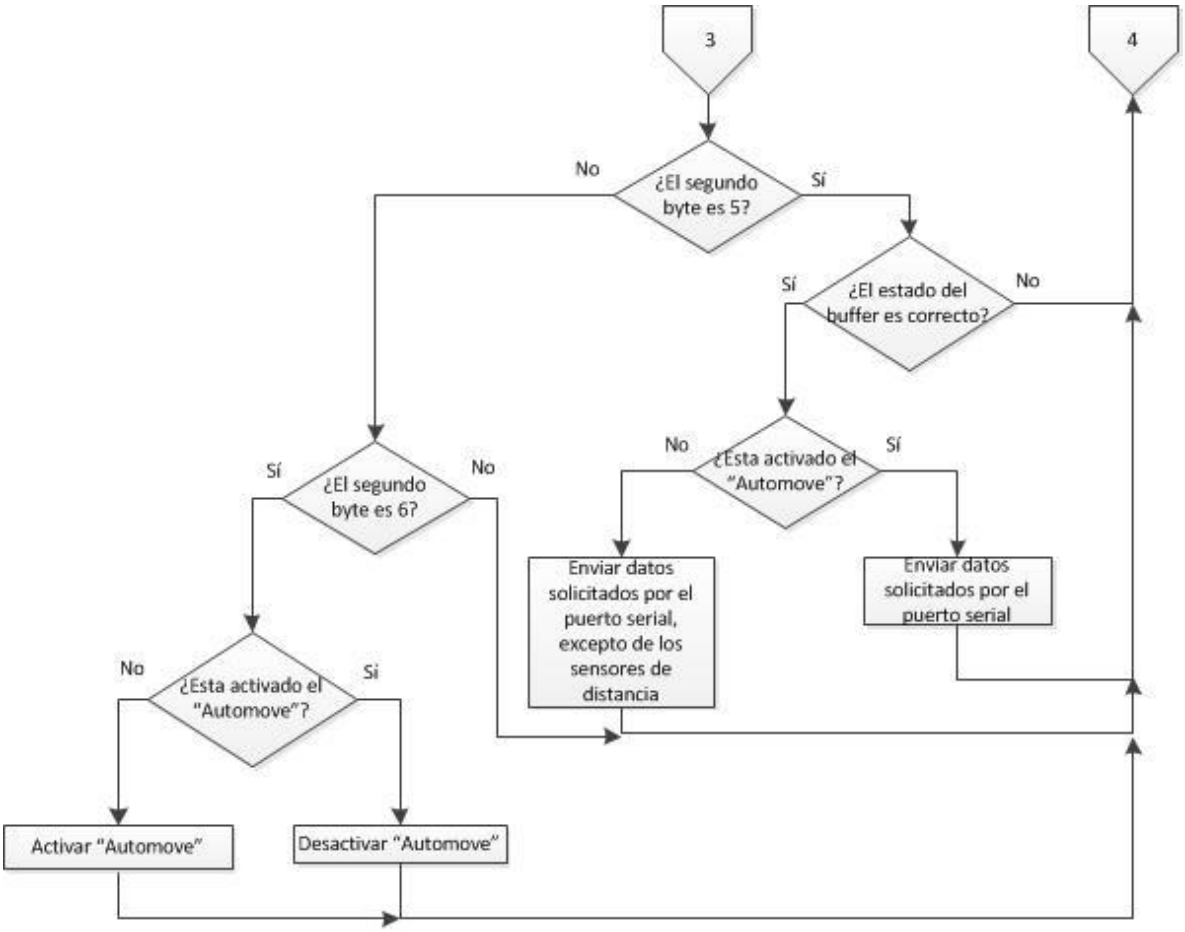


Figura 106.-Diagrama de flujo del Protocolo de Comunicación

El protocolo de comunicación diseñado maneja la trama en el formato que se muestra en la Figura 107, que consta de tres elementos:

- Un byte de sincronización.
- Un byte de comando.
- De cero a ocho bytes de información dependiendo del comando enviado.

Figura 107.- Formato de Trama admisible para el Protocolo de Comunicación

El primer byte de sincronización siempre debe ser 255, y el byte de comando puede tomar valores entre 1 a 6. El tamaño del paquete de información dependerá del comando. En la Tabla 41 se puede apreciar el tamaño del paquete que se envía con cada comando.

Comando	Tamaño de "Data"	Descripción
1	2 Bytes	Configuración de la LCD
2	8 Bytes	Configura velocidad servomotores
3	8 Bytes	Configura posición servomotores
4	4 Bytes	Configura velocidad y dirección Motores
5	1 Byte	Solicita lecturas y configuraciones
6	0 Bytes	Activa "Automove"

Tabla 41.- Comandos del Protocolo de Comunicación

Comando 1: el comando 1 configura los parámetros de la LCD, como encender o apagar el *BackLight* de la LCD y los parámetros que se muestra en la misma. El formato de trama para este comando se muestra en la Figura 108.

Figura 108.- Trama para el comando 1

En donde el byte E puede tomar valores de 1 o 2, dando los siguientes resultados:

- El byte E igual a 1 enciende el *BackLight* de la LCD.
- El byte E igual a 2 apaga el *BackLight* de la LCD.

El byte M puede tomar valores entre 1 a 4, dando los siguientes resultados:

- El byte M igual a 1 muestra los valores de velocidad de cada uno de los servomotores en la LCD.
- El byte M igual a 2 muestra la posición de cada uno de los servomotores en la LCD.
- El byte M igual a 3 muestra la velocidad y dirección de los motores en la LCD.
- El byte M igual a 4 muestra las lecturas adquiridas por los diferentes sensores.

Comando 2: el comando 2 configura la velocidad de los ocho servomotores que posee la plataforma. El formato de trama para este comando se muestra en la Figura 109.

Figura 109.- Trama para el comando 2

Los bytes de C1 a C8 configuran la velocidad de cada uno de los ocho servomotores de la plataforma, admitiendo valores comprendidos entre 0 a 127.

Comando 3: el comando 3 configura las posiciones de cada uno de los servomotores de la plataforma, el formato de trama para este comando se muestra en la Figura 110.

Figura 110.- Trama para el comando 3

Los bytes de P1 a P8 configuran la posición de cada uno de los servomotores que la plataforma posee, admiten valores entre 0 a 180, equivalente al ángulo en grados que tomarán los servomotores.

Comando 4: el comando 4 configura el la velocidad y dirección de los motores de la plataforma. El formato de la trama para este comando se muestra en la Figura 111.

Figura 111.- Trama para el comando 4

Los Bytes VI y VD, configuran la velocidad del motor izquierdo y derecho respectivamente, así como pueden tomar valores entre 0 a 100.

El byte DI puede tomar valores entre 0 a 2, dando como resultado:

- Si el byte DI es igual a 0 el motor izquierdo se detendrá, o se mantendrá detenido.
- Si el byte DI es igual a 1 el motor izquierdo girará en el sentido de las manecillas de reloj con la velocidad indicada en el byte VI.
- Si el byte DI es igual a 2 el motor izquierdo girará en contra de las manecillas del reloj con la velocidad indicada en el byte VI.

El byte DD puede tomar valores entre 0 a 2, dando como resultado:

- Si el byte DD es igual a 0 el motor derecho se detendrá, o se mantendrá detenido.
- Si el byte DD es igual a 1 el motor derecho girará en el sentido de las manecillas de reloj con la Velocidad indicada en el byte VD.
- Si el byte DD es igual a 2 el motor derecho girará en contra de las manecillas del reloj con la velocidad indicada en el byte VD.

Comando 5: el comando 5 solicita las lecturas de sensores, configuraciones y posiciones de cada uno de los servomotores, así como velocidad y dirección de los motores. El formato de la trama para este comando se muestra en la Figura 112.

Figura 112.- Trama para el comando 5

El byte D puede tomar valores entre 1 a 4, dando los siguientes resultados.

- Si el byte D es igual a 1, la plataforma devuelve la lectura de los sensores
- Si el byte D es igual a 2, la plataforma devuelve las configuraciones de velocidad actual de cada uno de los servomotores.
- Si el byte D es igual a 3, la plataforma devuelve la posición actual de cada uno de los servomotores.
- Si el byte D es igual a 4, la plataforma devuelve la dirección y velocidad del servo motor de rotación continua izquierdo y derecho.

Cada vez que se solicitan las distintas lecturas con el comando 5, la plataforma devolverá la información solicitada en un determinado formato:

1. En el caso de solicitar las lecturas de sensores cuando el *"Automove"* está desactivado la plataforma devolverá una trama con el formato que se muestra en la Figura 113. En donde a los bytes I, E y F se les asigna los valores "s", "/" y "#" del código ASCII, respectivamente. De S1 a S4 serán los valores (0 a 255) de lecturas de los sensores de distancia y S5 será el valor (0-255) de lectura del sensor de fuerza.

Figura 113.- Trama de vuelta cuando se solicita lectura de sensores

2. En el caso de que se solicite las lecturas de los sensores cuando *"Automove"* está activo la plataforma devuelve una trama con el formato que se muestra en la Figura 114. En donde los bytes I, E y F se les asigna los valores "b", "/" y "#" del código ASCII respectivamente. En este caso S4 será el valor (0-255) de la lectura del sensor de distancia ubicado en la pinza y S5 será el valor (0-255) de la lectura del sensor de fuerza.

Figura 114.-Trama de vuelta cuando se solicita lectura de sensores

3. Cuando el *"Automove"* está activo, cada 100 ms. la plataforma devolverá automáticamente los datos solicitados en el formato de trama que se muestra en la Figura 115. En donde los bytes I, E y F se les asigna los valores "a", "/" y "#" del código ASCII respectivamente. De S1 a S3 serán los valores (0-255) de lecturas de los tres sensores de distancia que hacen el barrido durante el *"Automove"*.

Figura 115.-Trama de vuelta cuando se solicita lectura de sensores

4. Al solicitar los valores de configuración de velocidad de los servomotores, la plataforma devolverá los datos solicitados en el formato de trama que se muestra en la Figura 116. En donde I, E, y F se les asigna los valores "c", "/" y "#" del codigo ASCII respectivamente. De C1 a C2 seran los valores (0-127) de configuración de velocidad de cada uno de los ocho servomotores de la plataforma.

Figura 116.-Trama de vuelta cuando se solicita las configuraciones de velocidad de los Servomotores

5. Al solicitar las posicones actuales de los servomotores, la plataforma devolvera los datos solicitados en el formato de trama que se muestra en la Figura 117 . En donde los bytes I, E y F se les asignara los valores "p", "/" y "#" del código ASCII respectivamente. De P1 a P8 son los valores (0-180) en grados de las posiciones actuales de cada uno de los ocho servomotores de la plataforma.

Figura 117.- Trama de vuelta cuando se solicita las posiciones de los servomotores

6. En el caso de solicitar las lecturas de configuraciones de velocidad y dirección de los servomotores de rotación continua, la plataforma devolverá los datos solicitados en el formato de trama que se muestra en la Figura 118. En donde los bytes I, E y F se les asignara los valores "m", "/" y "#" del código ASCII respectivamente. Para este caso DI es la dirección del motor Izquierdo, VI la velocidad del motor Izquierdo, DD la dirección del motor derecho y VD la velocidad del motor derecho. Para el caso de los bytes DI y DD tenemos que:

- Si es igual a 0 significa que el motor está detenido.
- Si es igual a 1 significa que está girando en el sentido de las manecillas del reloj
- Si es igual a 2 significa que girando en contra de las manecillas del reloj.

Figura 118.- Trama de vuelta cuando se solicita las lecturas de velocidad y dirección de los servomotores de rotación continúa

Comando 6: el comando 6 activa o desactiva, según sea el caso, la función de "Automove" o de barrido que realizan los sensores de distancia. El formato trama para este comando se muestra en la Figura 119.

Figura 119.- Trama para el comando 6

Cabe considerar que cuando el "Automove" está activado no se podrá modificar la posición de los tres últimos servomotores haciendo uso del comando 3.

4.3 Diseño e implementación del sistema de control y movimiento

4.3.1 Descripción de los componentes electrónicos que requiere la plataforma

A fin de cumplir los requisitos funcionales, la plataforma de transporte necesita varios componentes electrónicos y electromecánicos fundamentales, y de todos ellos en esta sección se describirán los más importantes. La plataforma posee:

- 7 Servomotores HT3001
- 1 Servomotor HS-755HB
- 2 Servomotores de rotación continua AR-3606HB
- 2 Pololu Micro Serial Servo Controller
- 4 Sensores GP2Y0A21YK
- Un sensor FlexiForce A201
- Módulo Bluetooth HC-06

El módulo Bluetooth no se describe en esta sección, en virtud de que ya fue descrito en la sección previa que trata sobre el protocolo de comunicación.

Características Servomotor HT3001: es un servomotor de 0 a 180 grados de rotación, de bajo costo, propósito general, que posee dos rodamientos que ayudan a reducir la fricción y mejorar el rendimiento [83].

Figura 120.- Servomotor HT3001 [83]

Las características de este servo se describen en la Tabla 42.

Dimensiones	40.7 x 20.5 x39.5 mm
Peso	43 g
Velocidad a 6 V	$0.12seg/60^0$
Torque a 6V	4.4 Kg.cm
Velocidad a 4.8 V	$0.15seg/60^0$
Torque a 4.8V	3.5 Kg.cm

Tabla 42.- Características Servomotor HT3001 [83]

Características Servomotor HS-755HB: es un servomotor de 0 a 180 grados de rotación, se puede observar en la Figura 121.

Figura 121.- Servomotor HS-755HB [84]

Las características del servomotor se detallan en la Tabla 43.

Dimensiones	58.9 x 29.0 x49.8 mm
Peso	110 g
Velocidad a 6 V	0.28 seg/60^0
Torque a 6V	13.2 Kg.cm
Velocidad a 4.8 V	0.23 seg/60^0
Torque a 4.8V	11.0 Kg.cm

Tabla 43.- Características Servomotor HS-755HB [84]

Características Servomotores AR-3606HB: es un servomotor estándar, diseñado especialmente para rotación continua, posee dos rodamientos para reducir la fricción, así un potenciómetro interno que permite ajustar el punto de referencia de la velocidad [85].

La velocidad máxima que puede alcanzar sin carga a 6V es de 71 RPM, y la velocidad máxima sin carga a 4.8V es de 62 RPM [85]

Figura 122.- Servomotor AR-3606HB [85]

Las características detalladas de este servo se presentan en la Tabla 44.

Dimensiones	40.5 x 20.0 x38.0 mm
Peso	40 g
Velocidad a 6 V	0.14 seg/60^0
Torque a 6V	93 Oz.in
Velocidad a 4.8 V	0.16 seg/60^0
Torque a 4.8V	83 Oz.in

Tabla 44.- Caracteristicas Servomotor AR-3606HB [85]

Características Pololu Micro Serial Servo Controller: es un controlador que permite comandar ocho servomotores simultáneamente a través de un puerto serial.

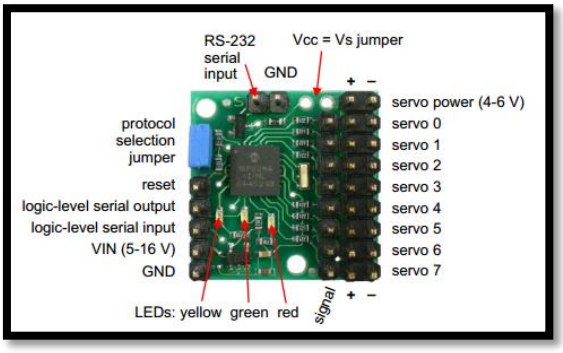


Figura 123.-Pololu Micro Serial Servo Controller [86]

Las características más importantes de controlador se detallan en la Tabla 45.

Número de servomotores	8
Rango de Ancho de Pulso	0.25-2.75 ms
Resolución	0.5 microsegundos
Voltaje de alimentación	5-16V
Voltaje servomotores	4.8-6.0V
Tasa de transmisión	Auto detección: 1200-38400 Baudios
Consumo de corriente	5 mA

Tabla 45.- Características Pololu Micro Serial Servo Controller [86]

Para la entrada serial se debe enviar 8 bits, sin paridad y un bit de parada (8N1), en niveles lógicos o RS-232 [86].

El LED verde indica que existe actividad serial, el LED naranja indica precaución y se enciende cuando se ha enviado un valor erróneo para la posición de un servomotor, finalmente el LED rojo se enciende cuando ha ocurrido un error fatal, con lo cual el controlador se desactivará hasta que sea reiniciado [86].

El controlador puede funcionar en dos modos [86]:

- Pololu Mode: el modo por defecto cuando el puente está abierto, este modo permite controlar múltiples controladores y acceder a las características especiales que posee este controlador.
- Mini SSC II Mode: es el modo más simple con el puente cerrado, permite controlar de manera simple la posición de cada servomotor.

Para las especificaciones más detalladas de los dos modos de funcionamiento se puede revisar el *DataSheet* del mismo adjunto en el Anexo o accediendo a la página web www.pololu.com.

Características sensor de distancia GP2Y0A21YK: sensor de distancia infrarrojo, se puede observar en la Figura 124.

Figura 124.- Sensor GP2Y0A21YK [87]

Las características del sensor se detallan en la Tabla 46.

Voltaje de alimentación	-0.3 a +7 V
Voltaje ideal de alimentación	4.5 a 5.5V
Voltaje de salida	-0.3 a Vcc + 0.3V
Temperatura de operación	-16 a 60 ^{0}C
Rango de distancia	20 a 150 cm
Voltaje de salida según distancia	1.8 a 2.3 V

Tabla 46.- Características Sensor GP2Y0A21YK [88]

La gráfica de la curva de la distancia frente al voltaje de salida para este sensor se muestra en la Figura 125.

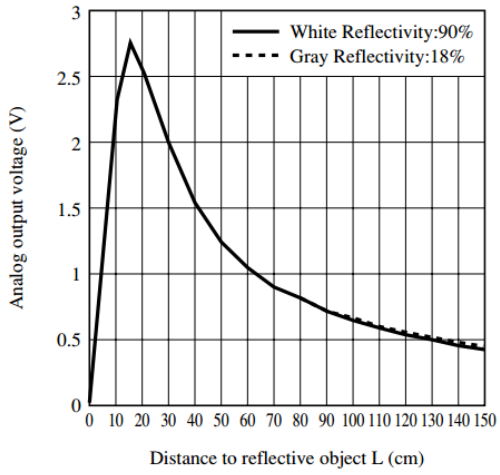


Figura 125.- Curva de Voltaje versus Distancia del sensor GP2Y0A21YK [88]

Características sensor FlexiForce A201: sensor de fuerza flexible, se muestra en la Figura 126.

Figura 126.- Sensor de Fuerza FlexiForce

Las características más importantes de este sensor se detallan en la Tabla 47.

Espesor	0.127 mm
Largo	203 mm
Ancho	14 mm
Zona Activa de sensado	9.53 mmm de diámetro
Rango de Fuerza	0 a 100 Lbs

Tabla 47.- Características Sensor FlexiForce [89]

La curva de voltaje de salida frente a la fuerza aplicada de este sensor se muestra en la Figura 127.

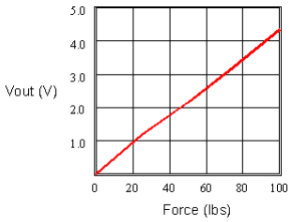


Figura 127.- Recta Voltaje versus Fuerza del sensor FlexiForce [89]

4.3.2 Diseño e implementación de la placa de control

El diseño de la placa fue realizado totalmente en Altium, una vez que se comprobó el su funcionamiento a través de simulación, se procedió a la fabricación de la placa final que sería montada sobre la plataforma y se volvería el sistema de control de la misma.

El micro controlador usado fue el PIC16F877A por su capacidad en memoria, su bajo costo y la facilidad para adquirirlo en nuestro mercado actual (2014).

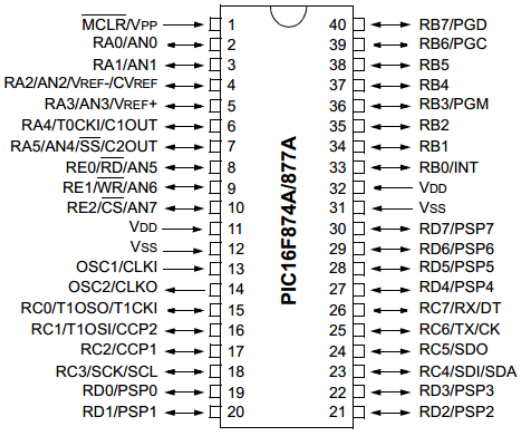


Figura 128.- Pines del micro controlador PIC16F877A [90]

La distribución de puertos y pines del micro controlador es:

- Puerto D conectado a LCD de 16 segmentos
- Puerto A0 utilizado como entrada analógica para un sensor infrarrojo
- Puerto A1 utilizado como entrada analógica para un sensor infrarrojo
- Puerto A2 utilizado como entrada analógica para un sensor infrarrojo
- Puerto A3 utilizado como entrada analógica para un sensor infrarrojo
- Puerto A5 utilizado como entrada analógica para el sensor de Fuerza
- Puerto C6 utilizado como transmisor serial conectado al pin RX del módulo Bluetooth
- Puerto C7 utilizado como receptor serial, conectado al pin TX módulo Bluetooth.
- Puerto B0 conectado al pin *State* del módulo Bluetooth.
- Puerto B2 encender o apagar el BackLight de la LCD.
- Puerto B3 usado para activar el primer controlador "Pololu Micro serial Servo Controller" encargado de controlar los 8 servomotores.
- Puerto B4 usado para activar el segundo controlador "Pololu Micro serial Servo Controller" en cargado de controlar los 2 servomotores de rotación continua.
- Pin C0 usado como salida serial para comunicarse con el primer controlador "Pololu Micro serial Servo Controller" en cargado de controlar los 8 servomotores.
- Pin C1 usado como salida serial para comunicarse con el segundo controlador "Pololu Micro serial Servo Controller" en cargado de controlar los 2 servomotores de rotación continua.

Durante el diseño se implementaron dos fuentes de alimentación totalmente separadas, para manejar tanto la parte de control como la parte de potencia independientemente, evitando así que el ruido de los servomotores se induzca en la etapa de control y pueda afectar su correcto funcionamiento.

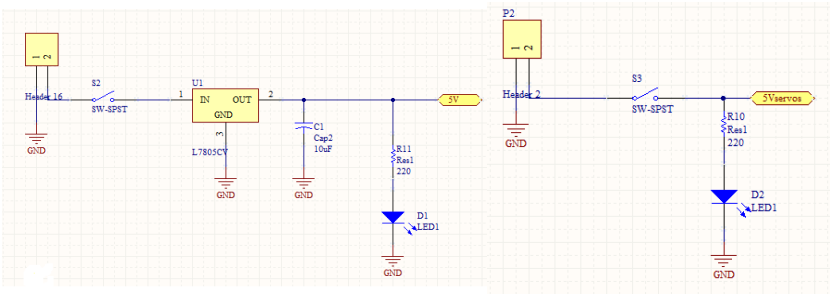


Figura 129.- Esquemas eléctricos para la etapa de Control y Potencia

El módulo Bluetooth va conectado a un socket tipo hembra, P6, el mismo que está conectado a la alimentación del circuito de control y a los pines correspondientes del micro controlador.

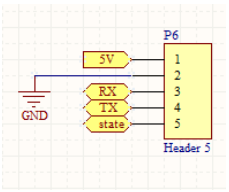


Figura 130.- Socket para el módulo Bluetooth

El primer controlador "Pololu micro serial servo Controller" va acoplado sobre la placa mediante nueve sockets tipo hembra P12 a P20, mientras que el P12 se conecta a la alimentación de control y a los pines respectivos del micro controlador. Ocho de los nueve sockets tipo hembra de P13 a P20 van conectados a ocho sockets tipo macho P21 a P28, permitiendo conectar fácilmente cada uno de los servomotores que se integraran a la plataforma. La alimentación de P21 a P28, o la de los servomotores está conectada a la etapa de potencia.

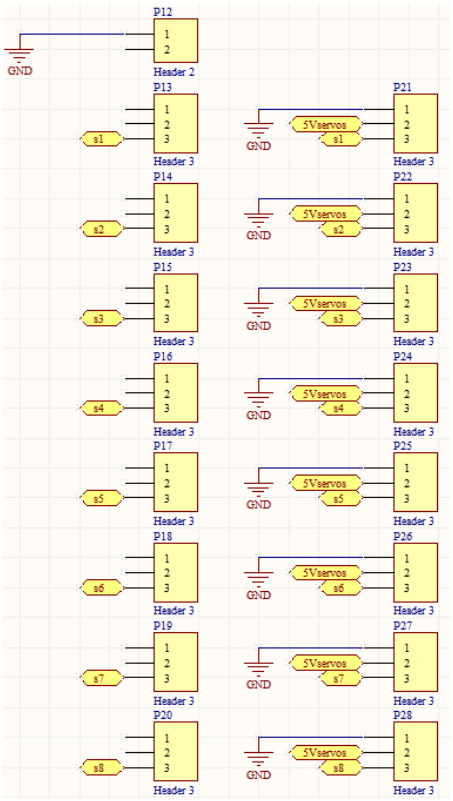


Figura 131.- Sockets para el primer Pololu Micro Serial Servo Controller

El segundo controlador "Pololu Micro serial Servo Controller" es el encargado de comandar la velocidad y dirección de dos servomotores de rotación continua. El controlador va acoplado sobre la placa mediante nueve sockets tipo hembra de P29 a P37, en donde P29 va conectado a la alimentación de control y a los pines correspondientes del micro controlador. El socket P30 y P31van conectados con dos sockets tipo macho P38 y P39 respectivamente, lo que permite acoplar fácilmente los servomotores de rotación continua. La alimentación de P38 y P39 se conecta a la alimentación de potencia.

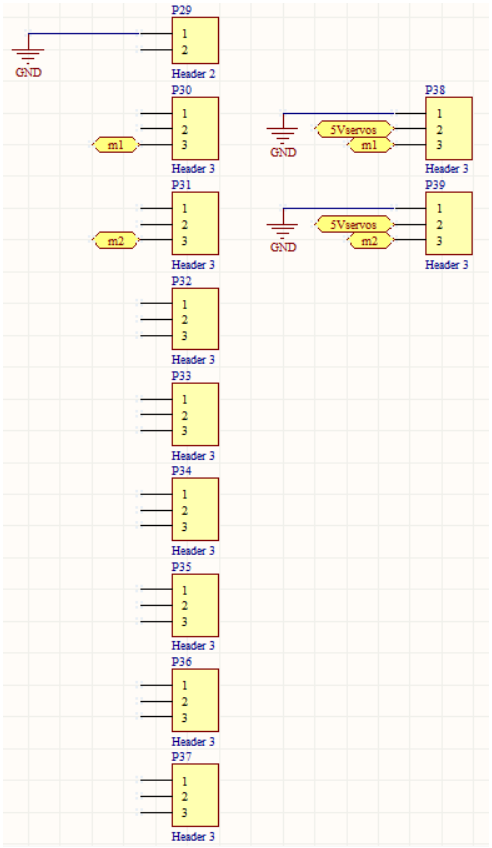


Figura 132.- Sockets para el segundo Pololu Micro Serial Servo Controller

Los cuatro sensores infrarrojos de distancia que posee la plataforma, van acoplados a cinco sockets tipo macho de P7 a P9, los mismos que están conectados a la etapa control, así como a los puertos analógicos correspondientes del micro controlador.

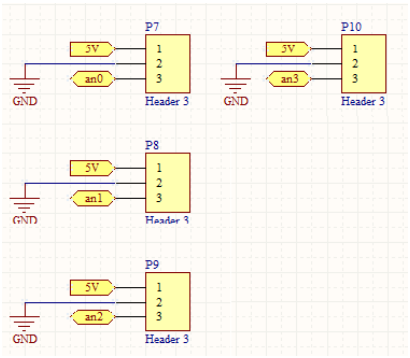


Figura 133.- Sockets para la conexión de los 5 Sensores Infrarrojos

Para el caso del sensor de fuerza es necesaria una etapa de pre amplificación antes que la señal ingrese al puerto analógico del micro controlador correspondiente. Para ello se diseñó el circuito que se muestra en la Figura 134, que consta de 3 Amplificadores operacionales LM324N, dos de ellos en configuración de seguidor de tensión, y uno de ellos como amplificador no inversor. Los seguidores de tensión son usados para evitar el problema del acoplamiento de impedancias y que este pueda afectar al valor real de lectura del sensor. El potenciómetro R9 permite regular la ganancia de la etapa de amplificación. El socket tipo macho P11 permite la conexión del sensor de fuerza.

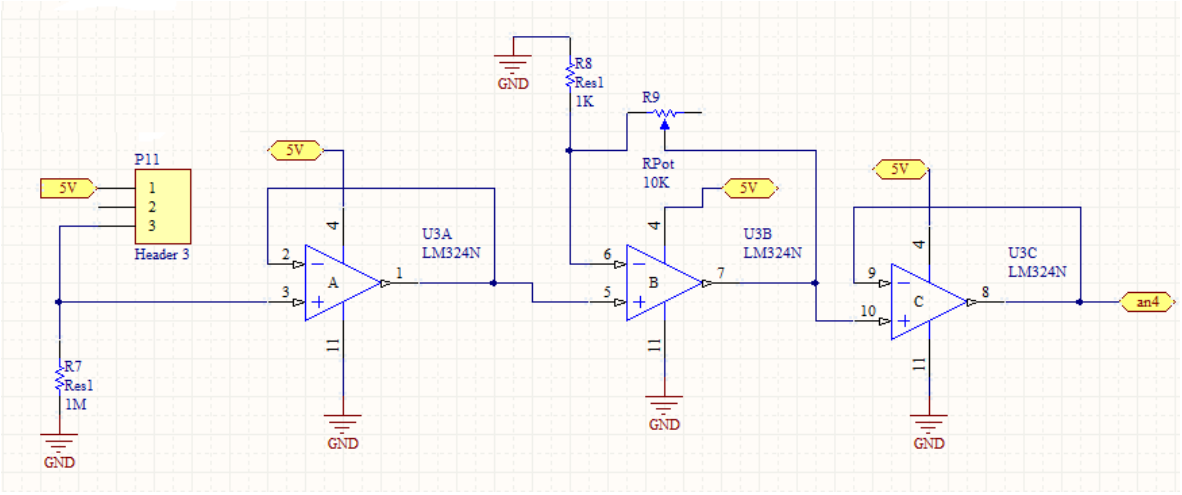


Figura 134.- Esquema eléctrico de la etapa de amplificación del Sensor de Fuerza

La ecuación de la etapa de amplificación estará dada como se muestra en la siguiente ecuación:

$$V_{out} = \left(\frac{R7}{R_S + R7} + \frac{R7.R9}{R_S.R8 + R7.R8}\right) V_{in}$$

En donde V_{out} es el voltaje de salida, V_{in} el voltaje de ingreso y R_S la resistencia del sensor de fuerza.

En la Figura 135 se muestra el esquema de conexión de la LCD de 16 segmentos, se utiliza un socket hembra P3, para facilitar la conexión de la misma cuando se ensamble a la placa final. El potenciómetro R1 permite regular el contraste de la LCD

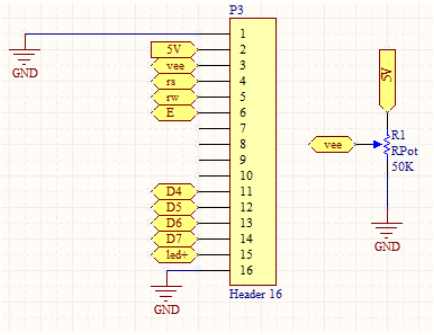


Figura 135.- Socket de conexión para la LCD de 16 Segmentos

Finalmente en la Figura 136 se puede observar el esquema de conexión de todos los componentes con el PIC16F877A. El cristal utilizado para el micro controlador es de 20Mhz.

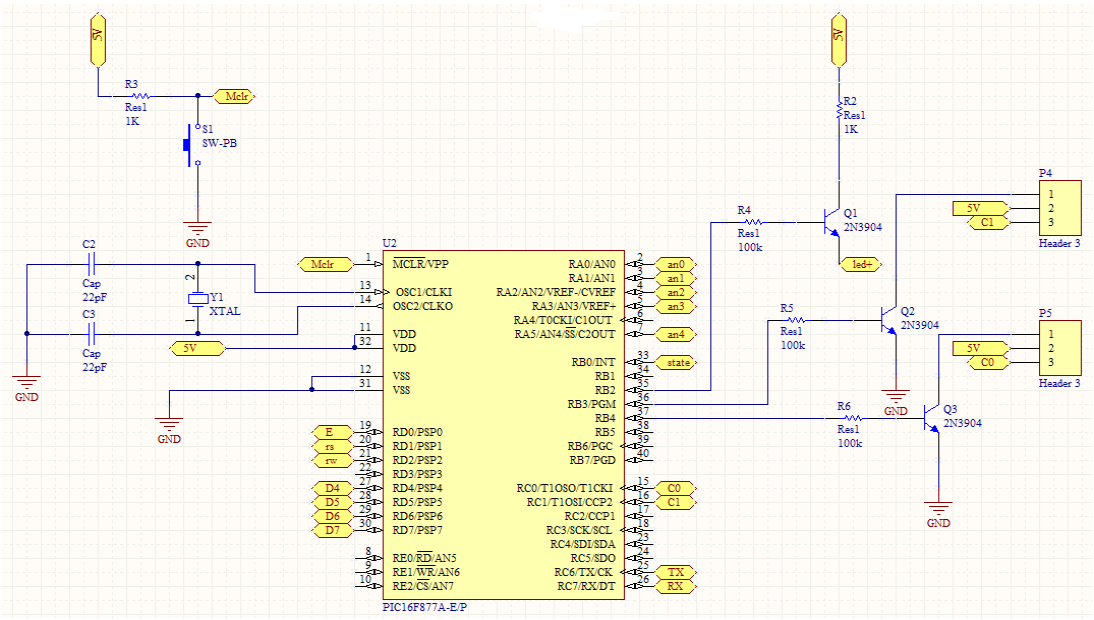


Figura 136.- Esquema de conexión del PIC16F877A

Una vez concluido y verificado mediante la simulación correspondiente, se procedió al diseño de la placa para su impresión. Se organizaron los componentes de tal manera que los elementos tengan un orden lógico y se optimice el espacio; además que una vez terminada que esta sea fácil de conectar y ensamblar en la estructura mecánica de la plataforma. En la Figura 137 se puede observar el ruteado de la placa en Altium.

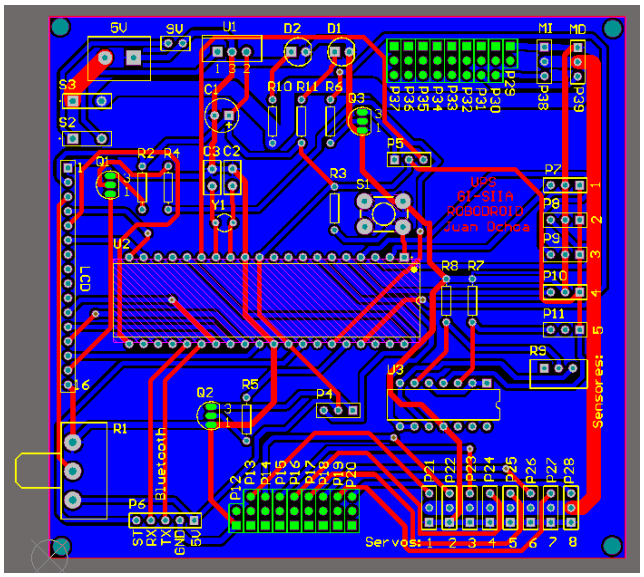


Figura 137.- Ruteado de la placa en Altium

Algunas de las características de ruteado de las pistas que se consideraron para asegurar el correcto funcionamiento o la prevención de fallas de la misma son:

- La ruteado de las pistas es a doble cara.
- Posee plano de tierra para evitar el ruido generado en el ambiente.
- Las pistas de tierra poseen un ancho preferido de 3mm
- Las pistas de control poseen un ancho preferido de 2.5 mm
- La pista de potencia que alimenta a los servomotores y servomotores de rotación continua posee un espesor de 5mm.
- El FootPrint de los transistores fue editado para poder facilitar su soldadura.

La placa concluida se muestra en la Figura 138, y la descripción de la distribución de los elementos más importantes se muestra en la Tabla 48.

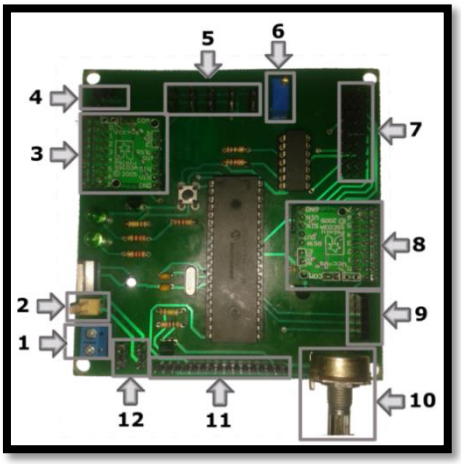


Figura 138.- Placa concluida

Número de elemento	Descripción
1	Conector para la alimentación de potencia, que incluye los ocho servomotores y los dos servomotores de rotación continúa. El voltaje de ingreso puede ser entre 4.8 a 6.0 V.
2	Conector para la alimentación de control. Soporta una voltaje de ingreso de 7.0 a 10.0 V.
3	Segundo controlador "Pololu Micro Serial Servo Controller", encargado de los servomotores de rotación continua.
4	Conectores para los dos servomotores de rotación continua de la plataforma (Izquierdo y Derecho).
5	Conectores para los cuatro sensores de distancia y el sensor de fuerza
6	Potenciómetro para regular la ganancia de la etapa de amplificación del sensor de fuerza.
7	Conectores para cada uno de los ocho servomotores.
8	Primer controlador "Pololu Micro Serial Servo Controller", encargado de los ocho servomotores de la plataforma.
9	Conector para el módulo Bluetooth HC-06.
10	Potenciómetro para regular el contraste de la LCD.
11	Conector para la LCD.
12	Conectores destinados a los interruptores de encendido.

Tabla 48.- Descripción de los Elementos de la Placa

4.3.3 Descripción de los aspectos importantes del software de la plataforma

El software programado en el micro controlador PIC16F877A se encarga de gestionar el protocolo de comunicación que se vio en la sección anterior, así como los distintos elementos de hardware que la plataforma posee. En si el micro controlador es un intermediario entre al Smartphone y la plataforma de desplazamiento, que se encarga de la gestión de los distintos componentes electrónicos y electromecánicos que esta posee dependiendo de la información que envié el Smartphone a través de la tecnología Bluetooth.

La programa desarrollado para el micro controlador fue escrito totalmente en C, mediante el programa CCS desarrollado específicamente para micro controladores PIC, con el fin de obtener la máxima optimización de recursos en estos dispositivos [91]. Posee una amplia librería de funciones predefinidas, comandos de pre procesado y ejemplos; además suministra controladores para diversos dispositivos como LCD, convertidores AD, relojes en tiempo real, EPROM serie, etc. [91]

Los elementos básicos de un programa C de CCS C contiene son [91]:

- **Directivas de preprocesador:** controla la conversión del programa a código maquina

- **Programas o Funciones:** el conjunto de instrucciones. Siempre debe existir un método principal *main().*
- **Instrucciones:** indica cómo se debe comportar el PIC en todo momento
- **Comentarios:** permite comentar cada línea del programa.

Ahora bien, dentro de las configuraciones importantes del micro controlador que se efectúan a través del programa son:

- Se configuro para usar un cristal de alta velocidad (20 MHZ).
- Se desactivo el *Watch Dog Timer*.
- Se utilizan los canales analógicos de AN0 a AN4, con una resolución de 8 bits.
- La velocidad de transmisión de la comunicación serial se configuró a 9600 Baudios.
- Se definió la LCD para ser usada en el puerto D.
- El puerto C y B se definen con las configuraciones estándar

El programa como tal no posee pausas puesto que para cualquier evento se utilizan interrupciones. Para el caso de la recepción serial Bluetooth por el pin C6 se utilizó la interrupción ***INT_RDA***, que es una interrupción por hardware que permite detener momentáneamente el programa en cualquier momento cuando exista datos en cola en el puerto serie.

Para el caso del "Automove" o el barrido que hace con los sensores infrarrojos y refrescar los datos que se muestran en la LCD, se utilizó la interrupción ***INT_TIMER1*** que es lanzada cada vez que se desborda el TIMER1. Para configurar el tiempo en el que se desborda el TIMER1, es necesario el valor de precarga que tendrá, es decir, desde qué valor iniciará el conteo. El tiempo de desbordamiento del TIMER1 es:

$$T = T_{CM} Prescaler(65536 - Carga\ TMR1) \text{ [91]}$$

En donde T_{CM} es el ciclo máquina y está definido como:

$$T_{CM} = \frac{4}{F_{osc}} \text{ [91]}$$

Por lo tanto para este caso tendríamos que:

$$T_{CM} = \frac{4}{20\ MHz} = 2x10^{-7}\ seg$$

Para obtener un tiempo de desbordamiento cada 100 ms, se puede despejar la ecuación del desbordamiento del *timer*, considerando que se usa un *Preescaler* igual a 8, se tiene como resultado lo que se muestra a continuación:

$$Carga\ TMR1 = 65536 - \frac{T}{T_{CM} Prescaler}$$

$$Carga\ TMR1 = 65536 - \frac{100ms}{(2x10^{-4}ms)(8)} = 3036$$

Cuando el *"Automove"* se encuentra activado, cada vez que se desborde el TIMER1 (cada 100 ms), el micro controlador moverá en 2 grados la posición de los 3 últimos servomotores sobre los cuales se montan 3 sensores infrarrojos. Por tanto cada 100 ms se devolverá las lecturas de los sensores de distancia correspondientes. Además dentro del TIMER1 se implementó un contador que cada vez que se desborde 10 veces o 1 segundo se refrescarán los datos que se muestran en la LCD.

La tercera interrupción utilizada es la ***INT_EXT***, que es una interrupción por hardware aplicable al pin B0 que permite detectar pulsos, tanto en flanco de subida como de bajada. Esta interrupción es de gran importancia ya que el pin B0 al cual pertenece va conectado al pin *State* del módulo Bluetooth, lo que permite detectar cuando se ha perdido la comunicación con el smartphone, permitiendo de esta manera detener cualquier proceso o actividad que esté realizando la plataforma a fin de prevenir daños al usuario o a la plataforma.

El diagrama de flujo del programa del micro controlador se muestra en las Figuras 104, 105 y 106 de la sección previa que trata sobre el protocolo de comunicación. El código del micro controlador se puede encontrar en el anexo digital correspondiente.

4.3.4 Diseño y construcción de la estructura mecánica de la plataforma

La plataforma fue diseñada en su totalidad usando el programa de computador Inventor de Autodesk, para posteriormente pasar a su fabricación y ensamblaje. En su mayor parte, la plataforma está construida de aluminio y Nylon. Dicha plataforma consta básicamente de tres elementos:

- Base
- Cabeza
- Pinza

4.3.4.1 Base de la plataforma

Está construida casi en su totalidad de aluminio, a fin de reducir el peso total que tendrá todo el sistema de soporte. Los elementos de ensamblaje son de acero inoxidable.

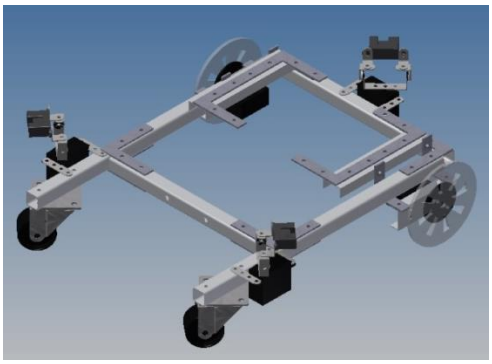


Figura 139.- Captura de la Base de la Plataforma

Características más Importantes:

- Sus dimensiones se pueden revisar en el Anexo 1.
- Soporta 3 servomotores de 3.2 kg.cm de torque, los mismos que se encargan de proveer de movimiento a los tres sensores infrarrojos de distancia.
- Soporta los dos servos de rotación continua encargados de proveer el desplazamiento en cualquier dirección de todo el sistema.
- Sobre ella se ensambla la Pinza y la Cabeza para conformar todo el sistema.
- La ruedas ensambladas a los servomotores de rotación continua son de acrílico de 3mm de espesor y poseen un diámetro de 9.5 cm.
- La velocidad máxima de desplazamiento es de 19.12 cm/seg o 0.68 Km/h
- Las ruedas delanteras son libres, lo que permiten el desplazamiento en todas las direcciones de la plataforma.
- Sobre ella se monta la placa de control y las baterías.

Las direcciones en las que se puede desplazar la base de la plataforma se muestran en la Figura 140.

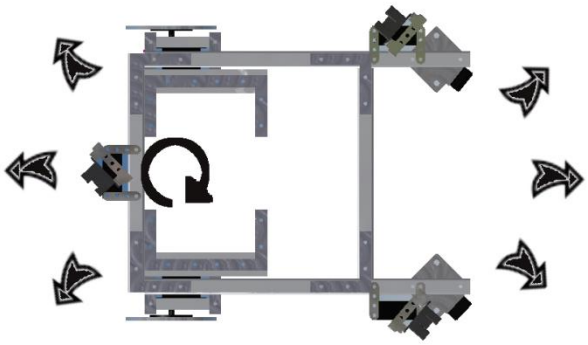


Figura 140.- Direcciones en las que se puede desplazar la plataforma

Para diseños y detalles constructivos más específicos se pueden revisar las láminas adjuntas en los Anexos correspondientes, así como los archivos de AutoCAD e Inventor adjuntos en el disco.

4.3.4.2 Cabeza

La cabeza está fabricada en su totalidad de Nylon y, es el elemento sobre el cual se soportan los diferentes smarthphones que se conectarán vía Bluetooth a la plataforma. En la Figura 141 se muestra una captura de este elemento en cuestión.

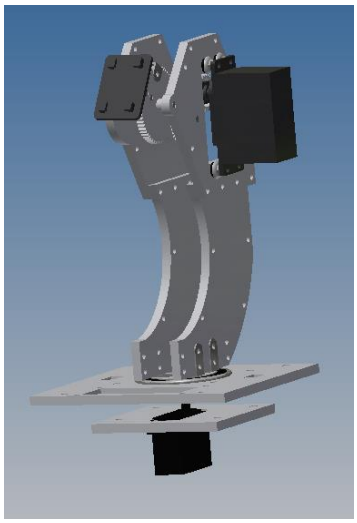


Figura 141.- Captura de la cabeza de la plataforma

Características importantes:

- Está construido en de Nylon de 6mm de espesor.
- Posee un rodamiento en la parte inferior para evitar el desgaste físico del servomotor, así como brindar un elemento de soporte más robusto para los distintos dispositivos que se monten a la plataforma.
- Sus dimensiones se pueden revisar los diseños adjuntos en el Anexo 1.
- Posee dos grados de libertad alrededor del eje Z y el eje X
- Posee un servomotor de 3.5 Kg.cm de torque para proveer el movimiento alrededor del eje Z.
- Posee un servomotor de 11.0 Kg.cm de torque encargado de proveer el movimiento alrededor del eje X.
- Puede sujetar dispositivos móviles que poseen un ancho de 5cm a 18 cm.
- Sobre ella se monta la LCD, así como los dos interruptores de encendido de la plataforma.

En la Tabla 49 se muestran dos sujetadores que pueden ser montados en la cabeza y se utilizan a su vez para sujetar los distintos teléfonos inteligentes

Tabla 49.- Adaptadores para los distintos Smartphones

Los dos grados de libertad que posee la cabeza se muestran en la Figura 142, el movimiento en el eje Z tiene un rango entre 0 a 180 grados y en el eje X un rango de 42 grados.

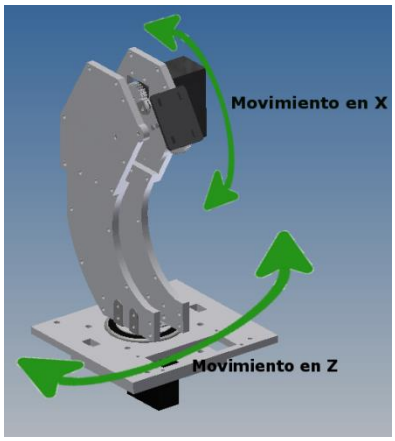


Figura 142.- Grados de Libertad de la Cabeza de la plataforma

En la Figura 143 se muestran los componentes mecánicos que posee en el interior de la cabeza y que se requieren para el movimiento en el eje X. En la Tabla 50 se muestran los parámetros de los tres piñones utilizados para el mecanismo.

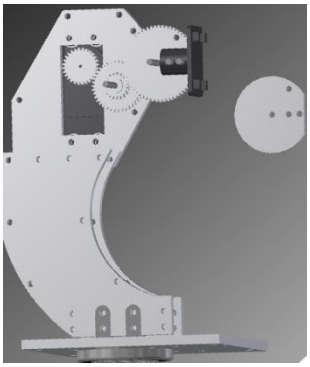


Figura 143.- Mecanismo interno de la Cabeza de la Plataforma

Engrane	Tipo	Radio Primitivo	Módulo	Número de Dientes
1	Motriz	11.2 mm	0.8 mm	Z=28
2	Distribuidor	16 mm 8 mm	0.8 mm 0.8 mm	Z=40 Z=20
3	Conducido	24 mm	0.8 mm	Z=60

Tabla 50.- características de los engranes de la cabeza de la Plataforma

Es necesario conocer cuanta fuerza posee el mecanismo para el movimiento en X, ya que de el dependerá la carga o el peso máximo de los smrathphones que sera capaz de soportar. En la Figura 144 se muestra un diagrama de la fuerzas para el calculo de las mismas.

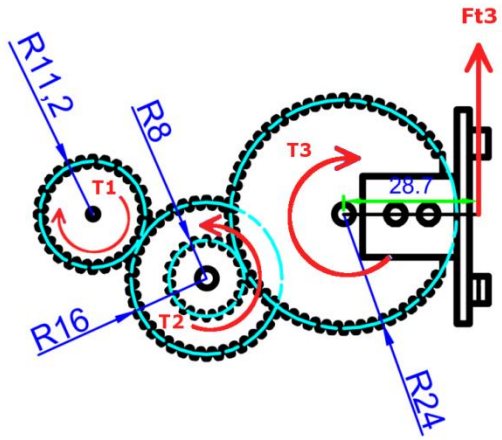


Figura 144.- Diagrama de Fuerzas del mecanismo de la Cabeza

En donde T_1 es el torque del piñon motriz que esta acoplado al servomotor con torque mayor. De ahí que $T_1 = 11.0\ Kg.cm$, por tanto la fuerza tangencial para este piñón estaría dada como:

$$Ft_1 = \frac{T_1}{R_1} = \frac{11.0\ Kg.cm}{1.12\ cm} = 9.82\ Kg$$

Calculando el torque en el segundo piñón o piñón conducido, considerando una eficiencia $e = 90\%$ para tomar en cuenta las pérdidas por fricción se tiene que:

$$T_2 = T_1.r.e = (9.82\ Kg)(1.6\ cm)(0.9) = 14.14\ Kg.cm$$

Y la fuerza tangencial Ft_2 que transmite hacia el engrane conducido está dada como:

$$Ft_2 = \frac{14.14\ Kg.cm}{0.8\ cm} = 17.68\ Kg$$

Por lo tanto el torque T_3 del engrane conducido y una eficiencia $e = 90\%$, estaría dado como:

$$T_3 = T_1.r.e = (17.68\ Kg)(2.4\ cm)(0.9) = 38.18\ Kg.cm$$

Finalmente, la fuerza tangencial Ft_3, la que es necesaria para poder levantar a los dispositivos móviles que se soporten estará dada como:

$$Ft_3 = \frac{38.18\ Kg.cm}{2.87\ cm} = 13.3\ Kg$$

La fuerza tangencial $Ft_3 = 13.3\,Kg$, es más que suficiente para soportar el peso de los smarthphones que se adaptarán a la plataforma.

Para diseños y detalles constructivos más específicos se pueden revisar las láminas adjuntas en los Anexos correspondientes, así como los archivos de AutoCAD e Inventor adjuntos en el disco.

4.3.4.3 Pinza

La pinza es el elemento utilizado para brindar a la plataforma la capacidad de manipular objetos, está construida en su mayoría de Nylon a excepción de las tenazas que fueron impresos en plástico ABS usando una impresora 3D. En la Figura 145 se muestra una captura de este elemento.

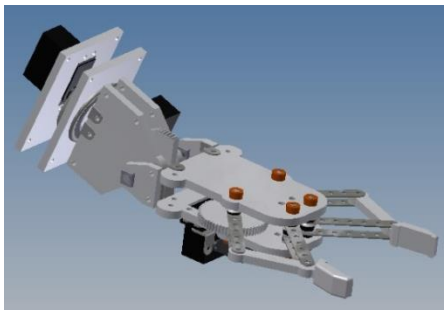


Figura 145.- Captura de la Pinza

Características Importantes:

- Esta construido de Nylon de 6 mm de espesor.
- Las tenazas de la pinza fueron impresas en platico ABS de 5.1mm.
- Las dimensiones de la pinza se pueden observar en al Anexo 1.
- Posee dos grados de libertad, alrededor del eje X y del eje Y.
- Posee dos servomotores de 3.5Kg.cm de torque que se encargan de los movimientos alrededor del eje X y el eje Y.
- Posee un servo motor de 3.5 Kg.cm de torque que se encarga de abrir y cerrar las tenazas de la misma.
- Posee en su parte inferior un sensor infrarrojo de distancia, para estimar la distancia al objeto que se desea manipular.
- Se ensambla en la parte frontal de la plataforma.
- Posee un rodamiento en el elemento que ensambla a la base, para evitar el desgaste del servomotor así como la fricción.

En la Figura 146 se muestra la pienza cuando esta completamente abierta y viceversa.

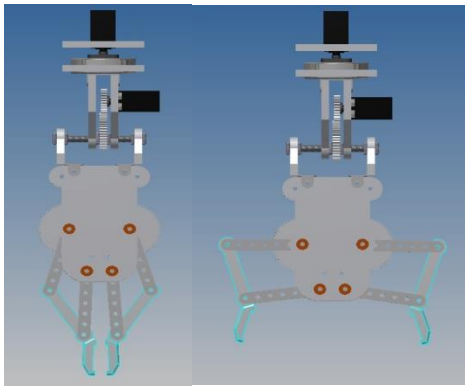


Figura 146.- Pinza totalmente cerrada y totalmente Abierta

En la Figura 147 presentada a continuación se aprecian los dos grados de libertad que posee este mecanismo. El movimiento en X tiene un rango de movilidad entre 0 a 180 grados y el movimiento en Y posee un rango de movilidad de 84 grados. El movimiento en el eje X brinda la funcionalidad de rotar la pinza, y el movimiento en Y permite elevar o bajar la misma.

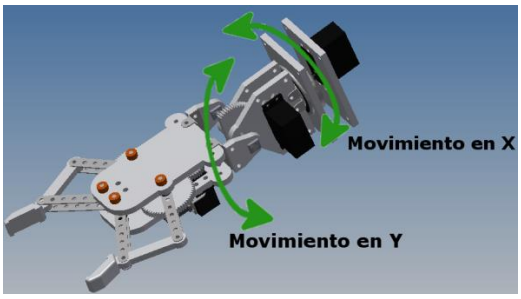


Figura 147.- Grados de libertad de la Pinza

El Figura 148 se puede observar el mecanismo interno de la pinza que sirve para abrir y cerrar las tenazas de la misma. En la Tabla 51 se describe las características que poseen los piñones necesarios para el mecanismo.

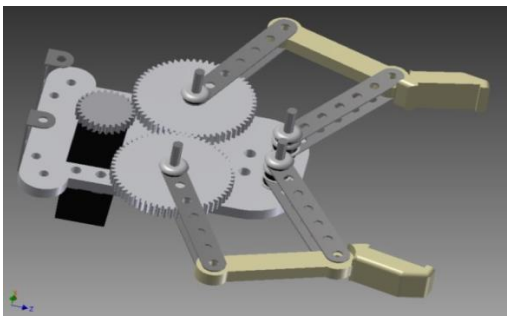


Figura 148.- Mecanismo interno de la Pinza

Engrane	Tipo	Radio Primitivo	Módulo	Número de Dientes
1	Motriz	11.2 mm	0.8 mm	Z=28
2	Conducido	24 mm	0.8 mm	Z=60

Tabla 51.- Características de los engranes de la Pinza

Es necesario conocer la fuerza que el mecanismo aplica sobre las tenazas y por tanto la que aplicará a los objetos que sujete. En la Figura 149 se ilustra un diagrama de las fuerzas en este mecanismo.

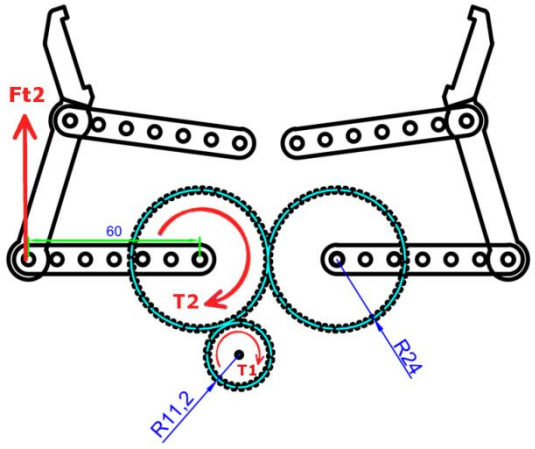


Figura 149.- Diagrama de fuerzas del mecanismo interior de la Pinza

Se considera que T_1 es el torque del piñon motriz acoplado al servomotor, por lo que tenemos que $T_1 = 3.5\ \text{Kg.cm}$. Por tanto la fuerza tangencial Ft_2 que transmite al piñón conducido está dada como:

$$Ft_1 = \frac{T_1}{r_1} = \frac{3.5\ Kg.cm}{1.12\ cm} = 3.125\ Kg$$

El torque en el piñon conducido considerando una eficiencia del 90% será de:

$$T_2 = T_1.r_1.e = (3.125\ Kg)(2.4\ cm)(0.9) = 6.75\ Kg.cm$$

Por lo tanto la fuerza transmitida a las tenazas de la pinza estará dada como:

$$Ft_2 = \frac{T_1}{r_2} = \frac{6.75Kg.cm}{6\ cm} = 1.125\ Kg$$

El mecanismo que se muestra en la Figura 150 provee el movimiento en X, que permite levantar o bajar la pinza y a su vez los objetos que la misma sujete.

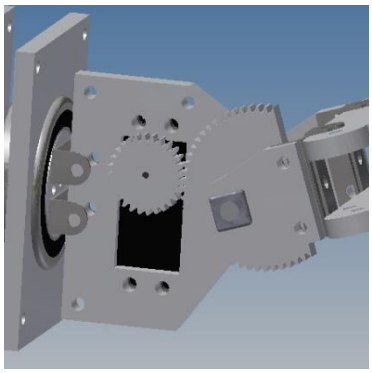


Figura 150.- Mecanismo interior del sistema elevador de la Pinza

Engrane	Tipo	Radio Primitivo	Módulo	Número de Dientes
3	Motriz	11.2 mm	0.8 mm	Z=28
4	Conducido	24 mm	0.8 mm	Z=60

Tabla 52.- Características de los engranes del sistema elevador de la Pinza

Ahora es necesario conocer cuanta fuerza posee el mecanismo para levantar la pinza, para de esta menera estimar el peso máximo que tendrá los objetos que va a levantar.

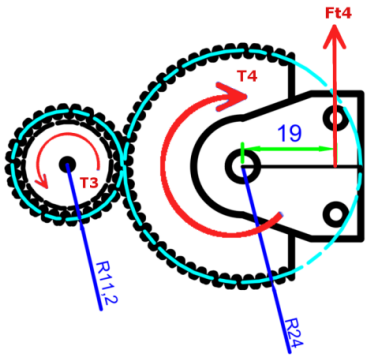


Figura 151.- Diagrama de Fuerzas del sistema elevador de la Pinza

Considerando que T_3 es el torque del piñón motriz acoplado al servomotor, tenemos que $T_3 = 3.5Kg.cm$ y por tanto la fuerza transmitida al piñón conducido estará dada como:

$$Ft_3 = \frac{T_3}{r_3} = \frac{3.5\ Kg.cm}{1.12\ cm} = 3.125\ Kg$$

Por lo tanto el torque en el piñón conducido considerando una eficiencia del 90% será de:

$$T_4 = T_3 r_4 e = (3.125\ Kg)(2.4\ cm)(0.9) = 6.75\ Kg.cm$$

Ahora la fuerza tangencial Ft_4 que es usada para levantar la pinza será de:

$$Ft_4 = \frac{6.75\ Kg.cm}{1.9\ cm} = 3.55\ Kg$$

Para diseños y detalles constructivos más específicos se pueden revisar las láminas adjuntas en los Anexos correspondientes, así como los archivos de AutoCAD e Inventor adjuntos en el disco.

4.3.5 Ensamblaje Total y distribución de los elementos

En esta sección se muestra la plataforma concluida como tal, así como la distribución de los elementos electrónicos y electromecánicos, y como estos se asocian con el protocolo de comunicación. En el Anexo 2 se pueden ver capturas de la construcción y ensamblaje de la plataforma.

En la Figura 152 que se muestra a continuación se puede observar una captura de pantalla de la plataforma ensamblada en Inventor, así como una fotografía de la plataforma finalmente construida.

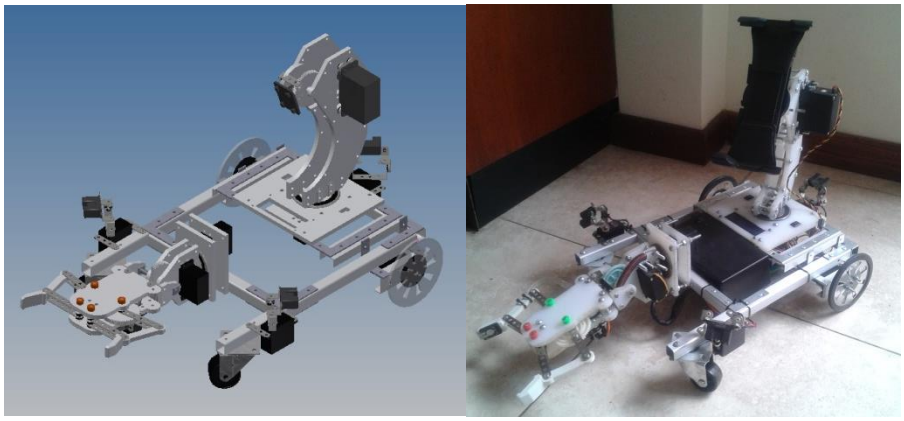


Figura 152.- Plataforma diseñada en Inventor (izquierda) y la fotografía de la plataforma ensamblada.

La distribución de los elementos de la plataforma y se muestra en la Tabla 53.

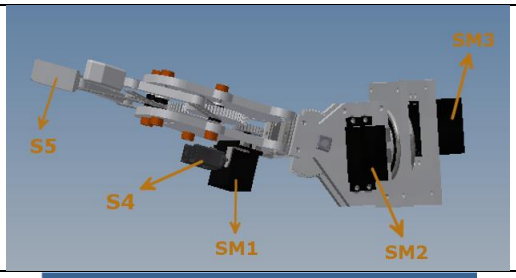
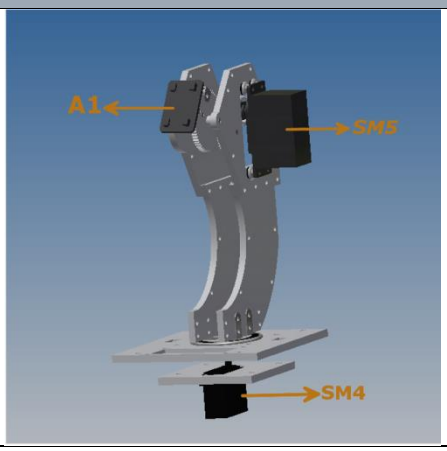
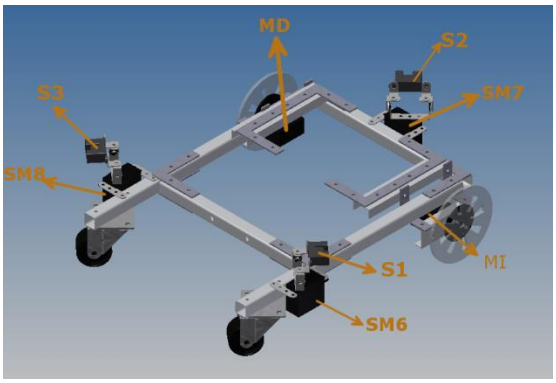
	• SM1: servo motor número 1 • SM2: servo motor número 2 • SM3: servo motor número 3 • S4: sensor número 4; sensor de distancia infrarrojo. • S5, sensor número 5; sensor de fuerza.
	• A1: adaptador para los sujetadores de los dispositivos inteligentes • SM4: servo motor número 4 • SM5: servo motor número 5
	• MD: servo motor de rotación continua número 1. • MI: servo motor de rotación continúa número 2. • SM6: servo motor número 6 • SM7: Servo motor número 7 • SM8: Servo motor número 8 • S1: sensor de distancia infrarrojo número 1. • S2: sensor de distancia infrarrojo número 2 • S3: sensor de distancia infrarrojo número 3.

Tabla 53.- Distribución de los elementos de la Plataforma

4.4 Pruebas de funcionamiento

En esta sección se tratan las pruebas que se realizaron a fin de comprobar el correcto funcionamiento del protocolo de comunicación, así como la respuesta de la plataforma ante cada comando enviado hacia ella.

Las pruebas realizadas se efectuarán usando las aplicaciones previamente descritas ya en este capítulo, usando los distintos dispositivos Android que se describieron al inicio de este capítulo.

4.4.1 Plan de pruebas

En el plan de pruebas se consideran los aspectos como los comandos enviados y las respuestas que tiene la plataforma ante ellos. Para ello se enviaron los distintos comandos que se trataron en la sección 4.2.3 que habla sobre el protocolo de comunicación, tomando en consideración los siguientes aspectos:

- Se enviaron tramas con todos los bytes correctos.
- Se enviaron tramas con el byte de sincronización erróneo.
- Se enviaron tramas con el byte de comando erróneo.
- Se enviaron tramas con errores dentro de los bytes que conforman el paquete "Data".
- Se enviaron tramas repetidas.
- Se enviaron tramas incompletas.
- Se enviaron tramas con un número mayor de bytes que lo normal.

Con las pruebas se espera que la plataforma:

- Configure la velocidad de los 8 servomotores cuando reciba el comando correcto, y por el contrario, no lo haga si existe un dato erróneo o información repetida
- Configure la velocidad y la dirección de los 2 servomotores de rotación continua, cuando reciba el comando correcto, y viceversa, que no lo haga si existe un dato erróneo o información repetida
- Configure la posición de los 8 servomotores cuando reciba el comando correcto, por el contrario no lo haga si existe un dato erróneo o información repetida
- Se active o desactive el *"Automove"* ante el comando adecuado.
- Configure los parámetros de la LCD ante los comandos correctos.
- Devuelva las lecturas de los sensores adecuados ante el comando correcto.

Las pruebas evaluarán si las respuestas a los distintos comandos enviados a la plataforma son las esperadas, calificándolas como "Correctas" o "Incorrectas". Finalmente se verificará si los componentes mecánicos se comportan según lo esperado, además de que la plataforma detenga correctamente todo proceso cuando se pierda la conexión Bluetooth.

4.4.2 Resultados de las pruebas del protocolo de comunicación y la plataforma

Las pruebas se realizaron usando primero la aplicación "TestProtocolo" que como se vio en las secciones anteriores permitía enviar tramas ingresando el número de bytes así como su valor. En la Tabla 54 se muestran las capturas de la aplicación cuando se realizaron las distintas tramas para las pruebas.

Tabla 54.- Capturas de la aplicación "TestProtocolo" durante las pruebas

En la Tabla 55 se muestra una fracción de todos los comandos enviados a la plataforma, así como la respuesta a cada una las tramas. La tabla completa de todas las tramas enviadas para las pruebas se muestran en el Anexo 3.

Comandos enviados(Bytes)										Respuesta Esperada	
Sincro	Comando	Data									
1	2	3	4	5	6	7	8	9	10	Correcto	Incorrecto
100	1	1	0							X	
100	1	1								X	
255	2	20	20	20	20	20	20	20	20	X	
255	2	140	30	40	50	60	70	80	90	X	
255	3	180	50	60	20	30	60	70	180	X	
255	4	1	50	1	50					X	
255	4	1	100	1	100	90	10			X	
255	5	2								X	
255	6									X	
255	6	1								X	

Tabla 55.- Fracción de comandos enviados durante las Pruebas del Protocolo de Comunicación

En total se enviaron un total de 80 tramas, el porcentaje de comandos exitosos se muestran en la Figura 153.

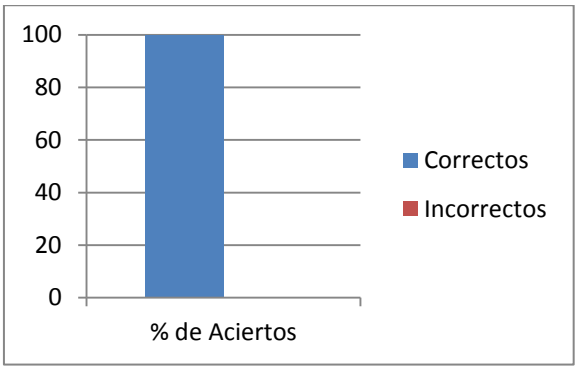


Figura 153.- Gráfica del porcentaje de Aciertos en las pruebas del Protocolo de Comunicación

La siguiente prueba se realiza utilizando la aplicación "Robodroid", la misma posee más elementos distribuidos para permitir un control más fácil de la plataforma. Las capturas de la aplicación durante las pruebas se muestran en la Tabla 56.

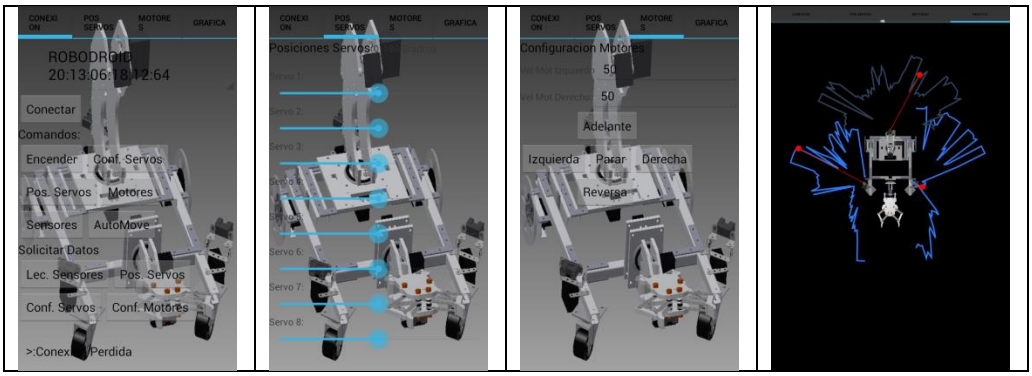


Tabla 56.-Capturas de la aplicación "Robodroid" durante las pruebas

Una fracción de los elementos que se evaluaron se muestra en la Tabla 57. Todos los elementos evaluados se muestran en el Anexo 3.

Elemento a Evaluar	Respuesta	
	Si	No
¿Al presionar el botón "Encender", de la sección de comandos, la LCD de la plataforma se encendió?	X	
¿Al presionar el botón "Sensores" en la sección de comandos, la plataforma mostro las lecturas de los 5 sensores en la LCD?	X	
¿Cuándo se presiona el botón "Automove" cuando el *Automove* está inactivo, este se activa?	X	
¿Al presionar el botón "Conf. Servos" de la sección Solicitar Datos, la	X	

plataforma devuelve la configuración de velocidad de los 8 servomotores?		
¿Cuándo cambia cualquiera de los valores de las 8 barras de progreso, las posiciones de cada uno de los 8 servomotores correspondientes cambia acorde a dichos valores?	X	
¿Al presionar el botón "Adelante" la plataforma se desplaza hacia adelante con la velocidad ingresada?	X	

Tabla 57.- Fracción de la pruebas realizadas con la aplicación "Robodroid"

El porcentaje de aciertos para este punto se muestran en la Figura 154.

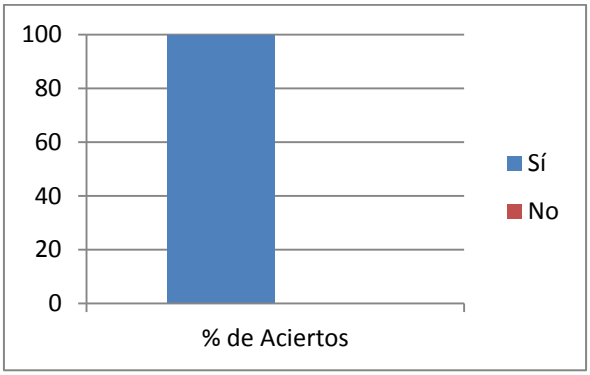


Figura 154.- Porcentaje de acierto de las pruebas realizadas con la aplicación "Robodroid"

Una vez concluidas las pruebas y que todas fueron exitosas, se puede afirmar que la plataforma así como el protocolo de comunicación están listos para los pasos subsecuentes que conllevan al ensamblaje final del Asistente Personal Autónomo.

CAPITULO 5: DESARROLLO E IMPLEMENTACIÓN DE LA APLICACIÓN EN EL DISPOSITIVO MÓVIL.

5.1 Análisis de requerimientos funcionales

En esta sección se analizarán los requisitos funcionales, características, así como los recursos que requerirá la aplicación a instalar en dispositivos móviles Android y que es necesaria para convertir al smartphone y a la plataforma en un asistente personal autónomo como tal.

5.1.1 Recursos de la aplicación

Dentro de los recursos que posee el dispositivo inteligente y a los cuales la aplicación debe tener acceso se encuentran:

- Acceso a la cámara frontal de la cámara del dispositivo si este lo posee
- Acceso a Internet
- Acceso al Adaptador Bluetooth del dispositivo inteligente
- Acceso a los puertos TCP/IP del dispositivo inteligente
- Almacenamiento de datos
- Acceso al micrófono
- Acceso a los parlantes
- Manejo de los recursos táctiles de la pantalla.
- Acceso al sensor de luz si el dispositivo lo posee
- Acceso al acelerómetro
- Capacidad de almacenamiento

5.1.2 Requisitos y características de la aplicación

Se debe considerar que la aplicación instalada en el dispositivo inteligente, será la encargada de convertir al mismo integrado conjuntamente con la plataforma, en un Sistema de Asistencia Personal, por lo que debe cumplir algunos requisitos y características, entre los cuales se tiene:

- Ser compatible con la mayor cantidad de dispositivos móviles Android existentes en el mercado.
- Soportar desde la versión 2.3 de Android.
- Ser capaz de manejar los recursos gráficos y sonidos.
- Capacidad de manejar tareas asíncronas.
- Poseer un avatar interactivo con el cual el usuario pueda interactuar.
- Contar con un sistema que se encargue del control del protocolo de comunicación con la plataforma.
- Realizar la adquisición de las lecturas de sensores enviados desde la plataforma.
- Capacidad de conectarse a un servidor de asistencia remota vía TCP/IP.
- La capacidad de interacción con el usuario utilizando la visión Artificial

- Poseer actividades interactivas basadas en la realidad aumentada y visión artificial
- Capacidad de reconocer cuatro gestos realizados con la mano.
- Capacidad de detectar y reconocer rostros humanos mediante la visión artificial
- Capacidad reconocer tres objetos mediante la visión artificial.
- Emplear técnicas de conversión de texto a voz.
- Capacidad de leer correo electrónicos.
- Gestión de una base de datos para el almacenamiento de información.
- Emplear actividades que requieran cierta autonomía, a fin de que el robot cuente con un grado básico de naturalidad
- Incorporar actividades lúdico-recreativas para los niños que interactúen con el sistema.
- Incorporar tareas que puedan ayudar durante el proceso desarrollo o terapia de personas con discapacidad.

5.2 Diseño modular y UML.

En esta sección se analizarán los elementos y componentes que conforman la totalidad de la aplicación de asistencia "Anbaro" (Android Based Robot).

5.2.1 Funcionamiento general del sistema

Antes de presentar el diagrama modular y el diagrama UML de la aplicación, es necesario comprender cómo funcionará el sistema, visto como un conjunto. En la Figura 155 se puede apreciar un diagrama de bloques que representa nuestro sistema.

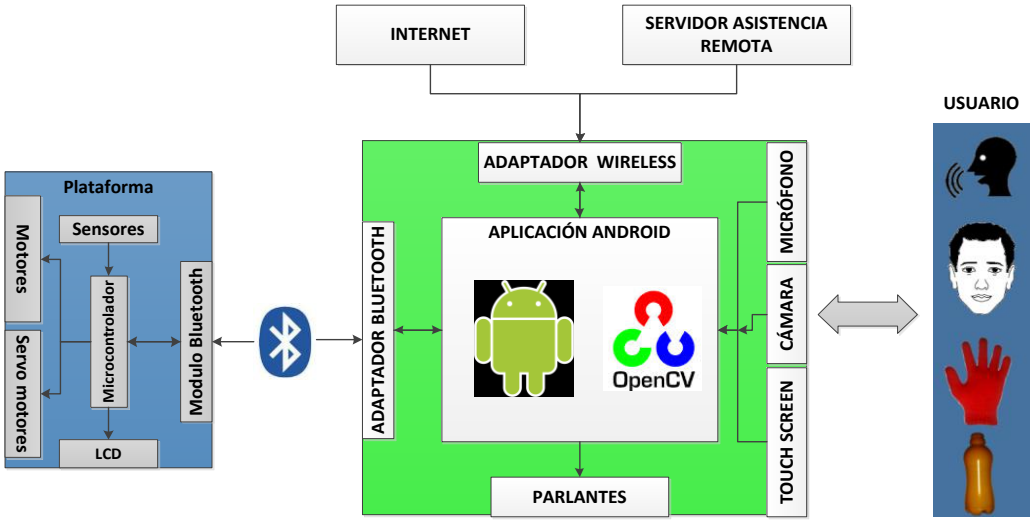


Figura 155.- Diagrama de Bloques Descriptivo del Sistema

El usuario podrá interactuar con la aplicación usando la cámara, la pantalla táctica del dispositivo móvil (*Touch Screen*) o el micrófono del dispositivo inteligente. Para cada interacción que el usuario tenga con el dispositivo móvil la respuesta de la aplicación y por tanto, de todo el sistema, será diferente. Además, podemos apreciar que la aplicación posee

la capacidad de gestionar recursos de hardware como el adaptador Bluetooth, elemento que es necesario para comunicarse con la plataforma; el adaptador Wireless, que permite a la misma acceder a internet, así como conectarse el servidor de asistencia remota diseñado exclusivamente para este proyecto. Por ello, es importante mencionar que existe una gran cantidad de recursos y componentes que la aplicación gestiona a fin de convertir a un teléfono inteligente en un asistente personal autónomo que pueda interactuar con los usuarios. En las secciones que siguen se describirá con mayor detalle cada uno de los componentes de la aplicación, cómo funcionan y cómo se relacionan entre ellos.

5.2.2 Diseño modular de la aplicación.

En la Figura 156 se puede apreciar el diagrama modular de toda la aplicación, así como los componentes que posee cada módulo.

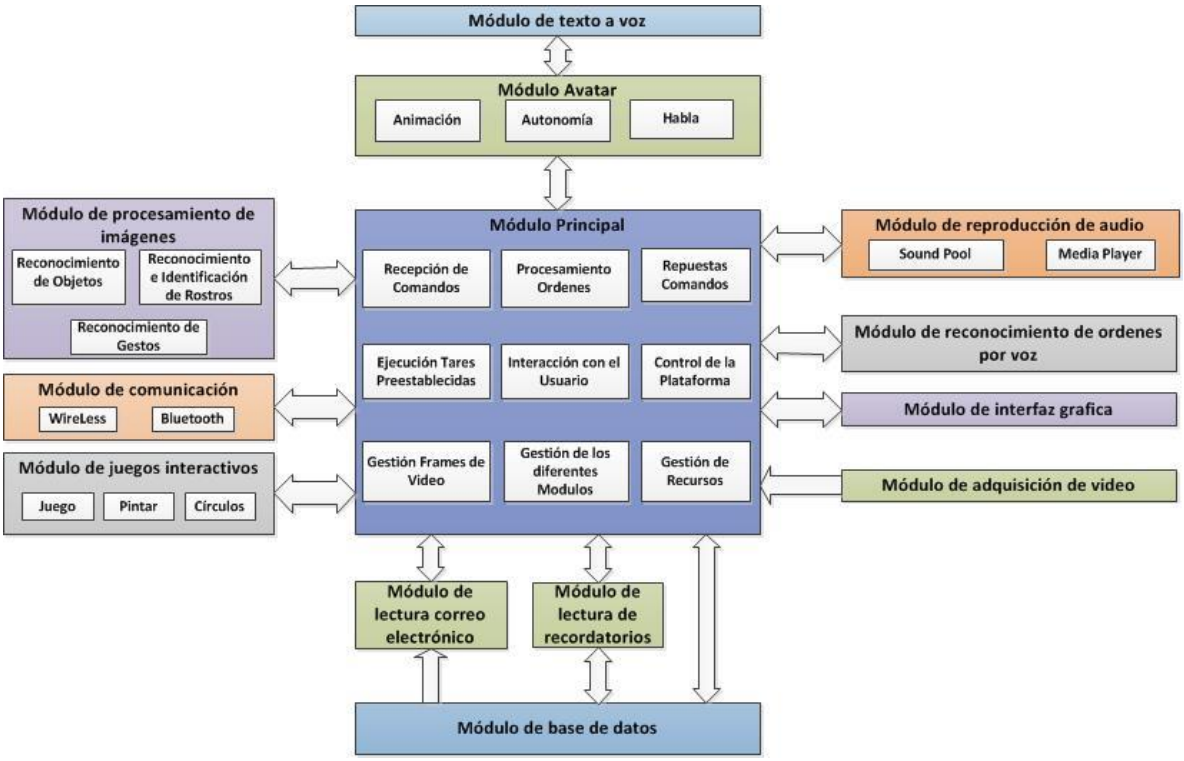


Figura 156.- Diagrama Modular de la Aplicación

A continuación se detallan las características y funcionalidades más importantes de cada módulo:

- Módulo de interfaz gráfica: está constituida por todos los elementos gráficos que se presentarán en la pantalla del dispositivo y que servirán como medio de interacción con la aplicación.

- Módulo de reconocimiento de órdenes por voz: se encarga de transformar los comandos de voz emitidos por el usuario en texto, a fin de procesarlos posteriormente.
- Módulo de adquisición de video: emplea la librería OpenCV lo que le permite capturar imágenes o video que a través de la cámara del dispositivo. Con ello será posible analizar las imágenes para realizar detección y reconocimiento facial o identificación de gestos u objetos.
- Módulo de reproducción de audio: se encarga de la reproducción de efectos de sonido y archivos de audio mediante las clases de Android *SoundPool* y *MediaPlayer*, respectivamente.
- Módulo de texto a voz: es uno de los módulos más importantes, puesto que con él se puede interactuar con el usuario de una manera más natural. Permite que el asistente reproduzca sus mensajes empleando una voz similar a la de las personas.
- Módulo Avatar: él se encarga de la gestión de los elementos que conforman el avatar virtual que se presentará en la pantalla del dispositivo. Es el encargado de brindarle las animaciones necesarias como las expresiones faciales o la capacidad de simular que habla.
- Módulo de lectura de correo electrónico: permite leer automáticamente los correos electrónicos del correo que el usuario haya almacenado.
- Módulo de lectura de recordatorios: permite acceder a los recordatorios que el usuario haya ingresado, para posteriormente leerlos cuando sea oportuno.
- Módulo de base de datos: se encarga de almacenar los recordatorios, así como el correo electrónico ingresado por el usuario a través de la interfaz gráfica.
- Módulo de procesamiento de imágenes: este módulo es de gran importancia, ya que permite procesar las imágenes adquiridas por la cámara del dispositivo, a fin de realizar la detección e identificación de rostros, reconocimiento de objetos y de los gestos que se realicen con la mano.
- Módulo de comunicación: se encarga de gestionar las comunicaciones, como son el acceso a internet, conexión al servidor de asistencia remoto y la conexión con la plataforma móvil.
- Módulo de juegos interactivos: gestiona los tres juegos basados en la realidad aumentada gracias al trabajo previo realizado en el módulo de procesamiento de imágenes.
- Módulo Principal: es el modulo central encargado de la gestión de todos los recursos de hardware y software del dispositivo, procesamiento de órdenes y comandos; es el encargado de gestionar todos los módulos previamente citados. En él se encuentra la lógica del sistema de asistencia personal.

5.2.3 Diagrama UML de clases

Para comprender el desarrollo de la aplicación y los componentes en detalle, se ha seleccionado el diagrama estandarizado de clases UML. En la Figura 157 se muestra un diagrama clases simplificado, debido a que el diagrama completo es muy extenso. Para revisar cada clase, variable y método que posee el sistema, se puede acceder al Anexo Digital correspondiente. En la Figura 157 podemos apreciar cada una de las clases y su relación dentro de la aplicación.

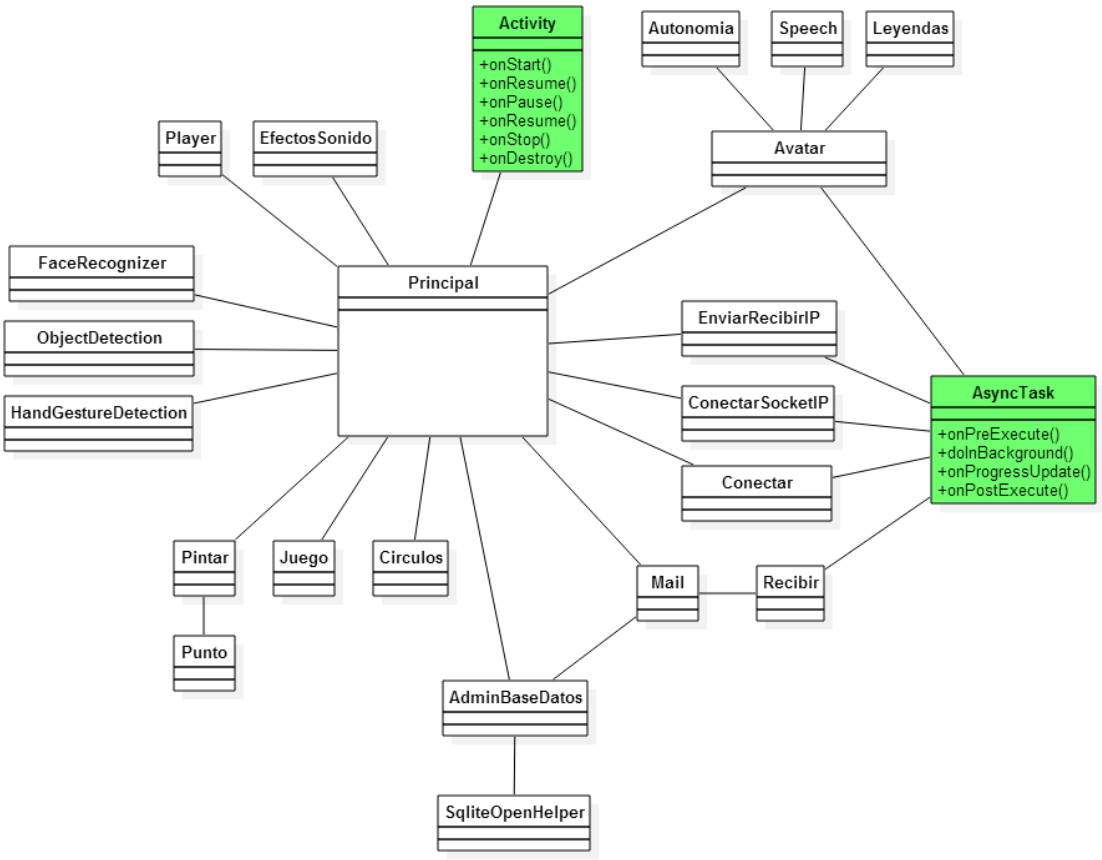


Figura 157.- Diagrama UML de la Aplicación

5.3 Desarrollo e implementación de la aplicación.

En esta sección se describirán los elementos que conforman la aplicación de asistencia personal, así como las características fundamentales de cada etapa de desarrollo. El código completo y la aplicación compilada se encuentran disponibles en el Anexo Digital, si se desea ahondar en más detalles.

Hay elementos en esta sección que ya fueron descritos en el Capítulo 2, por lo que no se profundizará más en ellos; al contrario existen partes fundamentales como las técnicas de procesamiento de imágenes que se usaron para trabajar con realidad aumentada y la interacción con el asistente personal, las que sí se profundizará.

5.3.1. Detalles preliminares

La aplicación fue diseñada para trabajar en entornos que van desde la versión 2.3 de Android (GingerBread) y fue compilada para la versión 4.4 KitKat. Asimismo, es importante destacar que es compatible con las futuras versiones de Android que puedan surgir. Sin embargo, hay que considerar que para el uso de la aplicación es necesario que el dispositivo posea una cámara frontal.

La aplicación posee una actividad principal, la misma se encuentra declarada dentro del Android Manifest. Además, se definió que la orientación que usará la aplicación es horizontal (*landscape*).

A fin de que la aplicación pueda acceder a varios componentes fundamentales del hardware del dispositivo móvil inteligente, es necesario agregar los permisos necesarios dentro del archivo *Manifest* de Android. Entre los permisos a solicitar destacamos los siguientes:

- Acceso al estado de la red
- Acceso a internet
- Acceso al adaptador Bluetooth
- Acceso a la cámara

En la Figura 158 se puede apreciar los permisos declarados dentro del Manifest.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.BLUETOOTH_PRIVILEGED"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

Figura 158.- Permisos de la Aplicación declarados dentro del Manifest

El proyecto de la aplicación posee un total de 10 clases escritas en Java que le brindan todas las capacidades de "asistente personal autónomo" al dispositivo móvil inteligente. Tres clases sirven únicamente como interfaz con otras tres clases que se desarrollaron totalmente en código nativo (C/C++). En la Figura 159 se muestra una captura de pantalla de las clases que contiene el proyecto.

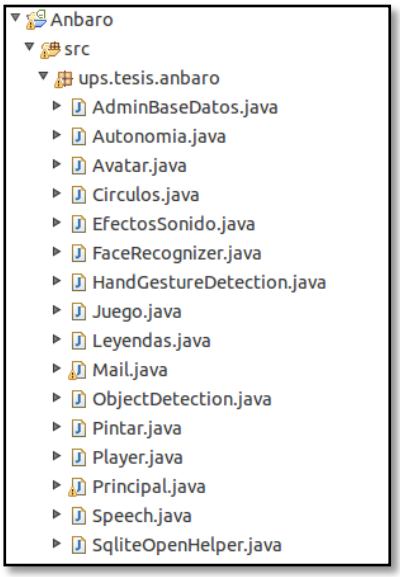


Figura 159.- Estructura del Proyecto de la Aplicación

Las actividades relacionadas con el reconocimiento de patrones, como son la identificación de gestos, objetos, así como la detección y el reconocimiento facial, se ejecutan en código nativo por dos razones fundamentales: la primera es que algunos métodos de OpenCV no están disponibles directamente en Java y la segunda es que el rendimiento mejora en alguna medida cuando los procesos exhaustivos como estos se ejecutan directamente como librerías del sistema operativo y no pasan por la máquina virtual de Android Dalvik. La aplicación posee tres clases nativas:

- *NativeFaceRecognition*
- *NativeHandDetection*
- *NativeObjectDetection*

En la Figura 160 se muestra las clases nativas dentro del directorio JNI (*Java Native Interface*) de nuestro proyecto

Figura 160.- Carpeta JNI del proyecto de la Aplicación

Dentro del **Android.mk** se definen algunas características con las que se compilará el código nativo, como es la versión mínima de Android y las arquitecturas de procesadores a las cuales se podrán soportar. En la Figura 161 se puede apreciar que se ha establecido como la versión mínima de Android la 10, que corresponde a la 2.3.3 (Gingerbread). Además, se definió que serán soportadas todas las arquitecturas procesadores existentes hasta la actualidad (2014).

```
APP_STL := gnustl_static
APP_CPPFLAGS := -frtti -fexceptions
APP_ABI := all
APP_PLATFORM := android-10
```

Figura 161.- Contenido del Archivo Android.mk

Dentro del archivo **Aplication.mk** se definen los parámetros de nuestra librería, en este caso se incluyen los archivos necesarios para la compilación, se especifican las clases nativas que integrarán la misma y el nombre que llevará. En la Figura 162 se puede apreciar la definición de este archivo dentro del proyecto. El nombre de la librería nativa que será vinculada con la parte del código escrito en Java es "*NativeProcessing*".

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

#OPENCV_CAMERA_MODULES:=off
#OPENCV_INSTALL_MODULES:=off
#OPENCV_LIB_TYPE:=SHARED
include /home/sebastian/android/OpenCV-2.4.9-android-sdk/sdk/native/jni/OpenCV.mk

LOCAL_SRC_FILES  := NativeHandDetection.cpp NativeFaceRecognition.cpp NativeObjectDetection.cpp
LOCAL_C_INCLUDES += $(LOCAL_PATH)
LOCAL_LDLIBS     += -llog -ldl

LOCAL_MODULE     := NativeProcessing

include $(BUILD_SHARED_LIBRARY)
```

Figura 162.-Contenido del Archivo Aplication.mk

La aplicación posee dos *layout*, el primero "activity_main.xml" contiene toda la interfaz gráfica de la aplicación y se vincula directamente con la actividad *Principal*; el segundo *layout* "borde.xml" es simplemente un estilo para definir el color de fondo y el color del borde de los diferentes contenedores del *layout* "activity_main.xml", su finalidad es únicamente estética. En la Figura 163 se muestran los *layouts* dentro del directorio correspondiente a nuestro proyecto.

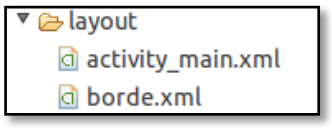


Figura 163.- Layouts de la Aplicación

Finalmente, todos los archivos adicionales que requiere la aplicación tales como imágenes, fondos, iconos, sonidos y demás archivos, se encuentran dentro del directorio *assets* o r*aw* de nuestro proyecto.

5.3.2 Descripción de los componentes de la aplicación

5.3.2.1 Reproducción de archivos de Audio

Dentro de la aplicación existen dos métodos para la reproducción de archivos de audio, cada uno tiene distintas características así como funciones específicas.

Media Player: El Media Player soporta la reproducción de archivos de audio en distintos formatos. Android provee todos los métodos necesarios para el control y administración del mismo. En la Figura 164 se puede apreciar el diagrama de flujo del funcionamiento del Media Player dentro de nuestro sistema.

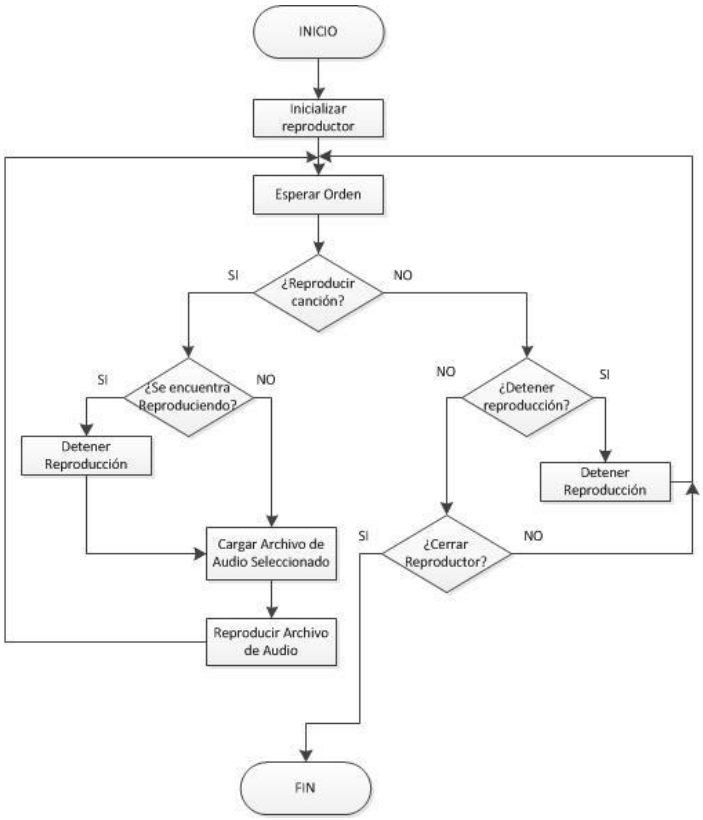


Figura 164.- Diagrama de flujo para el uso del Media Player

Como aprecia en el diagrama de flujo anterior, es posible reproducir los archivos de audio indicados, permitiéndonos detectar si el reproductor está ocupado con un trabajo previo o si se encuentra libre y listo para la reproducción de un nuevo archivo de audio. Cabe recalcar que este método posibilita la reproducción de un solo archivo de audio a la vez. Para el caso de nuestro proyecto se incluyó dentro del directorio *assets* diez canciones infantiles que pueden ser reproducidas con este método. La lista de las canciones disponibles y su identificador se puede apreciar en la Tabla 58.

ID	Descripción	Formato
1	Antón Pirulero	Mp3
2	La cucaracha	Mp3
3	Abuelito dime tú	Mp3
4	Al carro de la Patata	Mp3
5	Al Pasar la Barca	Mp3
6	El Cocherito	Mp3
7	Los Elefantes	Mp3
8	Hola Don pepito	Mp3
9	El Grillo Cri Cri cri	Mp3
10	Barquito de Papel	Mp3

Tabla 58.- Archivos de Audio que posee la Aplicación

Sound Pool: este método es otro medio para la reproducción de archivos de audio que a diferencia del anterior permite reproducir varios sonidos al mismo tiempo, pero el tamaño de los mismos no debe ser demasiado grande, por lo que es ideal para ser aplicado en juegos (efectos de sonido). En la Figura 165 se puede apreciar el diagrama de flujo para nuestro caso.

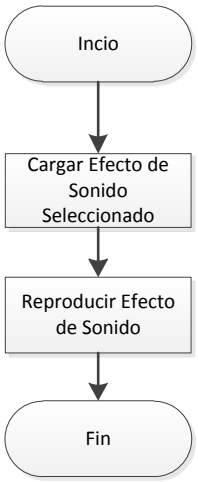


Figura 165.- Diagrama de Flujo para el manejo del Sound Pool

La aplicación posee 12 efectos entre sonidos ambientales y sonidos de animales, en la Tabla 59 se puede apreciar los sonidos que posee. Los archivos necesarios se almacena en el directorio *assets* de nuestro proyecto.

ID	Descripción	Formato
1	Aplausos multitud	Mp3
2	Estornudo Hombre	Mp3
3	Ronquido Hombre	Mp3
4	Ronquido Hombre	Mp3
5	Risas Bebe	Mp3
6	Gato Maullando	Mp3
7	Vaca Mugiendo	Mp3
8	Perros Ladrando	Mp3
9	Elefante Barritando	Mp3
10	Gallo Cantando	Mp3
11	Caballo Relinchando	Mp3
12	León rugiendo	Mp3

Tabla 59.- Efectos de sonidos que posee la Aplicación

5.3.2.2 Base de Datos

La base de datos se usa para almacenar información de manera permanente en el dispositivo. En nuestra aplicación se emplea la versión SQLite que casi posee las mismas funciones que las bases de datos para ordenadores. En la base de datos se crearon dos tablas: “CorreoElectronico” y “Recordatorios”. La tabla “CorreoElectrónico” posee dos campos de tipo cadena (String): correo y contraseña.

En esta tabla se puede almacenar el correo electrónico al que queremos que el asistente personal acceda y realice la lectura. Por otra parte, la tabla “Recordatorios” contiene cuatro campos de tipo cadena (String) que son:

- ID
- Hora
- Fecha
- Recordatorio

Con esta tabla se hace posible almacenar textos descriptivos que el usuario desea que el robot recuerde en una fecha y hora específica, para luego hacer la lectura de los mimos en el momento oportuno.

A fin de acceder de forma adecuada a la base de datos, se creó la clase de *Administración*, que posee métodos, para escritura, lectura y eliminación de datos de cada una de las tablas antes mencionadas.

5.3.2.3 Lectura de correos Electrónicos

Cada vez que se reciba un correo electrónico, el asistente personal autónomo procederá de manera automática a realizar la lectura del mismo, consultando para ello la dirección de correo que el usuario haya registrado en el asistente personal. El servidor de correo que soporta el sistema es GMail, y para conectarse a los servicios del mismo se utilizó la API Java Mail. En la Figura 166 se muestra el diagrama de flujo para la lectura de correos electrónicos.

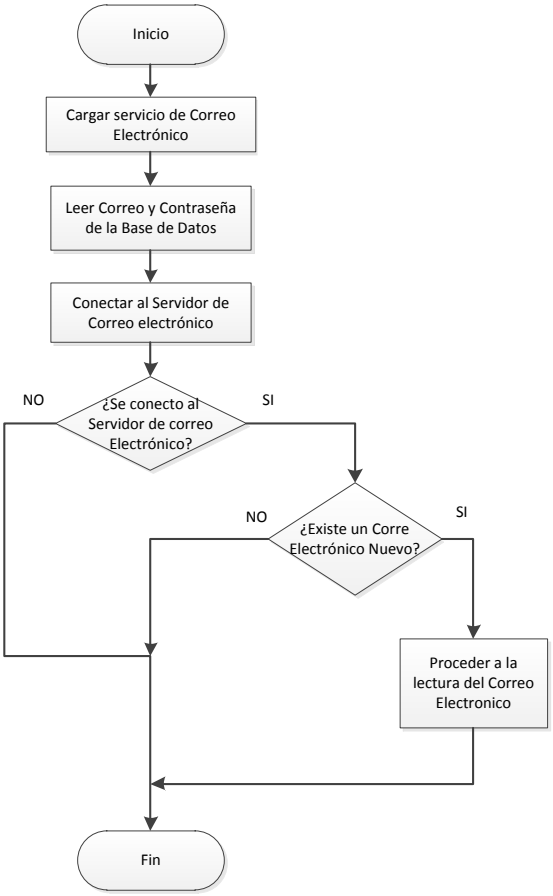


Figura 166.- Diagrama de Flujo para la Lectura de Correos Electrónicos

Mientras la aplicación este activa y exista conexión a Internet, siempre se estará verificando si existe un correo electrónico nuevo en la bandeja de entrada, esto gracias a que este proceso se realiza en segundo plano mediante una tarea asíncrona.

5.3.2.4. Lectura recordatorios

Un recordatorio es un elemento fundamental si se desea que el robot pueda servir como asistente personal, por ello, en la aplicación se permite almacenar recordatorios e indicar la hora y la fecha en la que se quiere que se le presente al usuario el evento. Este proceso funciona consultando la hora y fecha actual del sistema, para luego comparar si la misma coincide con algún recordatorio almacenado en la base de datos con dicha hora y fecha, si es así, la aplicación procederá a la lectura del mismo.

En la Figura 167 se muestra el diagrama de flujo de la lectura de recordatorios

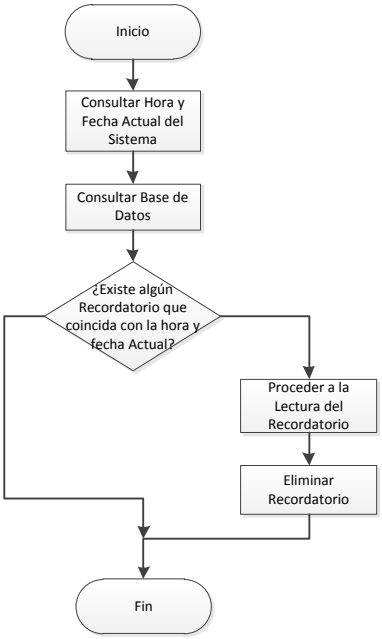


Figura 167.- Diagrama de Flujo para la Lectura de Recordatorios

5.3.2.5 Reconocimiento de órdenes por voz

El reconocimiento de ordenes por voz es uno de los medio utilizados para la interacción con la aplicación, ya que a determinados comandos la aplicación responderá con ciertas actividades o resultados. Debido que el procesamiento de imágenes es un proceso que consume muchos recursos, fue necesario optimizar la forma en cómo se procesaba la voz de cualquier usuario y la mejor solución fue utilizar la Google Speech API. Dicha API permite lanzar el reconocimiento por voz por defecto y una vez que identifica las órdenes dictadas las devuelve como texto un listado de posibles palabras pronunciadas y ordenadas por el nivel de probabilidad.

El lanzamiento de la actividad de reconocimiento de ordenes por voz se realiza por medio de un intento, de manera similar a como se hacía cuando se encendía el Adaptador Bluetooth (empleando el método ***onActivityResult()***). En la Figura 168 podemos apreciar la imagen cuando el reconocimiento por voz es lanzado

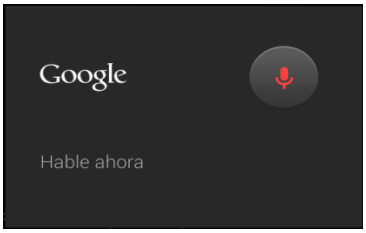


Figura 168.- Reconocimiento de Ordenes por Voz de Google

En la Figura 169 podemos apreciar el diagrama de flujo aplicado en este proyecto para el uso del reconocimiento de órdenes por voz.

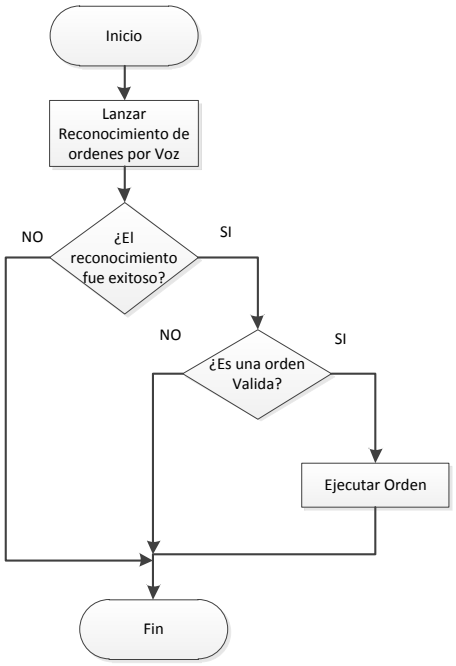


Figura 169.- Diagrama de Flujo para ale Reconocimiento de Ordenes por Voz

En la Tabla 60 se muestran los comandos que la aplicación acepta, así como los resultados que tendrá cada uno.

Orden	Resultado
Saluda	El asistente personal procederá a presentarse
Futuro	El asistente personal hablará sobre su futuros
Funciones	El asistente personal hablará sobres las funciones que posee actualmente
Cantar	El asistente personal procederá a reproducir una canción aleatoria de las que posee en su base de datos
Jugar	Se lanzará el Juego numero 1
Pintar	Se lanzará el Juego numero 2
Juego	Se lanzará el juego numero 3
Seguir Cara	Se activará el reconocimiento de rostros y se procederá a seguir a cualquier rostro que detecte.
Seguir Mano	El asistente personal, procede a seguir la mano del usuario
Seguir Juan	El asistente personal procede a seguir el rostro del Usuario “Juan”
Seguir Vladimir	El asistente personal procede a seguir el rostro del usuario “Vladimir”
Seguir Luis	El asistente personal procede a seguir el rostro del usuario “Luis”
Seguir Salome	El asistente personal procede a seguir el rostro del usuario “Salome”
Seguir Botella	El asistente personal procede a seguir el objeto correspondiente
Seguir Lata	El asistente personal procede a seguir el objeto correspondiente
Seguir Pelota	El asistente personal procede a seguir el objeto correspondiente

Cuéntame un Cuento	El asistente personal cuenta un cuento de los que posee en su base de datos.

Tabla 60.- Comandos de voz que admite el Asistente Personal

Si el comando pronunciado no se encuentra en la tabla anterior, el avatar responderá con el mensaje “no entiendo esa orden”.

5.3.2.6 Conversión Texto a voz o TTS

Otra de las partes primordiales y que le brindan más naturalidad al asistente personal, es la capacidad de expresarse a través de lenguaje. Para esto se deberá convertir en expresiones auditivas los textos que representan lo que quiere decir el asistente, empleando el paquete que Android trae incorporado por defecto. Sin embargo, también es factible acceder al paquete que incluye cada modelo de dispositivo móvil. Para nuestro caso se configuraron las opciones para hacer uso del paquete de Samsung, ya que incorpora más opciones y tipos de voz, como es el caso de una voz robótica, alternativa que el paquete de Android no incluye.

Este elemento dentro de la aplicación se integra directamente con el avatar para brindar la sensación de que el asistente personal habla. La conversión texto a voz puede ser lanzada dentro de la aplicación, ya sea en respuesta a una orden por voz, a la lectura de un cuento o leyenda, para la lectura de un recordatorio, la lectura de un correo electrónico o cuando el asistente entra en modo de autonomía.

En la Figura 170 se puede observar el diagrama de flujo dentro que explica el proceso para la conversión de texto a voz.

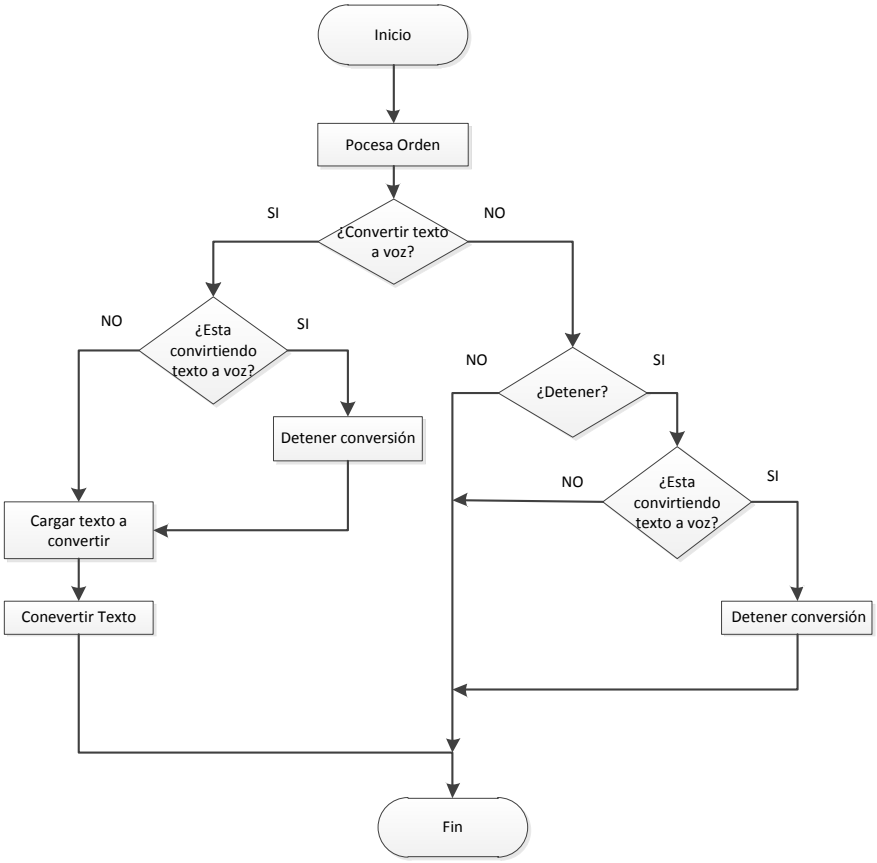


Figura 170.- diagrama de Flujo para la conversión de Texto a Voz

5.3.2.7 Módulo de comunicación

Dentro de los elementos que brindan más libertad para el control del asistente personal se encuentra la conexión al servidor de asistencia remota. Para ello, dentro de la aplicación se implementó un *socket* Java cliente que se conectará al *socket* o al puerto del servidor correspondiente. El proceso de conexión y de comunicación se maneja de manera similar que con el dispositivo Bluetooth, es decir, se tiene dos tareas asíncronas. La primera tarea se encarga de crear y conectarse al socket correspondiente (en nuestro caso empleando el puerto 6000), una vez terminada esta etapa se ejecuta la segunda tarea asíncrona que sirve como método para la escritura y lectura de la información transmitida y recibida, desde y hacia el servidor de asistencia remota.

5.3.2.8 Adquisición de Video

La adquisición de imágenes, o la captura de video es el paso inicial para todo el procesamiento de imágenes, aspecto que le brinda al asistente personal sus funciones características. Para acceder a la cámara es necesario implementar la clase ***CvCameraViewListener2*** que es provista por OpenCV. Esta clase posee los siguientes métodos:

- *onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame)*
- *onCameraViewStarted(int width, int height)*
- *onCameraViewStopped()*

Antes de acceder a estos métodos es necesario conectarse a la librería de OpenCV mediante la clase ***BaseLoaderCallback***. El diagrama de flujo llevar a cabo esta tarea dentro de nuestro proyecto sería el que se muestra en la Figura 171.

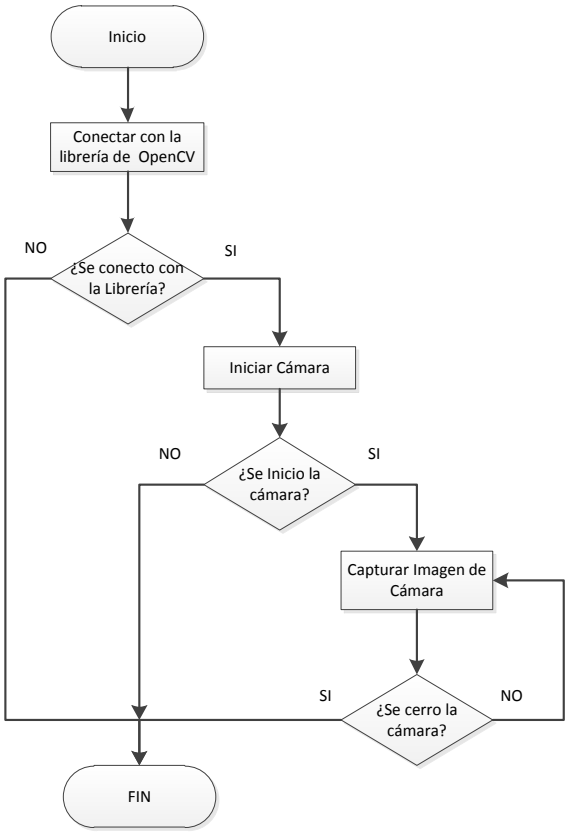


Figura 171.- Diagrama de flujo para la adquisición de Video

Una vez que se ha conectado correctamente a la librería de OpenCV y la cámara haya iniciado adecuadamente, se podrá capturar las imágenes en el método ***onCameraFrame()*** para su posterior procesamiento.

En el método ***onCameraViewStarted()*** es posible extraer la resolución de la cámara en pixeles con la que se adquirirá las imágenes.

5.3.2.9 Avatar Animado

El avatar interactivo es el elemento gráfico que interactúa con el usuario y a la vez le brinda al asistente personal un aspecto más humano. Para la creación del Avatar primero se genera una imagen *bitmap* sobre el cual se dibujarán los elementos que lo conforman. El tamaño de dicha imagen y la superficie de dibujo creada dependerán de la resolución de la pantalla que posea cada dispositivo, lo que permite que el avatar pueda ser escalado. La formación del avatar es la suma de varias imágenes, en donde la combinación correcta de cada una de ellas crea los distintos gestos o expresiones faciales que este puede realizar.

Los elementos o imágenes que se utilizan para la creación del avatar se muestran en la Tabla 61.

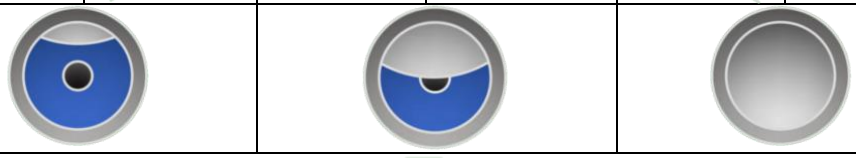
Elemento	**Imágenes**
Cejas	
Ojos	
Nariz	
Lentes	
Bocas	

Tabla 61.- Recursos Gráficos utilizados para crear el Avatar Animado

La combinación adecuada de los elementos mostrados en la tabla anterior, en conjunto con una tarea asíncrona que siempre se está ejecutando crea la animación del Avatar interactivo. El avatar puede realizar las siguientes expresiones faciales:

- Neutralidad
- Alegría
- Tristeza
- Sorpresa
- Aburrimiento

Hay determinados movimientos, como el movimiento de cejas o el parpadeo que se ejecutan de manera automática (pseudoaleatoria), lo que da más naturalidad al mismo. Además, es importante mencionar que el avatar tiene la capacidad de hablar cada vez que se requiera, ya que este se integra directamente con la conversión de texto a voz, verbigracia, cuando reciba un correo electrónico o deba realizar un recordatorio, llevará a cabo dichas tareas a través de la voz.

Las respuestas que el avatar realiza como resultado a los comandos provenientes del reconocimiento de voz se muestran en la Tabla 62.

Comando	Respuesta hablada
Saluda	"Hola mi nombre es Robodroid"
Funciones	"Soy el primer autómata basado en dispositivos móviles Android, dedicado a la ayuda en el área de discapacidad, con aplicaciones basadas en la realidad aumentada, visión artificial y reconocimiento de ordenes por voz"
Futuro	"En el futuro espero ser capaz de ayudar a niños, con discapacidad motriz, discapacidad leve y autismo, colaborando en el proceso de rehabilitación mediante actividades lúdico-interactivas, que ayuden al el desarrollo psicomotriz de niños"
Cantar	
Jugar	"Que bien..!!, .me encanta jugar"
Juego	
Pintar	"Pintar es muy divertido..!! "
Seguir mano	"Se activó el seguimiento de mano"
Seguir cara	"Se activó el seguimiento de rostros"
Seguir Juan	"Siguiendo a Juan"
Seguir Luis	"Siguiendo a Luis"
Seguir Vladimir	"Siguiendo Vladimir"
Seguir Salome	"Siguiendo Salome"
Segur Botella	"Identificando una botella"
Seguir Lata	"Identificando una lata"
Seguir Pelota	"Identificando una pelota"
Cuéntame un Cuento	

Tabla 62.- Respuestas habladas por el Avatar a cada comando de voz

En respuesta al comando de voz "Cuéntame un cuento" el avatar procede a la lectura de una leyenda o un cuento infantil. La aplicación posee almacenada cuatro cuentos infantiles

y cuatro leyendas tradicionales del Ecuador, en la Tabla 63 se pueden apreciar un listado de los mismos, así como su identificador

ID	Descripción
1	La leyenda de Cantuña
2	La leyenda de la Dama Tapada
3	La leyenda del Padre Almeida
4	La leyenda de los Cañarís
5	El Cuento de la Liebre y la tortuga
6	El cuento de Caperucita Roja
7	El cuento del Patito Feo
8	El cuento de la Cenicienta

Tabla 63.- Cuentos y Leyendas disponibles dentro de la Aplicación

El avatar también emite frases cuando entra en modo de autonomía (espera). Este modo se activa cuando el robot no detecta ningún rostro, mano u objeto dentro de la escena durante un periodo de tiempo de un minuto aproximadamente. Las tres frases que habla cuando está en modo de autonomía se muestran en la Tabla 64.

ID	Respuesta Hablada
1	"Hey..!! ¿Hay alguien aquí? necesito un poco de atención, estoy aquí "
2	"Oh..!! Estoy muy aburrido. ¿Alguien para jugar?"
3	"Ouch..!! Me siento un poco oxidado de no hacer nada"

Tabla 64.- Frases de autonomía que posee el Avatar

En la Figura 172 se puede apreciar una captura del Avatar completo.

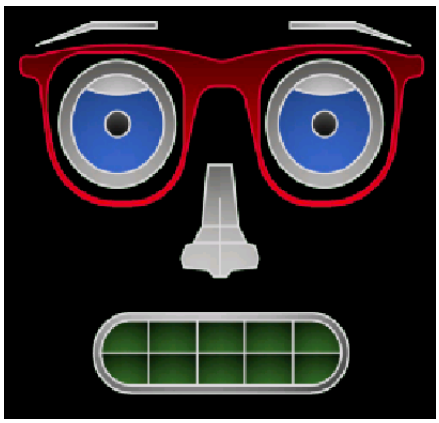


Figura 172.- Captura del Avatar Animado finalizado

5.3.2.10 Procesamiento de Imágenes

En este punto se proveerán detalles del procesamiento de imágenes y algoritmos usados para realizar el reconocimiento de gestos, objetos, la identificación y reconocimiento facial, que son la base con la cual el usuario puede interactuar con el asistente personal autónomo.

-***Reconocimiento de Gestos realizados con la Mano.***

La detección y reconocimiento de gestos realizados con la mano nace de la necesidad de tener un medio para interactuar con el asistente personal autónomo. Puesto que no se tiene ningún método de escucha continua para que el asistente personal recepte una orden por parte del usuario y considerando que el reconocimiento de ordenes por voz es un proceso que no puede estar ejecutándose siempre ya que consume muchos recursos del dispositivo y debe ser lanzado únicamente cuando sea necesario, utilizar la detección de gestos como medio de interacción principal es una alternativa válida.

Antes de empezar con la explicación del funcionamiento y los algoritmos implementados cabe mencionar que esta etapa de la aplicación fue escrita en código nativo, además se realizaron primero aplicaciones de prueba en un ordenador para luego proceder a la migración al sistema operativo Android.

El sistema diseñado es capaz de reconocer 4 gestos diferentes que se efectúen con la mano y para la realización del mismo se siguieron varias etapas.

Fase de entrenamiento: consiste en la adquisición y la creación del corpus de imágenes que se van a utilizar para extraer las características que se utilizarán posteriormente en la fase de reconocimiento. En la Figura 173 se muestra el diagrama de bloque para la fase de entrenamiento.

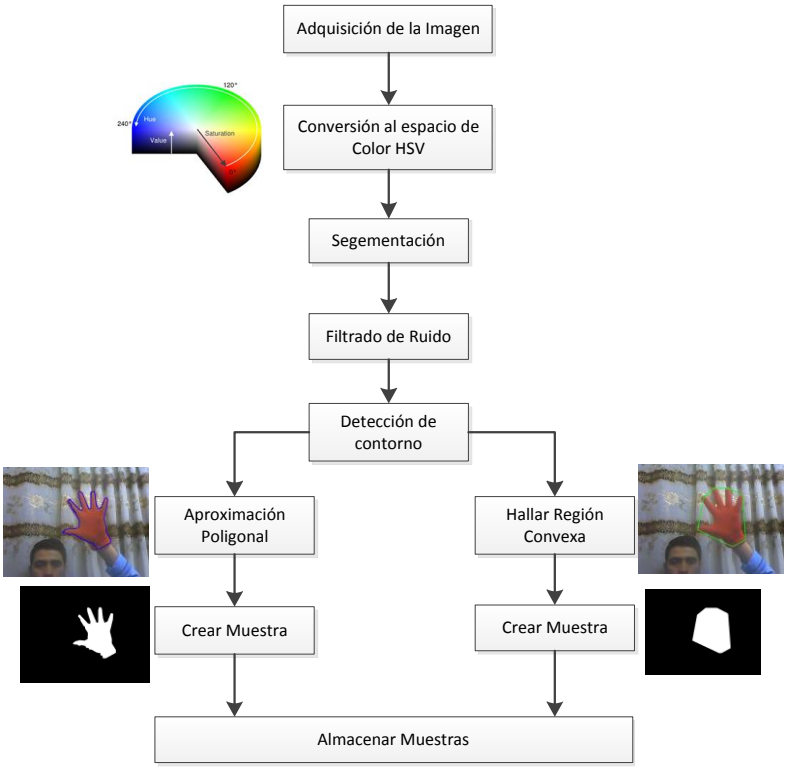


Figura 173.- Proceso para la creación de muestras para la Fase de entrenamiento

Dentro de las partes más importantes de la fase de entrenamiento podemos describir:

- Segmentación: la segmentación se basa en la sustracción fondo. Para poder efectuar este punto se seleccionó un color fácil de identificar y que no es muy común dentro de una escena.
- Filtrado de ruido: consta de la erosión y la dilatación de la imagen segmentada
- Aproximación poligonal: trata de representar los contornos mediante aproximaciones geométricas
- Región convexa: se identifican los puntos cóncavos dentro de los contornos, a la final se tiene los puntos que presentan más información sobre el contorno.

Durante la fase de entrenamiento se obtuvo un total de 200 muestras por cada mano teniendo un total de 800 muestras. En la Tabla 65 se muestran los ejemplos de las muestras adquiridas.

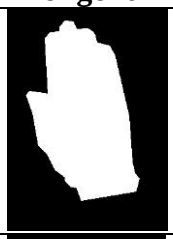
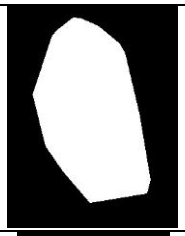
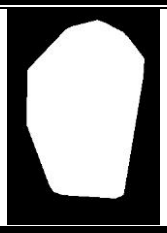
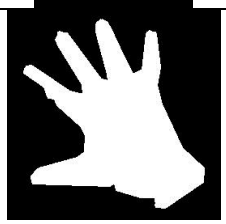
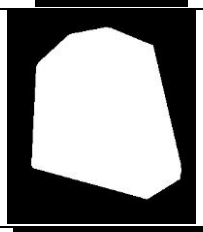
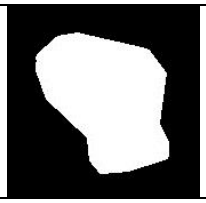
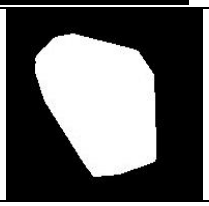
Muestra	Aproximación Poligonal	Región Convexa	Muestra Aproximación Poligonal	Muestra Región Convexa
Gesto 1				
Gesto 2				
Gesto 3				
Gesto 4				

Tabla 65.- Ejemplos de muestras tomadas para los cuatro gestos

Una vez que se poseen las muestras necesarias, se extraen las características tanto de la muestras de la aproximación poligonal, como las muestras de la región convexa. Para este caso se extrajeron como características los siete momentos invariantes de Hu, que nos brindan información acerca de la geometría de cada muestra.

Fase de corrección en la iluminación: Hay un elemento importante a considerar y que afecta en gran medida a cualquier proceso de reconocimiento o procesamiento de imágenes, que es el cambio y la variación en la iluminación que existe dentro del ambiente. Por ello, se implementaron varias técnicas a fin de corregir los cambios de iluminación que puedan surgir en distintas condiciones ambientales.

Espacio de color CIE Lab: como se vio en el Capítulo 3, este espacio de color provee una mejor aproximación de la forma cómo el ojo del ser humano captura la iluminación (frente al HSV).

CLAHE: la ecualización común del histograma no es óptima en escenas con alta iluminación, o con elementos que contrasten demasiado. Por ello, al ecualizar se perderá la mayoría de la información por el exceso de brillo. CLAHE (*Contrast Limited Adaptative Histogram Equalization*) permite hacer una ecualización adaptativa, el principio de esta ecualización se basa en dividir la imagen en una cuadrícula, por ejemplo de 8X8, para luego ecualizar y normalizar cada porción de la cuadrícula y de esta manera se evitar que el ruido en una parte de la escena afecte a toda la imagen. En la Figura 174 se muestra un paisaje en escala de grises.

Figura 174.- Paisaje en escala de grises sin aplicar ninguna ecualización

Ahora en la Figura 175 podemos apreciar la misma imagen a la que se ha aplicado la ecualización habitual frente a la ecualización mediante CLAHE.

Figura 175.- Ecualización habitual (izquierda) frente a CLAHE (derecha)

Como podemos apreciar, se puede notar una diferencia importante, ya que en el caso de la ecualizacion con CLAHE se observa una mejor definición de la imagen, es decir, no se ha perdido información y se ha afectado a la imagen por la iluminacion del fondo.

Sin embargo, se debe mencionar que en condiciones extremas no será posible corregir la imagen utilizando el método anterior, por lo que se plantea una solución adicional. Consideremos que se corrigió la iluminación en la escena lo máximo que se pudo, pero aun así la iluminación existente sigue afectando a la forma como la cámara percibe el color del objeto que se quiere segmentar. En nuestro caso recordemos que estamos usando el color rojo para hacer la segmentación y sustracción del fondo, entonces hay casos que la cámara no percibe el color rojo como tal, sino percibe un color más cálido o más oscuro, basándonos en eso la idea sería que en base a la iluminación que percibe la cámara se pueda ajustar el color que se segmentará.

En base al concepto anterior se calcula el histograma del canal "L", que es el canal de iluminación dentro del espacio de color CIE Lab. De esta manera tendremos el histograma de ejemplo que se muestra en la Figura 176, donde "I" es la intensidad del pixel y "N" es el número de pixeles que existen para cada intensidad de pixel.

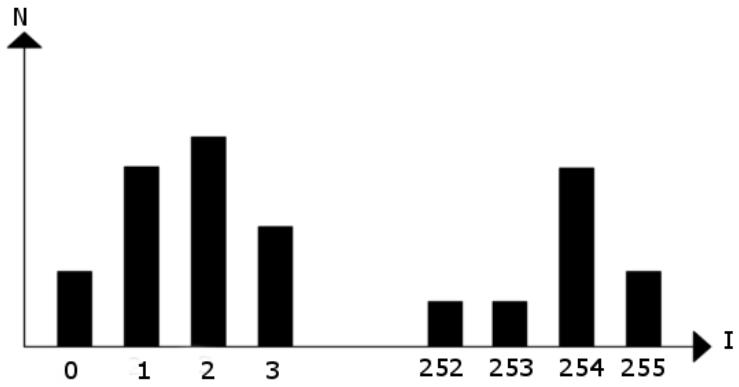


Figura 176.- Histograma de Ejemplo

Habitualmente dentro de una imagen se pueden tener objetos o elementos de color negro o blanco que afectan los valores del histograma, por ejemplo, si existiese un objeto oscuro dentro de la imagen el histograma nos diría que la escena está poco iluminada, cuando en realidad no lo está, y sucederá lo contrario cuando tengamos muchos pixeles con color blanco. De esta forma, para tratar de evitar este inconveniente dividimos al histograma global en dos sub histogramas y asignamos un peso a cada intensidad de pixel, otorgándole un mayor peso a los pixeles con intensidad intermedia, por tanto los pixeles que se

encuentren en los extremos, ya sea del blanco o del negro tendrán un peso mucho menor. En la Figura 177 se aprecia mejor dicho proceso.

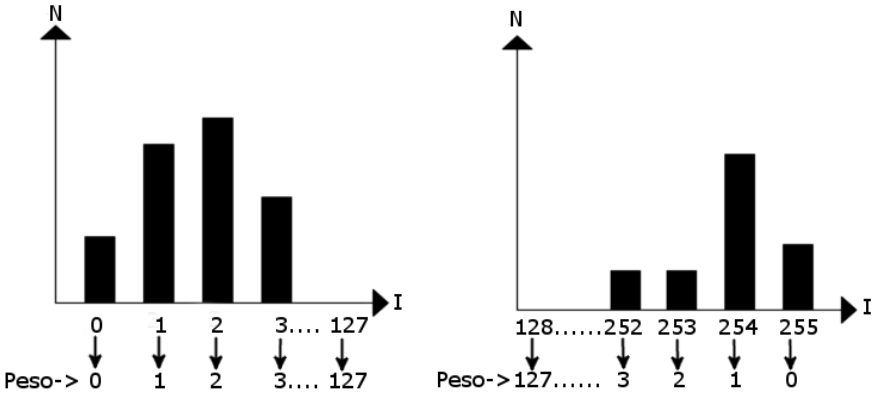


Figura 177.- División del histograma y asignación de pesos a cada Intensidad de Pixel

Ahora para cada histograma se crea un vector resultado de la multiplicación del número de pixeles para cada intensidad del histograma, por el peso previamente asignado. De esta manera tendremos que:

$$Vn = \{0xN_0, 1xN_1, \dots \dots . 127xN_{127}\}$$

$$Vb = \{127xN_{128}, 126xN_{129}, \dots \dots . 0xN_{255}\}$$

En donde N_i es número de pixeles en la intensidad indicada, Vn es el vector de valores para el histograma con las intensidades que tienden a negro, y Vb el vector de valores con las intensidades que tienden al blanco. En este punto es posible calcular las medias de Vn asi como de Vn, de tal manera tendremos:

$$mn = \frac{1}{128}\sum_{i=1}^{128} Vn_i$$

$$mb = \frac{1}{128}\sum_{i=1}^{128} Vb_i$$

Con las dos medias se puede extraer una relación entre la mitad izquierda, con la mitad derecha del histograma global, para ello tendremos un factor f que estará definido como:

$$f = \frac{mn}{mb}$$

Este factor nos da una perspectiva global de como fluctúa todo el histograma en distintas condiciones de iluminación, sin importar que existan dentro elementos totalmente negros o blancos, que no significa que se tenga mala o buena iluminación, respectivamente. El factor

f, variará tendiendo hacia cero si nuestro histograma contiene mayormente valores cercanos al blanco, o tenderá hacia el infinito si contiene valores cercanos al negro.

Ahora considérese que para la segmentación por color lo hacemos variando los valores que se toman de los canales correspondientes en el espacio de color CIE Lab. Dentro de OpenCV los canales de CIE Lab tienen los siguientes Rangos:

- Canal "L": de 0-255
- Canal "a": de 0-255
- Canal "b": de 0-255

Para nuestro caso, como se está segmentando el color rojo, simplemente se ajusta el canal "a". De esta manera se tomaron muestras del factor f y el ajuste en unidades que se debería realizar en el canal "a" en distintas condiciones de iluminación, a fin de determinar cuál es el valor necesario para identificar apropiadamente el color requerido. En la Tabla 66 se muestran las muestras tomadas.

Valor de f	**Ajuste en unidades que se debe aplicar en el canal "b"**
0,5	40
0,7	35
1	30
2,4	22
4,8	20
6,2	15
8,1	10
10	6

Tabla 66.-Muestras tomadas del factor f y el ajuste requerido en el canal correspondiente

Con las muestras de la Tabla 66 se puede estimar una línea de tendencia con la ayuda de software de computadora tal como se muestra en la Figura 178.

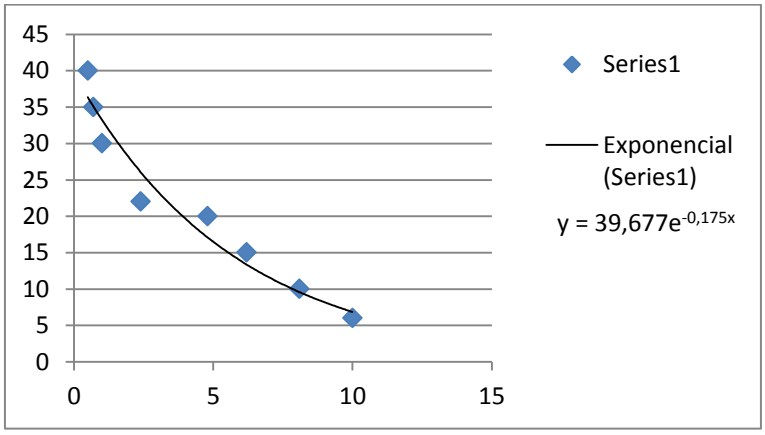


Figura 178.- Estimación de línea de tendencia para el ajuste de iluminacion

Por tanto la ecuación de ajuste quedaría definida como:

$$ajuste = 39{,}677e^{-0{,}175f}$$

Finalmente, se debe considerar que dentro de la aplicación el ajuste calculado previamente se sumará al límite inferior que el usuario defina en el canal correspondiente, permitiendo así tener un mayor control sobre la calibración óptima en distintas condiciones de iluminación.

Fase de reconocimiento: una vez que se tienen las muestras de la fase de entrenamiento, y se ha realizado las correcciones de iluminación, se puede llevar a cabo el reconocimiento de gestos. Para ello se efectúa una comparación de las características de un nuevo gesto (capturado por la cámara) con las muestras de entrenamiento a fin de identificar qué gesto se está realizando. El diagrama de bloques para esta etapa se muestra en la Figura 179.

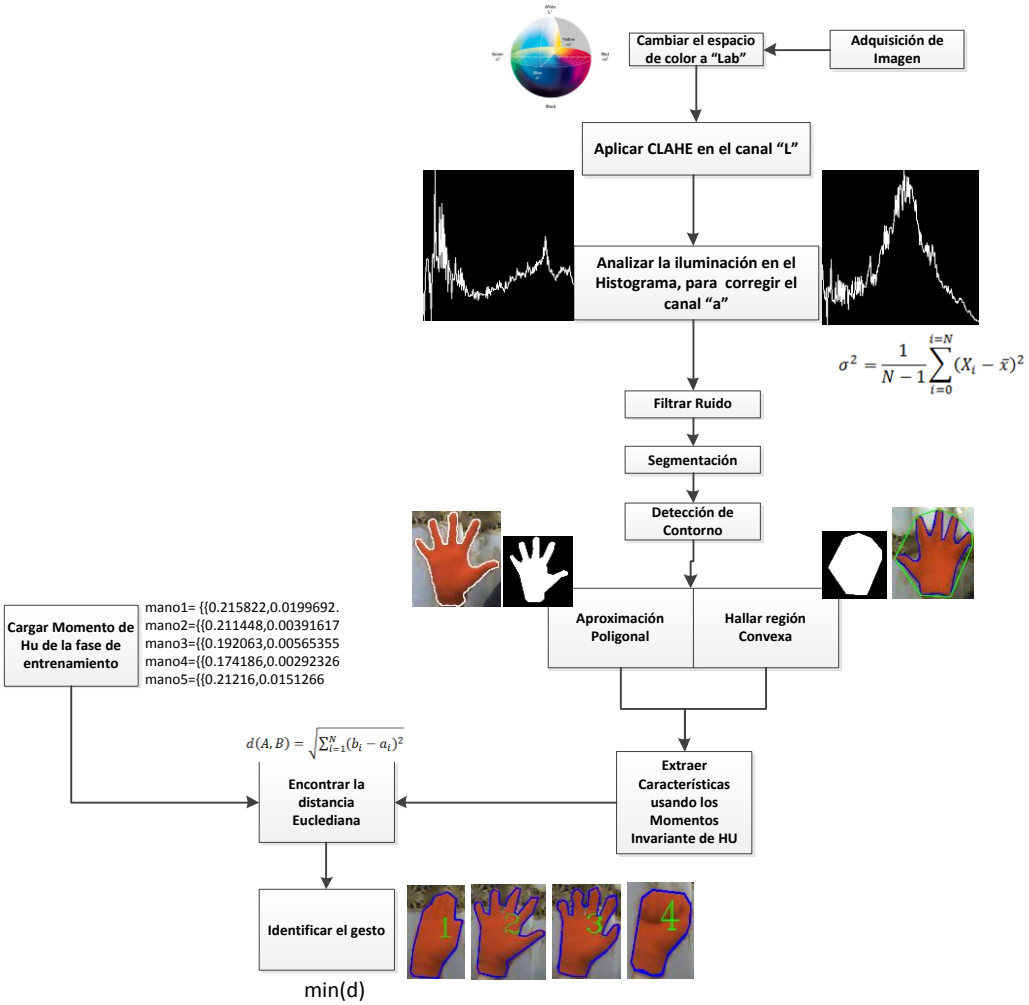


Figura 179.- Diagrama para el proceso de reconocimiento de gestos

Después de las etapas de corrección en la iluminación las etapas son similares a la fase de entrenamiento, en cuyo caso ya no se harán mención de las mismas. En este punto se hará una explicación más detallada de la forma en que se lleva a cabo el proceso para el reconocimiento de gestos.

Consideremos que para el nuevo gesto que se captura de la cámara, tenemos dos tipos de Momentos invariantes Hu:

- Momentos extraídos de la aproximación poligonal: $mAP[7]$
- Momentos extraídos de la región convexa: $mRC[7]$

De igual manera, tendremos dos tipos de momentos extraídos de las muestras de entrenamiento. Un paso para la identificación del gesto correcto es encontrar la distancia Euclidiana y dado que para cada gesto tenemos un total de 200 muestras, 100 de la aproximación poligonal y 100 de la región convexa, entonces tendremos un total de 8 matrices que contienen los momentos de Hu tanto de la aproximación poligonal como para la región convexa para los 4 gestos que queremos identificar:

- $gAP_1[7][100]$: momentos de la aproximación poligonal para el gesto 1
- $gAP_2[7][100]$:momentos de la aproximación poligonal para el gesto 2
- $gAP_3[7][100]$: momentos: de la aproximación poligonal para el gesto 3
- $gAP_4[7][100]$: momentos de la aproximación poligonal para el gesto 4
- $gRC_1[7][100]$: momentos de la región convexa para el gesto 1
- $gRC_2[7][100]$: momentos de la región convexa para el gesto 2
- $gRC_3[7][100]$: momentos de la región convexa para el gesto 3
- $gRC_4[7][100]$: momentos de la región convexa para el gesto 4

Ahora las distancias Euclidianas quedarían expresadas de la siguiente forma:

$$dAP_i[x] = \sqrt{\sum_{j=1}^{7}(mAP[j] - gAP_i[j][x])^2}$$

$$dRC_i[x] = \sqrt{\sum_{j=1}^{7}(mRC[j] - gRC_i[j][x])^2}$$

$$x \in \{1 \rightarrow 100\}$$

En donde dAP_i, es la matriz de distancias para la aproximación poligonal para cada gesto i, y dRC_i es la matriz de distancias para la región convexa para cada gesto i. Es obvio que para cada gesto tendremos una cantidad de 100 distancias mínimas, entonces ahora para cada gesto es necesario extraer la menor distancia de cada matriz de valores.

$$dmAP_i = \min(dAP_i)$$

$$dmRC_i = \min(dRC_i)$$

En donde $dmAP_i$, es la distancia mínima de la aproximación poligonal para el gesto i, y $dmRC_i$, es la distancia mínima de la región convexa para el gesto i.

Ahora bien, tenemos que determinar qué vector de distancias menores proporciona un mejor resultado. En base a las pruebas efectuadas se obtuvo que utilizando cada una por separado la precisión fue baja y si se combinaba la información de los dos vectores dicho valor se incrementa. Asignándole a cada una un peso, el porcentaje de reconocimiento se mejora, de tal manera tendríamos que nuestro arreglo de resultados res para cada gesto estará dado como:

$$res_i = p_1 * dmAP_i + p_2 dmRC_i$$

En donde p_1 es el peso para la distancia mínima de la aproximación poligonal, y p_2 es el peso para la distancia mínima de la región convexa. Según las pruebas realizadas el valor óptimo para $p_1 = 0.6$ y para $p_2 = 0.4$. de esta forma, nuestro vector de resultado estaría dado como:

$$res = \{p_1 * dmAP_1 + p_2 dmRC_1 \, , p_1 * dmAP_2 + p_2 dmRC_2 \; ; \; p_1 * dmAP_3 + p_2 dmRC_3 ; \; p_1 * dmAP_4 + p_2 dmRC_4\}$$

En este punto se puede extraer el mínimo y posición de nuestro arreglo de resultados para determinar a qué gesto corresponde. Si el resultado del menor no está dentro del rango de tolerancia se considera que no es una mano.

$$si \begin{cases} \min(res) < tolerancia; gesto = posMenor(res) \\ min(res) > tolerancia; gesto = 0 \end{cases}$$

Para calcular la tolerancia se tomaron muestras de la distancia mínima de 50 gestos, en la Figura 180 se muestra una fracción de las muestras

x={0.0130737,0.0128773,0.0183659,0.0165042,0.0100771,0.00813……

Figura 180.- Fracción de las muestras tomadas del reconocimiento de gestos

La media de las muestras es:

$$media = \frac{1}{50} \sum_{i=1}^{50} x_i = 0{,}0061265$$

La desviación estándar estaría dada como:

$$\sigma = \sqrt[2]{\frac{1}{49}\sum_{i=1}^{50}(x_i)^2} = 0.0041599$$

La tolerancia estará definida como la media más una desviación estándar:

$$tolerancia = \sigma + media = 0.010286$$

En la Figura 181 se puede apreciar el proceso de reconocimiento de los cuatro gestos.

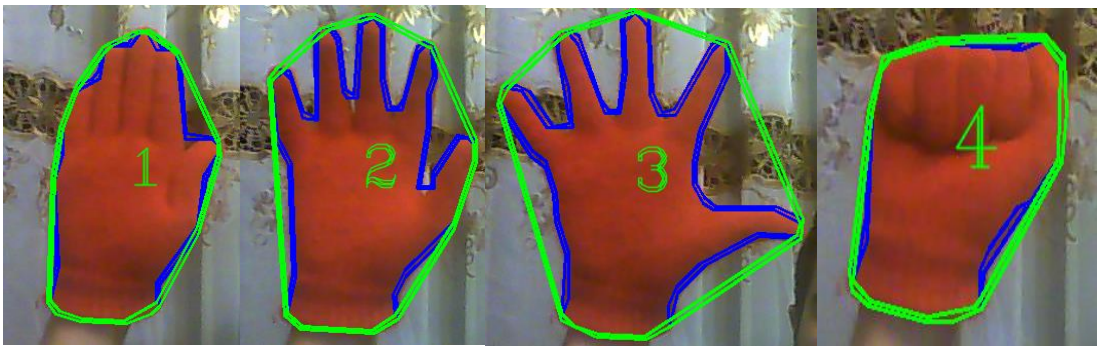


Figura 181.- Capturas del proceso de Reconocimiento de Gestos

Por último, es fundamental mencionar que el reconocimiento de gestos sirve como medio para interactuar con el avatar y los juegos interactivos basados en la realidad aumentada. Al realizar el gesto 1 seguido del gesto 4, se lanza el reconocimiento de ordenes por voz.

-Reconocimiento de Objetos

El asistente personal autónomo posee también de la capacidad de reconocer e identificar 3 objetos, basándose en los mismos principios que se utilizó durante la etapa de reconocimiento de gestos. Una diferencia que se puede recalcar es que el color usado para el reconocimiento de objetos fue el amarillo.

Fase de entrenamiento: para la creación de las muestras de entrenamiento se utilizó el mismo proceso que para el reconocimiento de gestos, por lo que no es necesario abundar más en este punto. En la Tabla 67 se representan las muestras extraídas para el reconocimiento de objetos

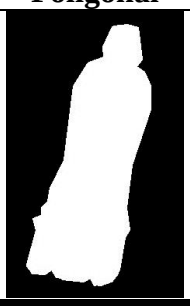
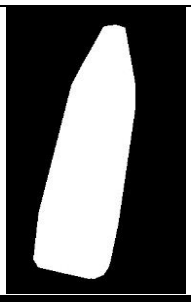
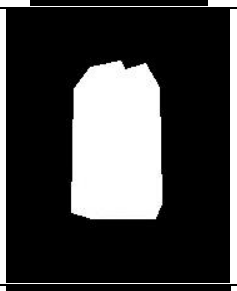
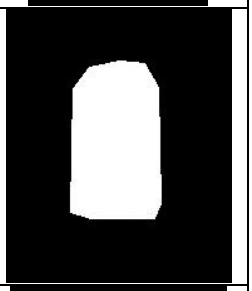
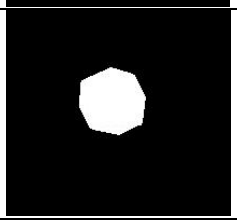
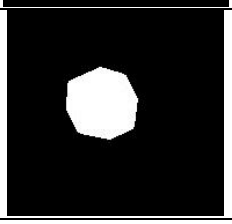
Muestra	Aproximación Poligonal	Región Convexa	Muestra Aproximación Poligonal	Muestra Región Convexa
Botella				
Lata				
Pelota				

Tabla 67.- Muestras tomadas para la fase de entrenamiento para el Reconocimiento de Objetos

Para este caso se capturó un número de 50 muestras de la aproximación poligonal y 50 muestras de la región convexa, dando un total de 100 para cada objeto.

Fase de reconocimiento: de manera similar que para el reconocimiento de gestos, se evaluó si la combinación de la aproximación poligonal y la región convexa mejoraban la precisión y se determinó que era suficiente el uso de la aproximación poligonal. El diagrama de bloques de este proceso se muestra en la Figura 182.

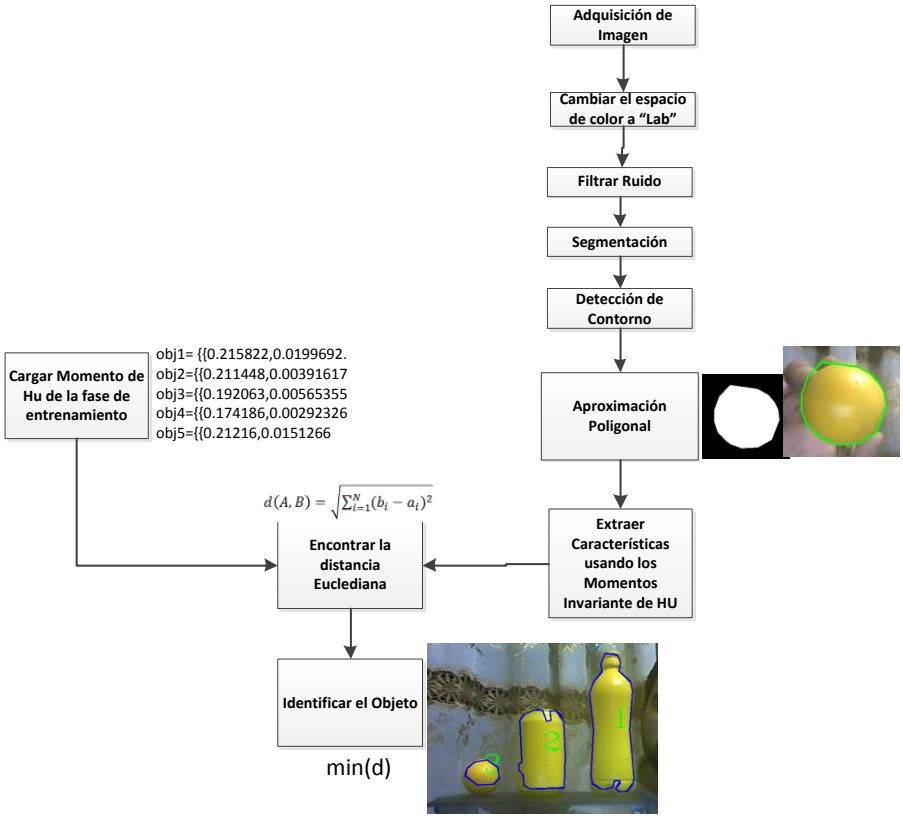


Figura 182.- Diagrama usado para el Reconocimiento de Objetos

La identificación del objeto se realiza de manera similar que con el reconocimiento de gestos, analizando la mínima distancia Euclidiana, con la única diferencia que solo se consideran los momentos invariantes de Hu de la aproximación poligonal.

En la Figura 183 se puede apreciar con mayor claridad el reconocimiento de objetos.

Figura 183.- Captura del proceso de reconocimiento de Objetos

-Reconocimiento e identificación Facial

Para esta etapa del proceso se usaron métodos que ya llevan algún tiempo a disposición de los usuarios a través de OpenCV. El diagrama de flujo para la identificación y reconocimiento de rostros se muestra en la Figura 184.

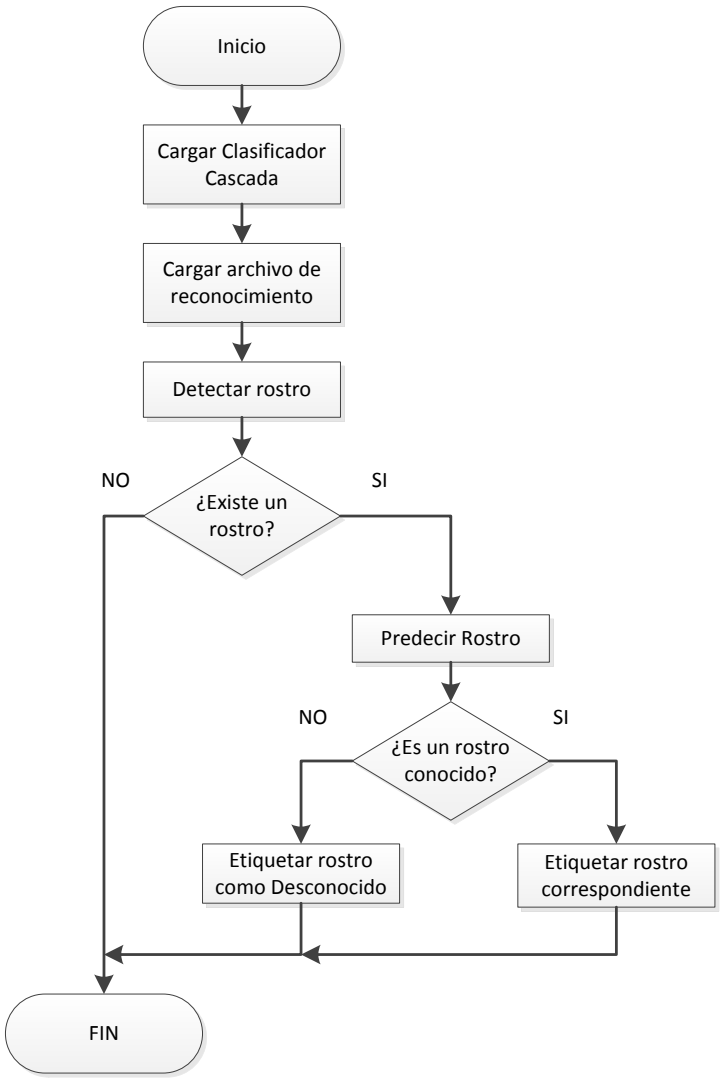


Figura 184.- Diagrama de flojo para la identificación y reconocimiento Facial

A fin de detectar rostros, se ha empleado el clasificador basado en Patrones Binarios Locales (Local Binary Patterns, LBP). Una vez que el sistema detecta un rostro, continua hacia la etapa de reconocimiento, en donde predice el rostro del sujeto identificado. Actualmente el sistema es capaz de reconocer cuatro rostros gracias al entrenamiento realizado usando el algoritmo FisherFaces.

El corpus de imágenes de sujetos para el entrenamiento consta de un total de 80 imágenes (20 imágenes por sujeto). En la Tabla 68 se muestra parte del corpus de entrenamiento

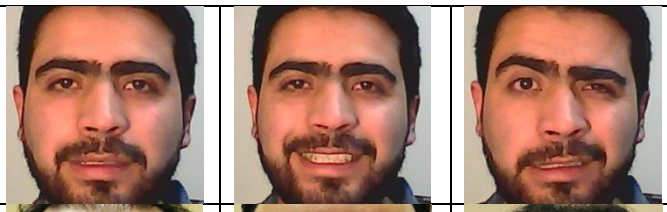
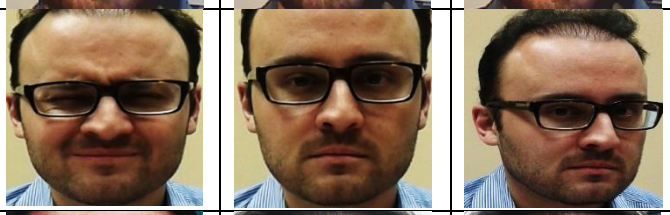
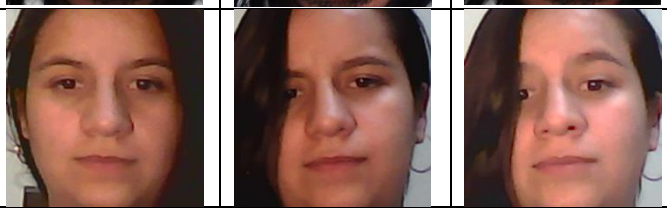
Persona	Muestras
1	
2	
3	
4	

Tabla 68.- Fracción del Corpus usado para el Reconocimiento Facial

Las muestras fueron capturadas todas con una misma resolución de 300x300 pixeles, con iluminación constante dentro de un ambiente controlado. Cabe mencionar que el filtro creado mediante el algoritmo FisherFaces no es muy robusto a cambios de iluminación, existe otro método basado en el algoritmo LBP pero el tiempo de procesamiento que este requiere no es óptimo para ser aplicado en el dispositivo móvil.

La información de los sujetos utilizados para el entrenamiento se muestra en la Tabla 69 que se presentan a continuación.

Persona	Nombre	Ocupación
1	Juan Ochoa	Colaborador/Investigación
2	Vladimir Robles	Docente/Investigación
3	Luis González	Colaborador/Investigación
4	Salome Ochoa	Estudiante

Tabla 69.- Sujetos usados para el entrenamiendo del Reconocimiento Facial

En la Figura 185 se muestra un ejemplo del reconocimiento de rostros funcionando.

Figura 185.- Captura del proceso de Reconocimiento facial en acción

5.3.2.11 Juegos Interactivos

Todos los juegos que se han diseñado emplean un concepto similar a la realidad aumentada. En tal virtud, se puede interactuar con ellos a través del reconocimiento de gestos que se detalló en secciones anteriores. Los juegos principalmente están orientados para niños, a fin de colaborar en el proceso de desarrollo psicomotriz.

-Juego 1: asociación de figuras geométricas básicas.

El primer juego está compuesto de elementos geométricos y su correspondiente asociación (ver Tabla 70).

Recursos Gráficos			
			
			

Tabla 70.- Elementos Gráficos necesarios para el Juego numero 1

La manera en la que se interactúa con este juego es mediante el movimiento de la mano. Con esto, en base a los gestos que realice el usuario se podrá jugar con él, desplazándose a través de la pantalla. En la Tabla 71 se muestra como se representan los gestos dentro del juego

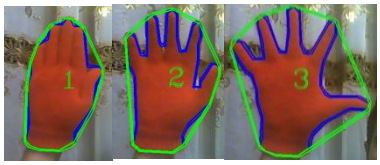
Gesto	Equivalencia en el Juego
	
	

Tabla 71.- Representación de los gestos de la mano dentro del juego Interactivo numero 1

El funcionamiento del juego es relativamente simple, ya que se debe asociar una ficha que representa un cuadrado, círculo, triángulo o una estrella, a su espacio correspondiente. A continuación detallamos algunos aspectos de interés del juego:

- El objetivo es capturar la ficha a través de los gestos que se realizan con la mano, de manera similar a como se lo hiciera en la realidad, con el fin de llevarlos al agujero correspondiente y hacer que encajen.
- Una vez que encajan automáticamente se genera otra ficha de manera aleatoria.
- En el caso de colocar la ficha en el agujero erróneo, la ficha se mantendrá en su sitio hasta que sea colocada en el que corresponde.
- Se puede mover o soltar la ficha en cualquier lugar de la pantalla.
- Para salir del juego se debe realizar el gesto número 4 sobre el icono de “salir”.

Las capturas del juego en funcionamiento se muestran en la Figura 186.

Figura 186.- Capturas del Juego Numero 1 en funcionamiento

-Juego 2: pintando con la mano.

Este juego al igual que el anterior se basa en realidad aumentada y se interactúa con él mediante los gestos realizados con la mano. En este juego existe una pantalla que contiene 3 elementos de color que permitirán dibujar figuras a mano alzada sobre un lienzo virtual (ver Tabla 72).

Recursos Gráficos		
		

Tabla 72.- Recursos gráficos necesarios para el Juego Numero 2

La equivalencia de los gestos dentro del juego se muestra en la Tabla 73.

Gesto	Equivalencia en el Juego
	
	

Tabla 73.- Representación de los gestos de la mano dentro del Juego Interactivo numero 2

El funcionamiento del juego es el siguiente:

- El objetivo es que con el gesto 1 o 2 se realice un movimiento a través de la pantalla, y a medida que se dé el mismo, se vaya pintando sobre un lienzo en blanco.
- Mientras se realice el gesto número 3 no se pintará nada sobre el lienzo.
- Cada vez que se realice el gesto 4 sobre los cuadros de color azul, rojo o verde, el color del dibujo cambiará acorde con el color elegido.
- Para salir del juego se debe realizar el gesto número 4 sobre el icono "salir".

Las capturas del juego en funcionamiento se muestran en la Figura 187.

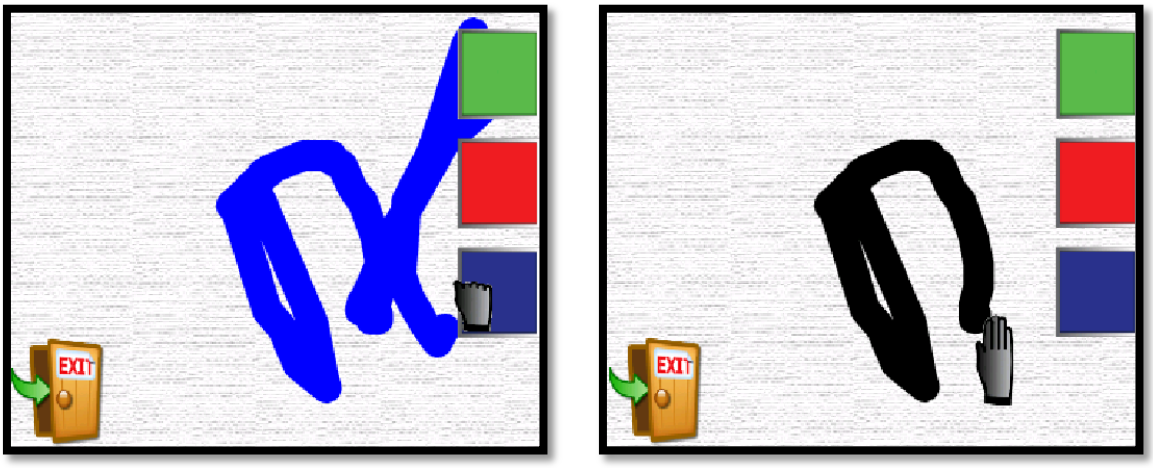


Figura 187.- Capturas del Juego Numero 2 en funcionamiento

-Juego 3: jugando con colores.

El último juego, no utiliza realidad aumentada y busca ayudar a elevar el nivel de atención de los niños. El juego se basa en que se cada cierto tiempo se mostrarán círculos de color rojo, después de un tiempo uno azul y después de otro periodo de tiempo uno amarillo. Este proceso se repetirá nueve veces y en cada ciclo la velocidad con la que aparece cada círculo de distinto color se irá incrementando.

La idea se basa en pedir que el niño realice actividades cuando ve un círculo de un color determinado, por ejemplo, cuando observe el círculo de color rojo levante las manos, con el círculo de color azul baje las manos y que con el de color amarillo aplauda.

Este juego es el único que necesita la supervisión de un adulto.

Las capturas del juego en funcionamiento se muestran en la Figura 188.

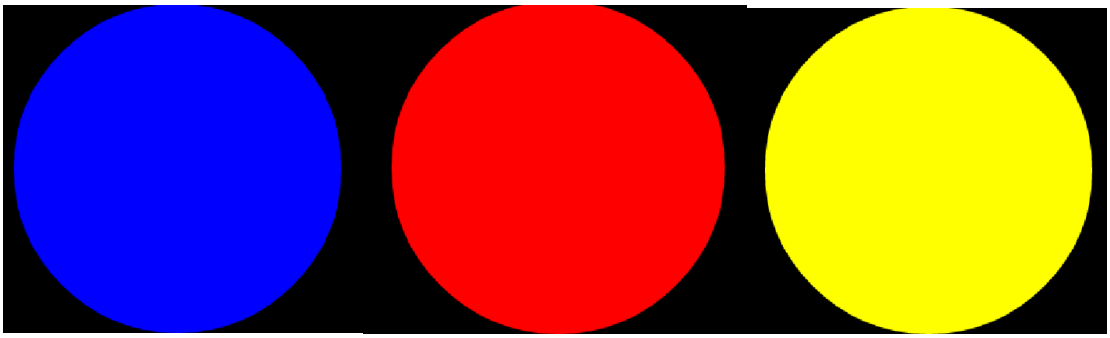


Figura 188.- Capturas del Juego Numero 3 en funcionamiento

5.3.2.12 Distribución elementos gráficos

En este punto se describirá la función y distribución de los componentes gráficos que forman parte de la aplicación. Los distintos elementos de la aplicación se encuentran distribuidos a lo largo de cuatro pestañas, aspecto que permite tener un orden lógico, así como un mejor manejo de la misma. Las pestañas disponibles son:

Pestaña "Inicio": se muestra cuando el usuario lanza por primera vez la aplicación, en dicha pestaña se encuentran los elementos necesarios para gestionar los parámetros de conexión con la plataforma y el servidor de asistencia remota, así como los parámetros necesarios para la lectura del correo electrónico. En la Figura 189 se muestra esta pestaña

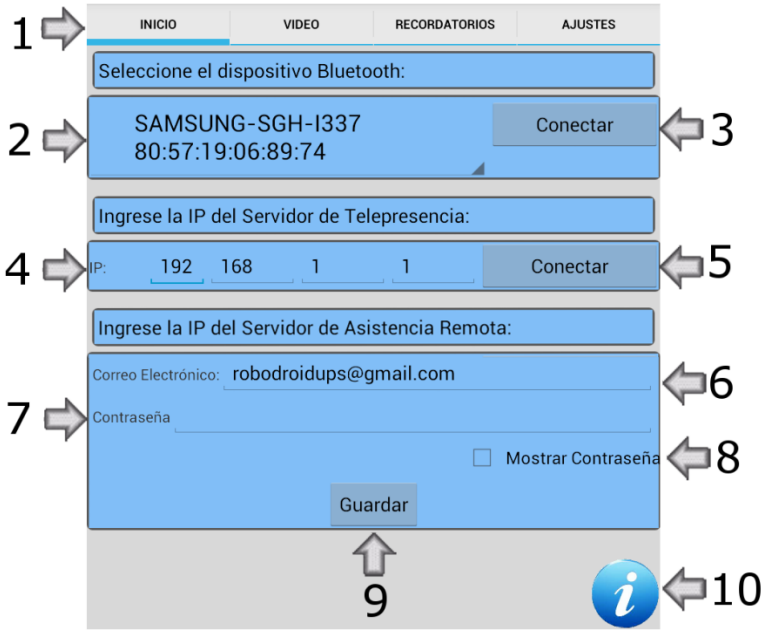


Figura 189.- Captura de la pestaña "INICIO"

La descripción de cada uno de los elementos para esta pestaña se encuentra en la Tabla 74.

Elemento	Tipo	Descripción
1	TabWidget	Contenedor de pestañas. Permite seleccionar entre las cuatro pestañas disponibles "INICIO", "VIDEO", "RECORDATORIOS" y "AJUSTES".
2	Spinner	Selector en donde se listan los dispositivos Bluetooth vinculados, también permite la selección de los mismos.
3	Button	Permite la conexión o desconexión al dispositivo Bluetooth seleccionado
4	EditText	Campos de texto numéricos para el ingreso de la IP del Servidor de Asistencia Remota
5	Button	Botón que permite la conexión o desconexión del Servidor de Asistencia Remota
6	EditText	Campo de texto para el ingreso del Correo Electrónico del Usuario
7	EditText	Campo de texto para el ingreso de la contraseña del Correo electrónico de usuario
8	CheckBox	Muestra u oculta la contraseña ingresada en el campo de texto correspondiente
9	Button	Almacena los datos de Correo electrónico y Contraseña ingresados por el usuario en la base de datos.
10	Button	Botón que muestra información acerca de la aplicación

Tabla 74.- Tabla Descriptiva de la pestaña "INICIO"

Pestaña "Video": esta pestaña posee una superficie de dibujo provista por OpenCV que permite visualizar la captura de video que realiza la cámara, además en esta aplicación se utiliza también para mostrar el avatar, el procesamiento de imágenes y los distintos juegos. Por defecto cuando la aplicación es lanzada muestra el Avatar animado. En la Figura 190 se muestra una captura de esta pestaña

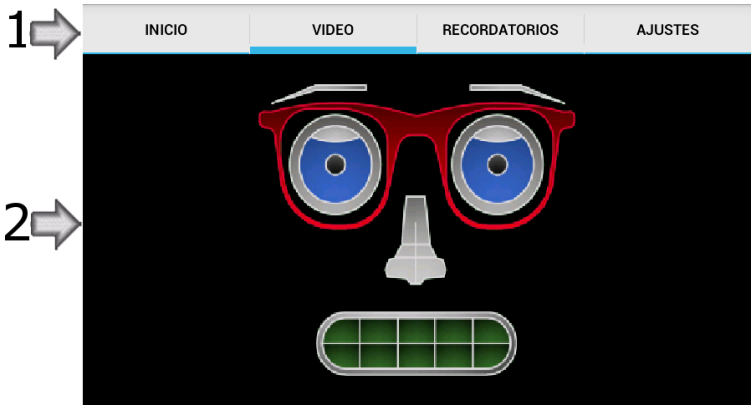


Figura 190.- Captura de la pestaña "VIDEO"

En la Tabla 75 se muestra una descripción de los elementos de esta pestaña.

Elemento	Tipo	Descripción
1	TabWidget	Contenedor de pestañas. Permite seleccionar entre las cuatro pestañas disponibles "INICIO", "VIDEO", "RECORDATORIOS" y "AJUSTES".
2	JavaCameraView	Superficie de dibujo provista por OpenCV, usada para mostrar el avatar animado, el video de la cámara, el procesamiento de imágenes y los juegos interactivos.

Tabla 75.- Tabla Descriptiva de la pestaña "VIDEO"

Pestaña "Recordatorios": permite almacenar en la base de datos de la aplicación los recordatorios que el usuario ingrese y que a su vez serán leídos por el avatar animado en la fecha y hora indicadas. En la Figura 191 se muestra una captura de esta pestaña.

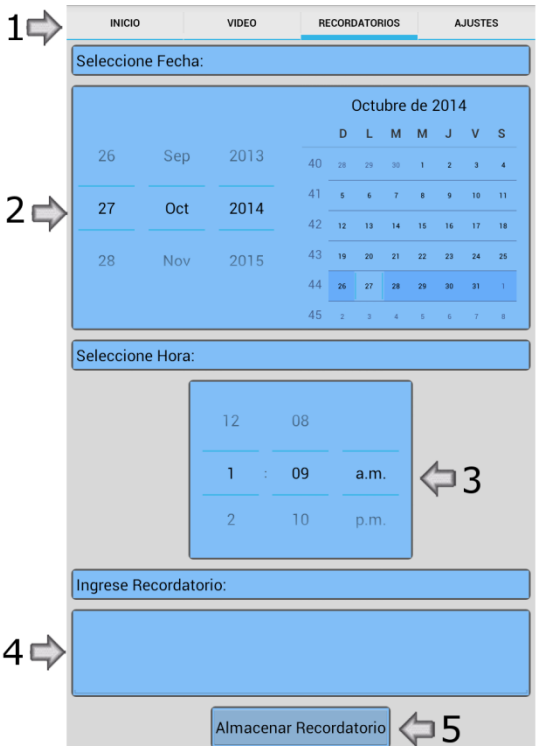


Figura 191.- Captura de la Pestaña "RECORDATORIOS"

En la Tabla 76 se describen los elementos que se contiene en esta pestaña.

Elemento	Tipo	Descripción
1	TabWidget	Contenedor de pestañas. Permite seleccionar entre las cuatro pestañas disponibles “INICIO”, “VIDEO”, “RECORDATORIOS” y “AJUSTES”.
2	DatePicker	Permite seleccionar la fecha en la que se requiere la lectura de un recordatorio.
3	TimePicker	Permite seleccionar la hora en la que se requiere la lectura de un recordatorio.
4	EditText	Campo de texto para el ingreso del recordatorio
5	Button	Almacena en la base de datos, la fecha, hora y recordatorio ingresados por el usuario.

Tabla 76.- Tabla Descriptiva de la pestaña "RECORDATORIOS"

Pestaña “Ajustes”: contiene los elementos necesarios para la configuración de los parámetros principales de la aplicación. Brinda un poco más de libertad al usuario, cuando la aplicación no dispone de conexión a internet y no es posible ejecutar el reconocimiento de órdenes por voz, además permite realizar los ajustes necesarios para que el procesamiento de imágenes se efectúe de manera adecuada en distintas condiciones de iluminación. En la Figura 192 se muestra una captura de esta pestaña.

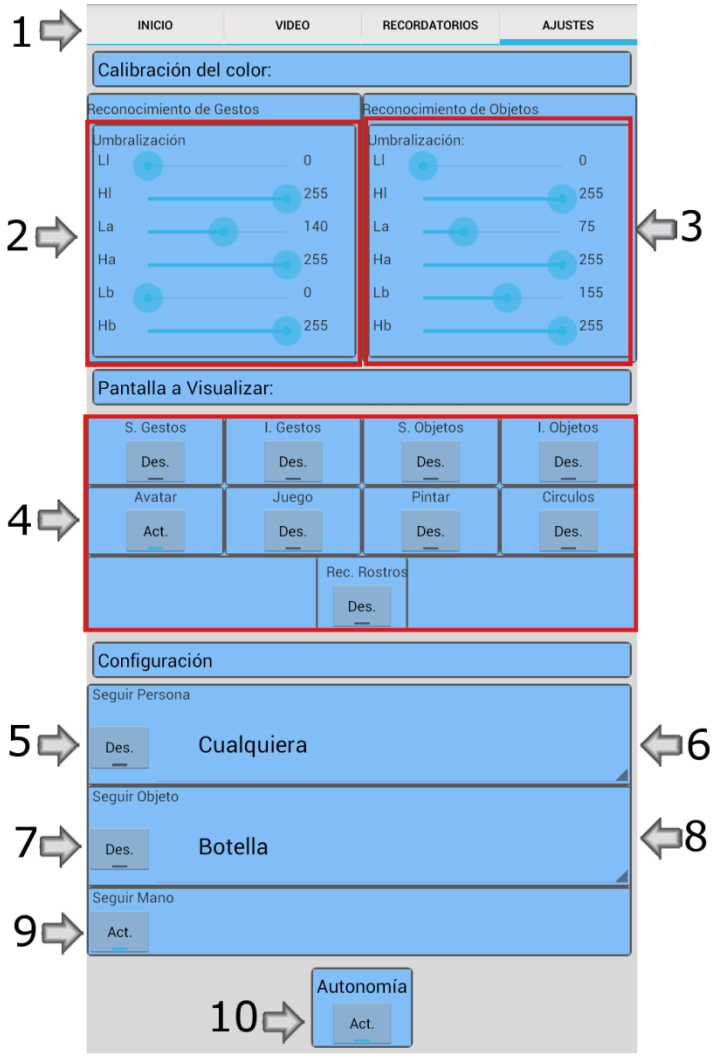


Figura 192.- Captura de la Pestaña "AJUSTES"

En la Tabla 77 que se describe cada componente para esta pestaña.

Elemento	Subelemento	Tipo	Descripción
1		TabWidget	
2	Ll	SeekBar	Permite seleccionar el valor mínimo del canal "L" para la segmentación por color usada para el reconocimiento de gestos.
	Hl	SeekBar	Permite seleccionar el valor máximo del canal "L" para la segmentación por color usada para el reconocimiento de gestos.
	La	SeekBar	Permite seleccionar el valor mínimo del canal "a" para la segmentación por color usada para el reconocimiento de gestos.
	Ha	SeekBar	Permite seleccionar el valor máximo del canal

			"a" para la segmentación por color usada para el reconocimiento de gestos.
	Lb	SeekBar	Permite seleccionar el valor mínimo del canal "b" para la segmentación por color usada para el reconocimiento de gestos.
	Hb	SeekBar	Permite seleccionar el valor máximo del canal "b" para la segmentación por color usada para el reconocimiento de gestos.
3	Ll	SeekBar	Permite seleccionar el valor mínimo del canal "L" para la segmentación por color usada para el reconocimiento de objetos.
	Hl	SeekBar	Permite seleccionar el valor máximo del canal "L" para la segmentación por color usada para el reconocimiento de objetos.
	La	SeekBar	Permite seleccionar el valor mínimo del canal "a" para la segmentación por color usada para el reconocimiento de objetos.
	Ha	SeekBar	Permite seleccionar el valor máximo del canal "a" para la segmentación por color usada para el reconocimiento de objetos.
	Lb	SeekBar	Permite seleccionar el valor mínimo del canal "b" para la segmentación por color usada para el reconocimiento de objetos.
	Hb	SeekBar	Permite seleccionar el valor máximo del canal "b" para la segmentación por color usada para el reconocimiento de objetos.
4	S. Gestos	ToogleButton	Cuando está activado, muestra en la superficie de dibujo de la pestaña "Video", el proceso de segmentación por color que se realiza para el reconocimiento de gestos.
	I. Gestos	ToogleButton	Cuando está activo muestra en la superficie de dibujo de la pestaña "Video", el proceso de reconocimiento de gestos.
	S. Objetos	ToogleButton	Cuando está activado, muestra en la superficie de dibujo de la pestaña "Video", el proceso de segmentación por color que se realiza para el reconocimiento de objetos.
	I. Objetos	ToogleButton	Cuando está activado, muestra en la superficie de dibujo de la pestaña "Video", el proceso de reconocimiento de objetos.
	Avatar	ToogleButton	Cuando está activado, muestra en la superficie de dibujo de la pestaña "Video", el Avatar animado.
	Juego	ToogleButton	Cuando está activado, muestra en la superficie de dibujo de la pestaña "Video", el juego 1.
	Pintar	ToogleButton	Cuando está activado, muestra en la superficie

			de dibujo de la pestaña “Video”, el juego 2.
	Círculos	ToogleButton	Cuando está activado, muestra en la superficie de dibujo de la pestaña “Video”, el juego 3.
	Rec. Rostros	ToogleButton	Cuando está activo muestra en la superficie de dibujo de la pestaña “Video” el proceso de identificación y reconocimiento facial.
5		ToogleButton	Cuándo está activo el asistente personal autónomo procede a seguir el rostro de la persona seleccionada en el elemento 6.
6		Spinner	Permite seleccionar el rostro de la persona que se desea seguir. Posee cinco opciones de usuario. • Cualquiera • Juan • Vladimir • Luis • Salome
7		ToogleButton	Cuándo está activado el asistente personal autónomo procede a seguir el objeto que se seleccione en el elemento 8.
8		Spinner	Permite seleccionar el objeto que se desea reconocer y por tanto al cual el asistente personal autónomo procederá a seguir, entre las opciones de objetos están: • Botella • Lata • Pelota
9		ToogleButton	Cuando se encuentre activado, el asistente personal autónomo procederá a seguir los gestos que realice con su mano el usuario.
10		ToogleButton	Cuando se está desactivado, el asistente no efectuará ninguna tarea de autonomía.

Tabla 77.-Tabla Descriptiva de la pestaña "AJUSTES"

Como se pudo apreciar, la aplicación posee los elementos gráficos necesarios para brindarle la funcionalidad requerida al asistente personal autónomo, elementos que fueron planteados en los objetivos de esta tesis.

5.3.2.13 Servidor de Asistencia Remota

El servidor de asistencia remota es un elemento adicional que se diseñó a fin de brindar más libertad y control sobre el asistente personal autónomo, permitiendo al usuario controlarlo de manera remota a través de la red local. Esta herramienta es de gran ayuda al momento diseñar y efectuar terapias con niños con discapacidad, ya que permite al terapista el control absoluto de cada elemento que posee este asistente. El Servidor de asistencia remota fue escrito totalmente en Java, en la Figura 193 se muestra el diagrama UML de clases usado para el mismo.

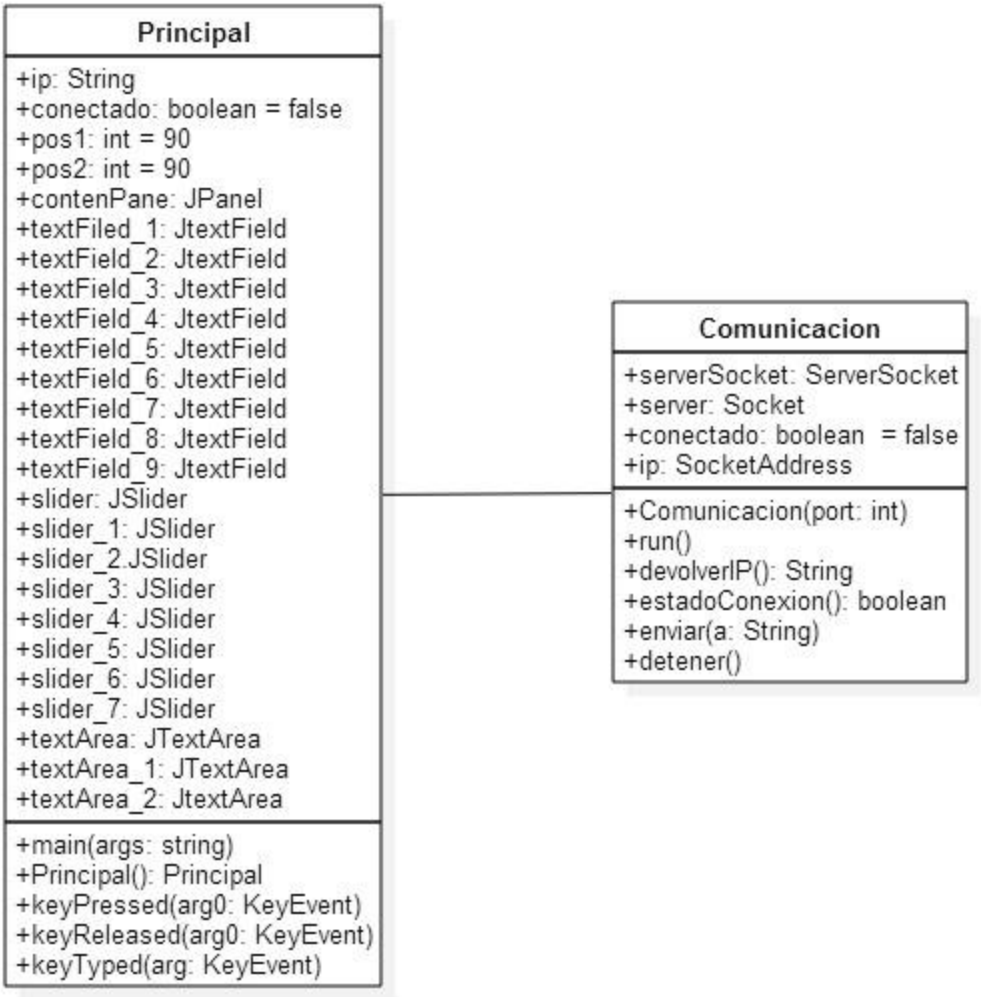


Figura 193.- Diagrama UML de clase del servidor de Asistencia Remota

Los elementos del servidor están distribuidos en tres pestañas:

- Detalles de conexión
- Interacción
- Movimiento

En las Figuras 194, 195 y 196 se pueden apreciar en detalle cada una de las tres pestañas antes mencionadas.

Pestaña "Detalles de Conexión": en esta pestaña se muestran algunos parámetros de conexión necesarios para que la aplicación de asistencia personal se pueda conectar al servidor:

- La IP actual del servidor
- El puerto que se usará para la conexión, en este caso el 6000
- Una ventana que muestra los registro de conexiones que se realicen.

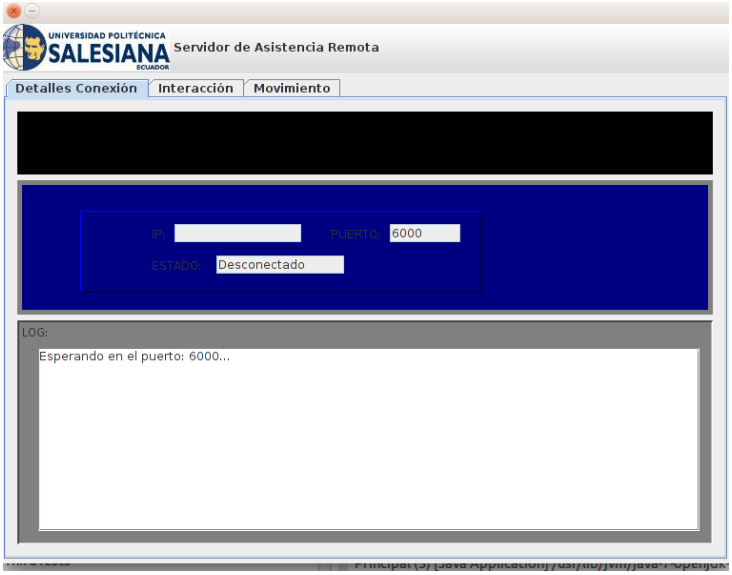


Figura 194.- Captura de la pestaña "Detalles de Conexión" del Servidor de Asistencia Remota

Pestaña "Interacción": Desde esta pestaña se pueden controlar todas las actividades que el asistente personal autónomo posee:

- Activar el seguimiento de manos
- Activar el seguimiento de objetos, pudiendo indicar que objeto se quiere seguir
- Activar el seguimiento de rostros, pudiendo indicar que rostro se quiere seguir.
- Se puede indicar que en la pantalla del dispositivo inteligente se muestre el avatar interactivo.
- Se puede indicar que en la pantalla del dispositivo inteligente se visualizará ya sea el juego 1, 2 o 3.
- Se puede mandar a reproducir una determinada canción, pasándole el identificador de la misma.
- Se puede solicitar que el robot relate un cuento, pasándole el identificador del mismo

- Se puede mandar a reproducir un efecto de sonido, pasándolo el identificador del mismo
- Se puede mandar a leer el texto que se ingrese en el campo correspondiente.
- Se puede lanzar el reconocimiento de órdenes por voz, en cuyo caso el resultado del mismo se observará en la pantalla correspondiente.

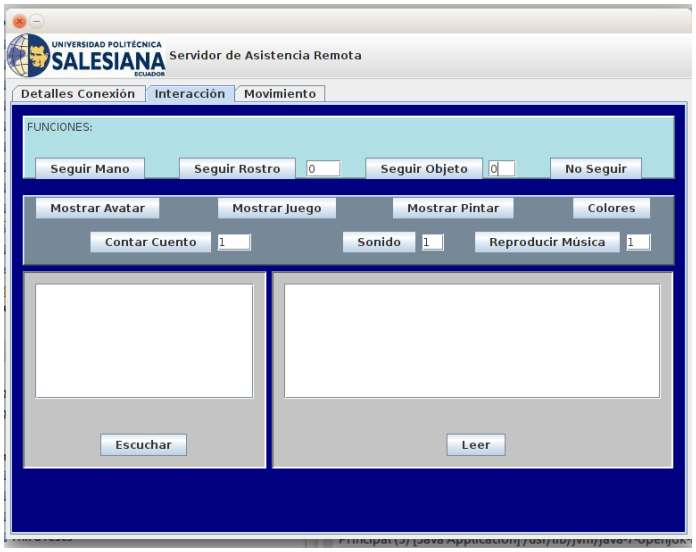


Figura 195.-Captura de la pestaña "Interacción" del Servidor de Asistencia Remota

Pestaña "movimiento": esta pestaña permite el control de todos los servomotores de la plataforma de manera fácil e intuitiva. Las opciones que se maneja con esta ventana son:

- Control de la posición de los servomotores que conforman la cabeza.
- Control de la posición de los servomotores de la pinza.
- Control de la posición de los servomotores de los tres sensores.
- Control de la dirección y velocidad de la plataforma

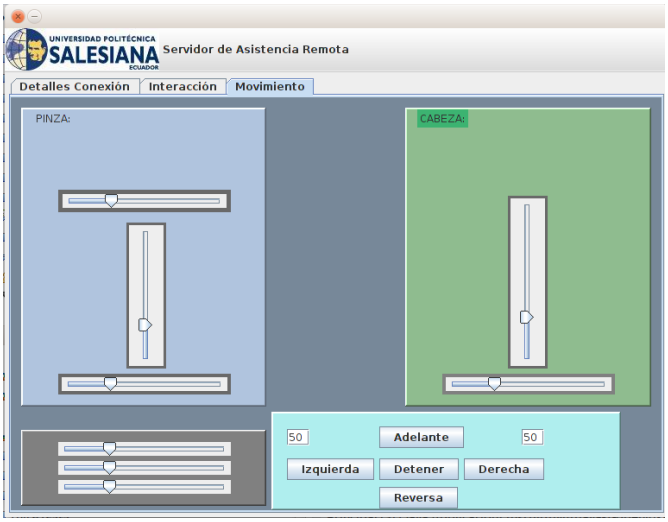


Figura 196.-Captura de la pestaña "Movimiento" del Servidor de Asistencia Remota

5.3.3 Funcionalidad y aplicabilidad

En esta sección se hará un resumen muy breve de todas las funciones que la aplicación de asistencia personal posee, como también las áreas en las que puede ser usado y los modos de funcionamiento que están permitidos.

5.3.3.1 Modos de Funcionamiento

La aplicación de asistencia personal puede funcionar de distintas maneras y en distintos modos, de acuerdo a la necesidad de cada usuario:

- Idealmente fue diseñada para trabajar en conjunto con la plataforma de desplazamiento. Cuando el dispositivo inteligente está integrado a la plataforma mecánica forma nuestro "asistente personal con desplazamiento autónomo".
- La aplicación de asistencia personal puede trabajar sin la plataforma de desplazamiento, es decir, solo con la aplicación instalada en el teléfono inteligente.
- El asistente personal puede funcionar con o sin el servidor de asistencia remota.
- Cuando el asistente personal está conectado al servidor de asistencia remota, puede trabajar con o sin la plataforma de desplazamiento.

5.3.3.2 Aplicabilidad

El proyecto no excluye a ningún grupo ni sector social, pero tiene como finalidad principal brindar servicios de asistencia en el área de discapacidad y soporte al adulto mayor, así como herramienta de soporte en el desarrollo de destrezas y habilidades motoras e intelectuales de niños. Además, puede servir para cualquier persona que lo quiera utilizar como herramienta de asistencia, estudio o entretenimiento.

Finalmente, se espera que este proyecto brinde las herramientas necesarias para que futuros desarrolladores puedan utilizar el autómata como herramienta de estudio en el área de la robótica y la inteligencia artificial, ya que en ella se podrán practicar e implementar nuevas técnicas según se requiera, permitiendo al asistente personal autónomo ir evolucionando y beneficiando una vez más a los sectores antes mencionados.

5.4 Pruebas de laboratorio y corrección de errores.

En esta sección se describirá el plan de pruebas diseñando para evaluar el funcionamiento y desempeño del asistente personal autónomo, así como cada elemento que forma parte del mismo.

5.4.1 Plan de pruebas

Para verificar el funcionamiento del sistema completo se aplicaron cuatro tipos de pruebas, cada una de ellas evalúa partes o funciones diferentes del mismo. De esta forma, se establecieron las siguientes pruebas:

- Pruebas Alpha
- Pruebas Beta
- Pruebas Teta
- Pruebas Zeta

5.4.1.1 Pruebas Alpha

Estas pruebas fueron diseñadas a fin de evaluar el porcentaje de reconocimiento gestos, así como el porcentaje de reconocimiento de objetos, variando los pesos en un rango 1.0 a 0.0 tanto para la distancia mínima de los momentos correspondientes a la región convexa, como la distancia mínima de los momentos correspondientes a la aproximación poligonal. Con esto se buscó determinar los pesos ideales y la mejor tasa de reconocimiento. Para cada etapa de reconocimiento se realizaron 11 pruebas variando los pesos como se muestra en la Tabla 78.

Prueba	**Pesos**	
	p_1	p_2
1	1.0	0.0
2	0.9	0.1
3	0.8	0.2
4	0.7	0.3
5	0.6	0.4
6	0.5	0.5
7	0.4	0.6
8	0.3	0.7
9	0.2	0.8
10	0.1	0.9
11	0.0	1.0

Tabla 78.- Tabla de los valores para p1 y p2 para las pruebas de Reconocimiento de Gestos

Para el caso del reconocimiento de gestos, se posee un corpus simétrico de 80 muestras, con 20 muestras por cada gesto. Con ello, para cada peso presentado en la tabla anterior, se realiza una prueba ingresando todas las muestras, a fin de contar cuántas veces un gesto es reconocido exitosamente. Una vez que se efectúan las once pruebas con los distintos pesos, se determina cuál es el valor ideal para el mismo, a fin de obtener la mejor tasa de reconocimiento.

Para el caso de reconocimiento de objetos se realizaron dos pruebas: la primera con las distancias mínimas de los momentos de la aproximación poligonal y la segunda con las distancias mínimas de los momentos de la región convexa. Para este caso el corpus de prueba consta de 60 muestras, con 20 muestras para cada objeto.

5.4.1.2 Pruebas Beta

Esta prueba consiste en comprobar la compatibilidad de la aplicación en distintos dispositivos, así como el uso de CPU que se hace en cada uno de ellos. Los datos se extrajeron con el dispositivo conectado a la perspectiva DDMS de Eclipse, a fin obtener el consumo mínimo y máximo del CPU para cada caso.

5.4.1.3 Pruebas Teta

Estas pruebas se realizaron en el laboratorio aplicando las evaluaciones que se muestra en el Anexo 5 a los integrantes que colaboran dentro del Grupo de Investigación en Inteligencia Artificial y Tecnologías de Asistencia de la Universidad Politécnica Salesiana.

Se tomaron un total de seis encuestas para valorar cada elemento de la aplicación. Las evaluaciones aplicadas fueron las siguientes:

1. Evaluación del aspecto estético de la aplicación.
2. Evaluación de la animación del avatar.
3. Evaluación del funcionamiento de la aplicación.
4. Evaluación de interacción con el usuario mediante la visión artificial.
5. Evaluación de la interacción con el usuario mediante reconocimiento de órdenes empleando comandos de voz.
6. Evaluación del servidor de asistencia remota.

Las personas que intervinieron en las encuestas de evaluación se muestran en la Tabla 79.

Pruebas de Laboratorio		
Nombre	**Edad (años)**	**Ocupación**
Daysi Arévalo	28	Ingeniera en Sistemas
Juan Guillermo	26	Estudiante/Investigación
Marco Capón	25	Estudiante/Investigación

Priscila Cabrera	25	Ingeniera en Sistemas
Diana Monje	26	Estudiante/Investigación
Diego Quisi	25	Técnico Docente de la UPS
Edison Güiñansaca	25	Estudiante/Investigación

Tabla 79.- Información de los encuestados en las Pruebas Teta

La valoración para todas la encuestas realizadas tenía como posibles respuestas "Si" o "No".

5.4.1.4 Pruebas Zeta

Estas pruebas decampo se planearon a fin de observar la aplicabilidad y la ayuda que este sistema puede brindar como herramienta de soporte dentro del área de discapacidad. La actividades de terapia fue aplicada en la Fundación "Jesús para los Niños" ubicada en el cantón Cañar, con la autorización de las autoridades y la ayuda de las terapistas de dicha fundación.

La terapia aplicada se conoce como "Actividades Iniciales" y tiene como objetivo principal despertar la atención de los niños e incrementar su nivel de concentración, preparándolos para las actividades de terapia que se llevarán a posteriori durante el trascurso del día.

El número de alumnos que participaron en el proyecto del asistente personal autónomo en la Fundación "Jesús para los Niños" fue de 26, de los cuales 20 sufren discapacidad intelectual moderada y grave, 2 tienen Parálisis Cerebral Infantil (PCI), 2 presentan multi-discapacidad, 1 niña presenta Trastorno de Déficit de Atención e Hiperactividad (TDAH) y 1 niño que sufre espectro autista.

Los profesionales que participaron íntegramente durante la terapia tienen los siguientes títulos:

- Psicólogos educativos
- Educadores especiales
- Psicólogos clínicos
- Estimulación temprana
- Terapista ocupacional

La terapia se efectuó con la ayuda del servidor de asistencia remota y se muestra en el Anexo 7.

La evaluación final para medir el desempeño del asistente personal autónomo se realizó a través de dos terapias, una con el asistente personal y otra sin él. Al final se aplicó una evaluación a los seis terapistas que estuvieron presentes durante las dos terapias y se evaluaron aspectos importantes tales como:

- El nivel de concentración de los niños
- El tiempo ahorrado con respecto a la terapia tradicional sin usar el asistente

- Analizar si los niños se concentran mejor durante las actividades de terapia.
- Si las actividades que posee el asistente personal autónomo son adecuadas para trabajar con niños con diferentes discapacidades
- En qué discapacidades se consideraría que puede existir mejores resultados gracias a la ayuda del asistente.
- Mejoras que podrían implementarse en el asistente.

La evaluación aplicada a los terapistas se puede encontrar en el Anexo 6.

5.5 Análisis de resultados.

5.5.1 Resultados de las pruebas Alpha

5.5.1.1 Resultados de las Pruebas para el Reconocimiento de Gestos de la Mano

Para $p_1 = 1.0$ y $p_2 = 0.0$ los resultados de reconocimiento se muestran en la Figura 197, en este caso se tiene una precisión total del 76.25%.

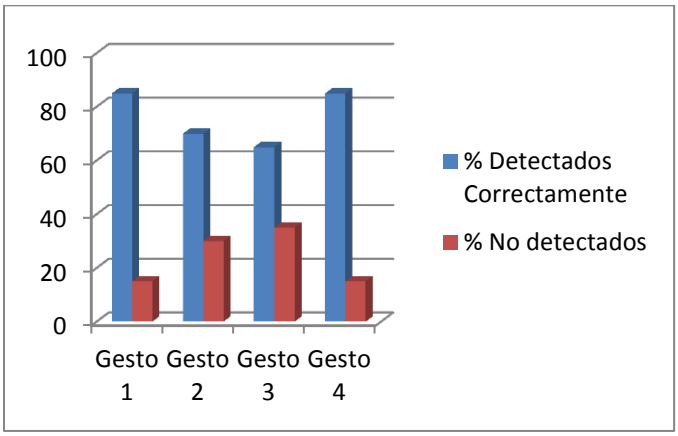


Figura 197.- Porcentajes de Gestos reconocidos para un peso p1=1.0 y p2=0.0

Para $p_1 = 0.9$ y $p_2 = 0.1$, los resultados de reconocimiento se muestran en la Figura 198, en este caso se tiene una precisión total del 83.75%.

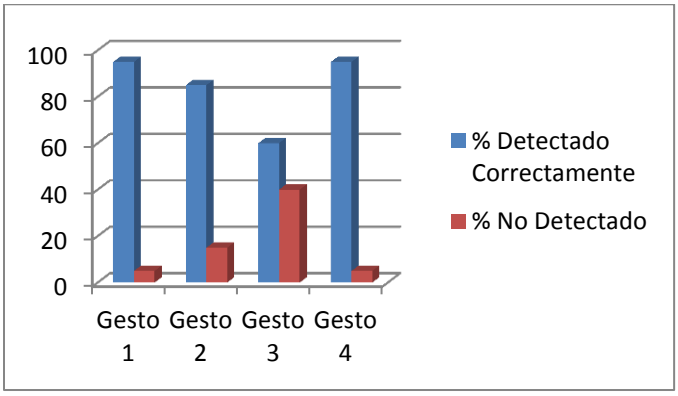


Figura 198.- Porcentajes de Gestos reconocidos para un peso p1=0.9 y p2=0.1

Para $p_1 = 0.8$ y $p_2 = 0.2$, los resultados de reconocimiento se muestran en la Figura 199, en este caso se tiene una precisión total del 91.25%.

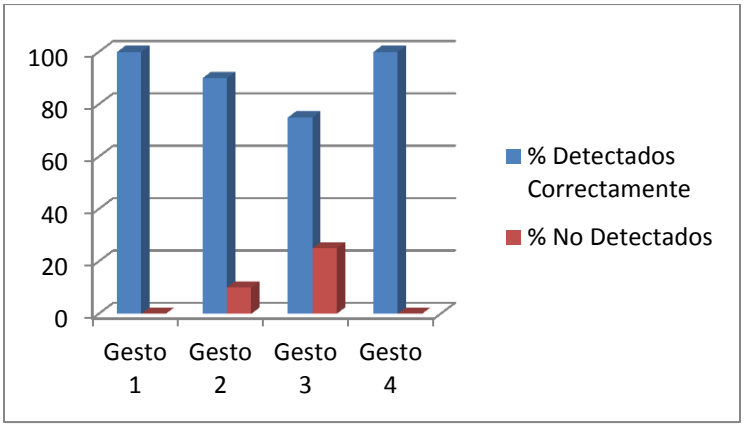


Figura 199.- Porcentajes de Gestos reconocidos para un peso p1=0.8 y p2=0.2

Para $p_1 = 0.7$ y $p_2 = 0.3$, los resultados de reconocimiento se muestran en la Figura 200, en este caso se tiene una precisión total del 90%.

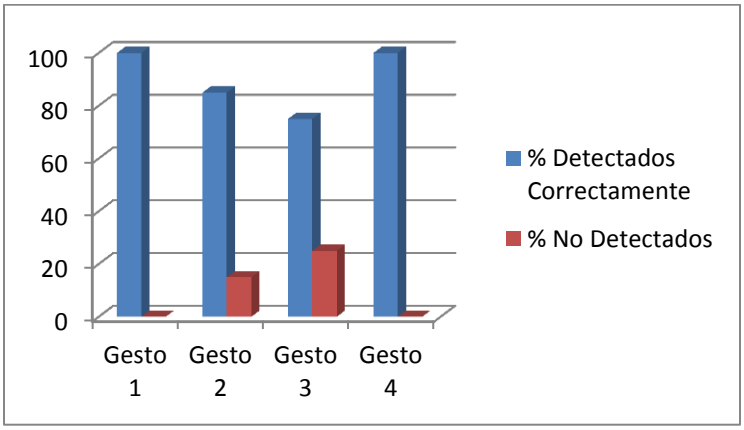


Figura 200.- Porcentajes de Gestos reconocidos para un peso p1=0.7 y p2=0.3

Para $p_1 = 0.6$ y $p_2 = 0.4$, los resultados de reconocimiento se muestran en la Figura 201, en este caso se tiene una precisión total del 93.75%.

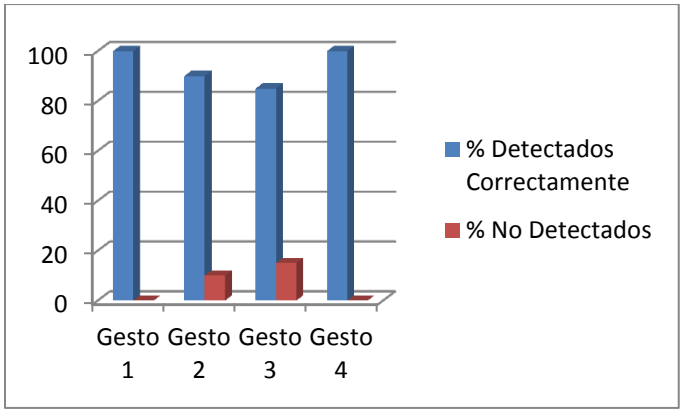


Figura 201.- Porcentajes de Gestos reconocidos para un peso p1=0.6 y p2=0.4

Para $p_1 = 0.5$ y $p_2 = 0.5$, los resultados de reconocimiento se muestran en la Figura 202, en este caso se tiene una precisión total del 85%.

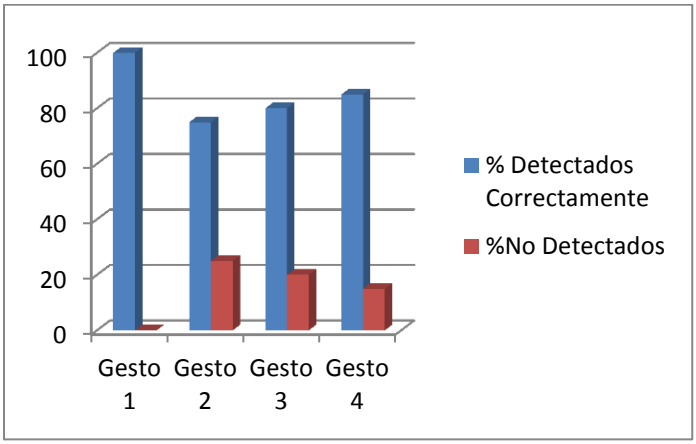


Figura 202.- Porcentajes de Gestos reconocidos para un peso p1=0.5 y p2=0.5

Para $p_1 = 0.4$ y $p_2 = 0.6$, los resultados de reconocimiento se muestran en la Figura 203, en este caso se tiene una precisión total del 87.5%.

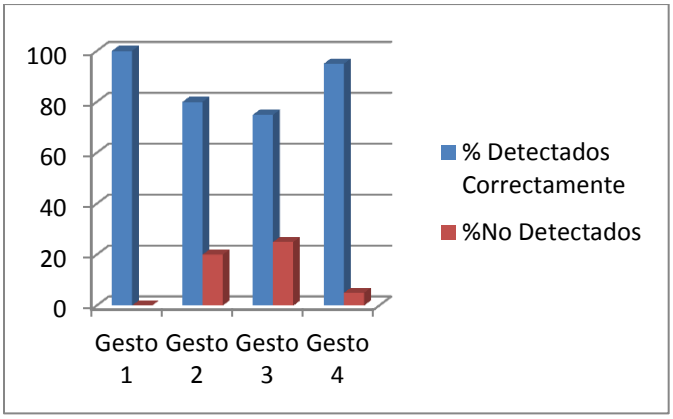


Figura 203.- Porcentajes de Gestos reconocidos para un peso p1=0.4 y p2=0.6

Para $p_1 = 0.3$ y $p_2 = 0.7$, los resultados de reconocimiento se muestran en la Figura 204, en este caso se tiene una precisión total del 83%.

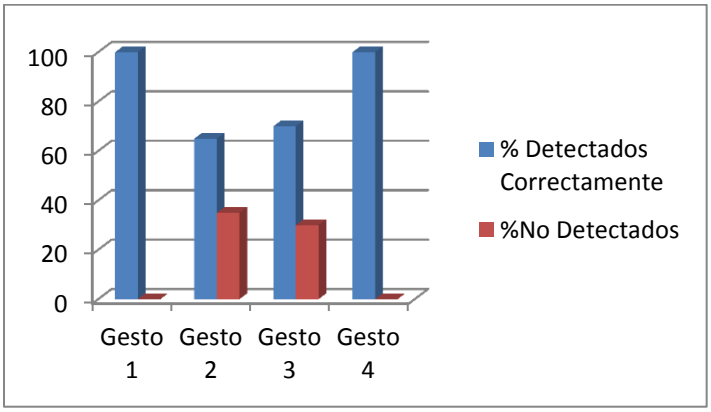


Figura 204.- Porcentajes de Gestos reconocidos para un peso p1=0.3 y p2=0.7

Para $p_1 = 0.2$ y $p_2 = 0.8$, los resultados de reconocimiento se muestran en la Figura 205, en este caso se tiene una precisión total del 83.75%.

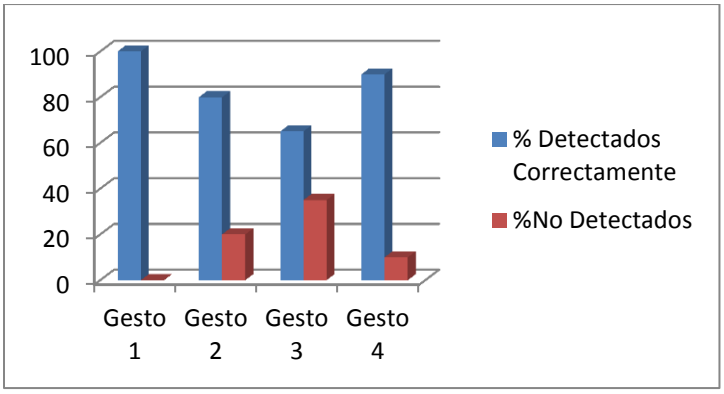


Figura 205.- Porcentajes de Gestos reconocidos para un peso p1=0.2 y p2=0.8

Para $p_1 = 0.1$ y $p_2 = 0.9$, los resultados de reconocimiento se muestran en la Figura 206, en este caso se tiene una precisión total del 75%.

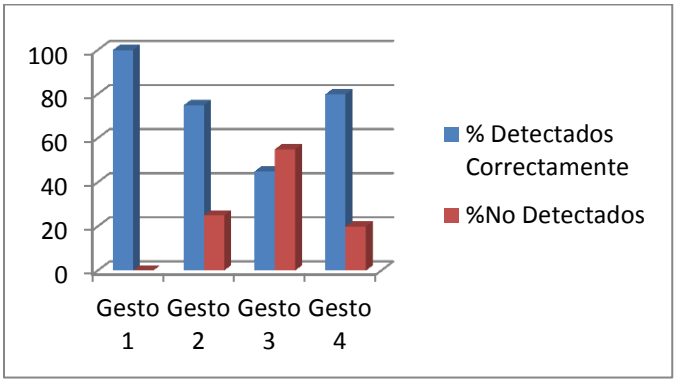


Figura 206.- Porcentajes de Gestos reconocidos para un peso p1=0.9 y p2=0.1

Para $p_1 = 0.0$ y $p_2 = 1.0$, los resultados de reconocimiento se muestran en la Figura 207, en este caso se tiene una precisión total del 78.75%.

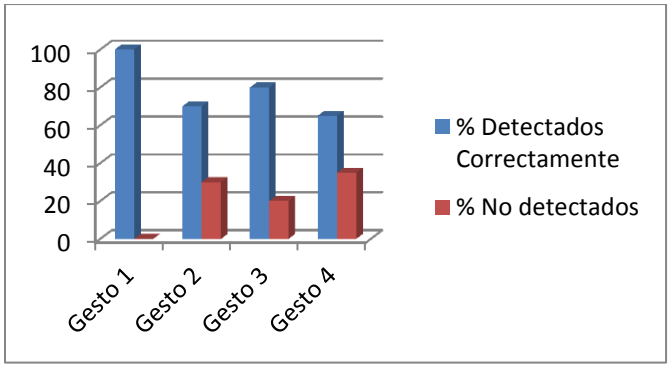


Figura 207.- Porcentajes de Gestos reconocidos para un peso p1=0.0 y p2=1.0

5.5.1.2 Resultados de las Pruebas para el reconocimiento de Objetos

Usando la aproximación Poligonal, se tiene una precisión del 98.33% como se muestra en la Figura 208.

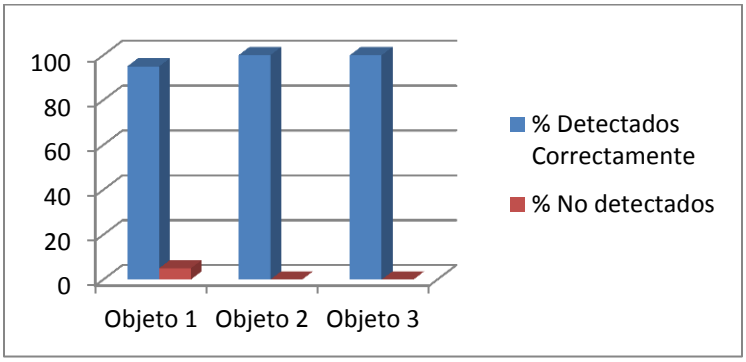


Figura 208.- Porcentaje de reconocimiento de Objetos usando la Aproximación Poligonal

Usando la Región Convexa, se tiene una precisión del 95% como se muestra en la Figura 209.

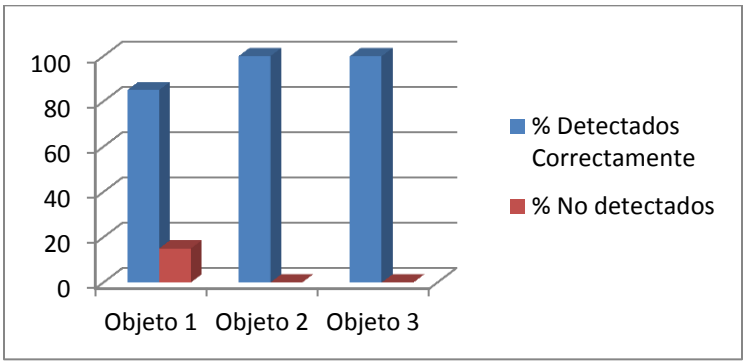


Figura 209.- Porcentaje de Reconocimiento de Objetos usando la Región Convexa

5.5.2 Resultados de las pruebas Beta

- Compatibilidad: como fue previsto, la aplicación fue instalada en los ocho dispositivos inteligentes. En siete de ellos la aplicación se ejecutó satisfactoriamente con toda normalidad. El único dispositivo en el cual la aplicación no pudo ser ejecutada fue en el "Huawei U8180", ya que este dispositivo no posee cámara frontal y su procesador es insuficiente para ejecutar las operaciones realizadas durante el procesamiento de imágenes.
- Uso del CPU: En la Figura 210 se muestran los resultados del consumo mínimo y máximo del CPU, cuando la aplicación está siendo ejecutada. Según las pruebas se tiene que el consumo del CPU oscila del 36% al 67% del total del mismo.

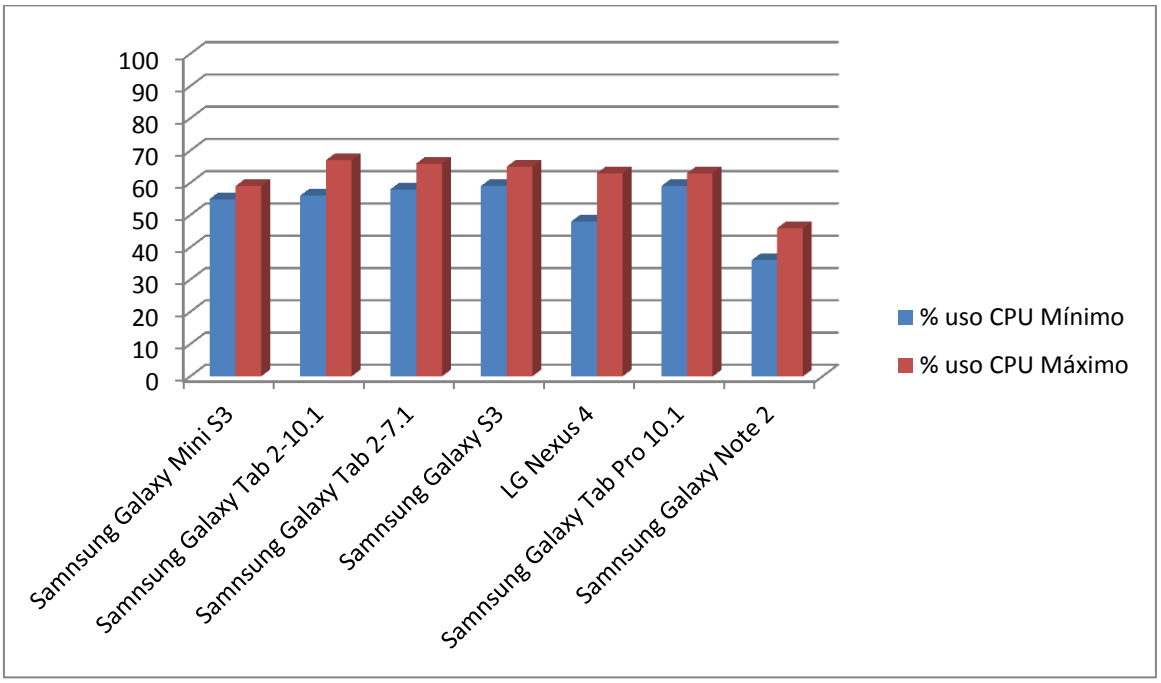


Figura 210.- Porcentaje de consumo del CPU para los distintos teléfonos Inteligentes

El consumo más bajo del CPU se aprecia en los teléfonos de Gama Alta como es lógico, tal es el caso para el "Samsung Galaxy Note 2".

5.5.3 Resultados de las pruebas Zeta

5.5.3.1 Resultados de la Evaluación del aspecto estético de la aplicación.

En la Figura 211 se puede apreciar los resultados por pregunta para esta evaluación, se tiene un 95% de respuestas afirmativas y un 5% de respuestas negativas.

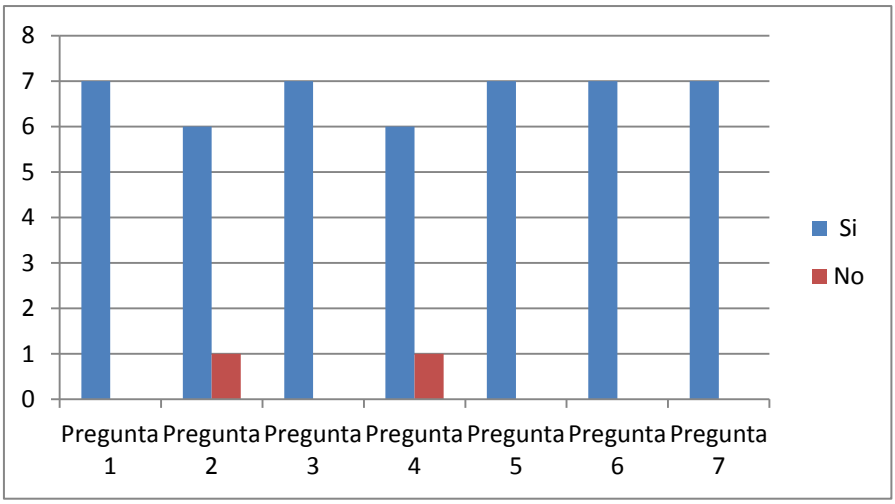


Figura 211.- Resultados de la Evaluación del Aspecto estético de la aplicación

Observaciones: los encuestados sugieren que se podrían aplicar estilos a cada tema y hacer que la aplicación sea más intuitiva.

5.5.3.2 Resultados de la Evaluación del Avatar Animado.

En la Figura 212 se puede apreciar los resultados por pregunta para esta evaluación, se tiene un 85% de respuestas afirmativas y un 15% de respuestas negativas.

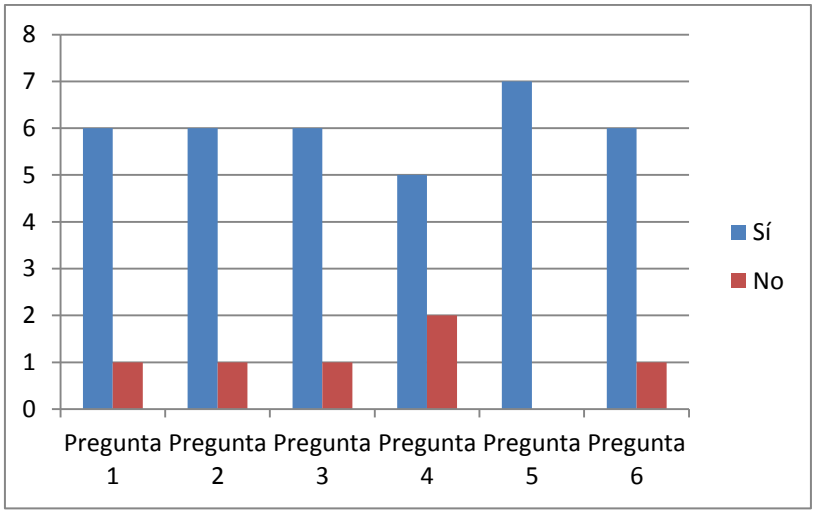


Figura 212.- Resultados para la Evaluación del Avatar Animado

Observaciones: los encuestados sugieren que se le podría dar más naturalidad al avatar si se le incluyera un cuello para el movimiento de la cabeza, dándole un aspecto más a un humano. También sugieren que los movimientos cuando el avatar animado habla sean más fluidos.

5.5.3.3 Resultados de la Evaluación del Funcionamiento de la Aplicación.

En la Figura 213 se puede apreciar los resultados por pregunta para esta evaluación, se tiene un 95% de respuestas afirmativas y un 5% de respuestas en blanco.

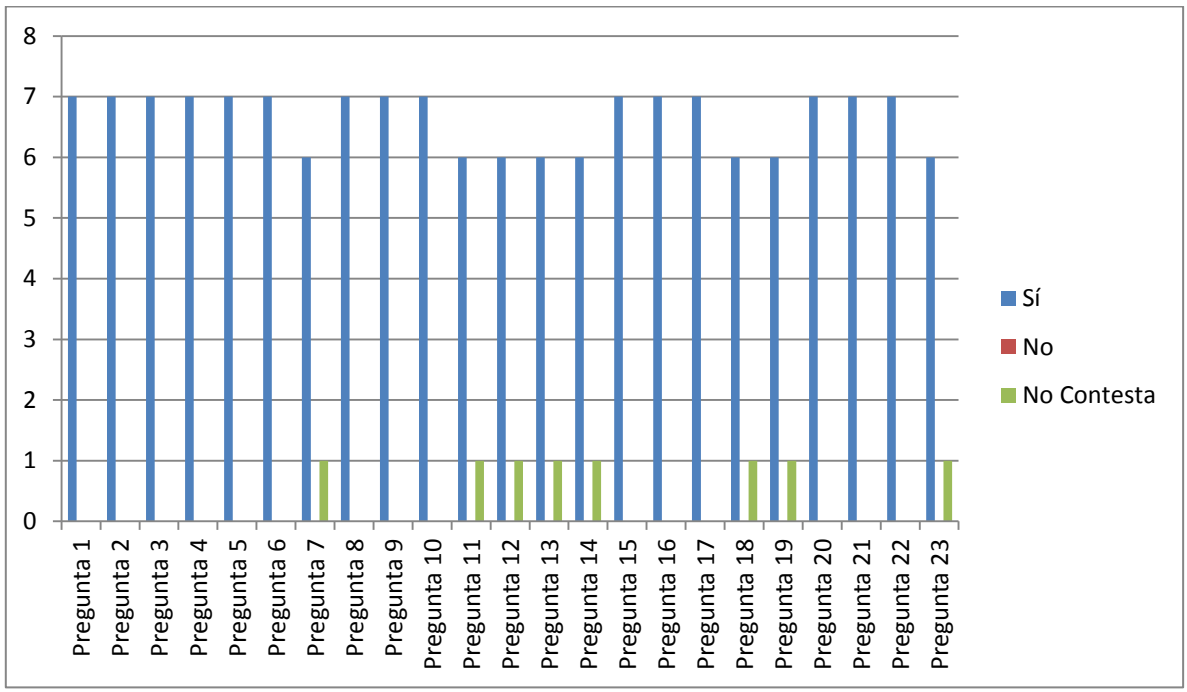


Figura 213.- Resultados para la evaluación de Funcionamiento de la Aplicación

Observaciones: los encuestados sugieren que se debería poder visualizar el listado de los recordatorios almacenados a fin de eliminarlos o modificarlos según sea el caso, también que la contraseña que se almacena debería estar encriptada. Los encuestados también sugirieron que debería existir una opción para restablecer los valores por defecto para la segmentación por color. Y como observación final se sugiere incluir textos de ayuda o un manual de usuario.

5.5.3.4 Resultados de la Evaluación de Interacción con el usuario mediante la Visión Artificial.

En la Figura 214 se puede apreciar los resultados por pregunta para esta evaluación, se tiene un 94% de respuestas afirmativas, un 4% de respuestas negativas y un 2% de respuestas en blanco.

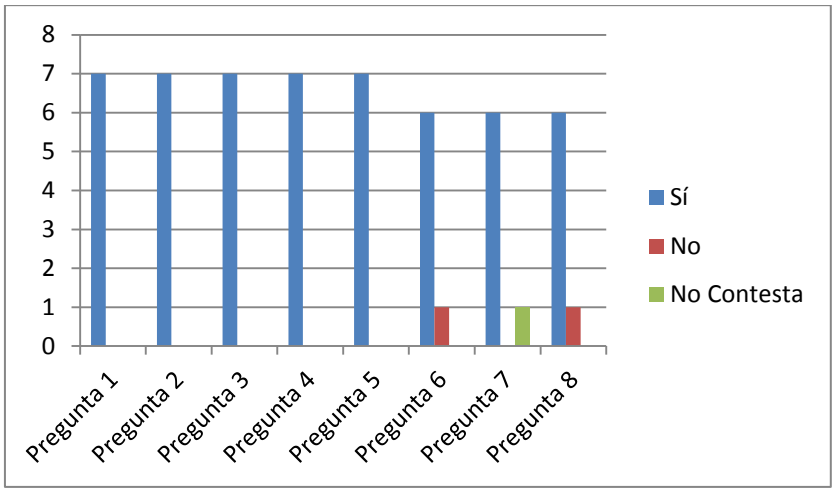


Figura 214.-Resultados para la Evaluación de Interacción con el Usuario mediante la Visión Artificial

5.5.3.5 Resultados de la Evaluación de la Interacción con el usuario mediante Reconocimiento de órdenes por Voz.

En la Figura 215 se puede apreciar los resultados por pregunta para esta evaluación, se tiene un 98% de respuestas afirmativas y un 2% de respuestas negativas.

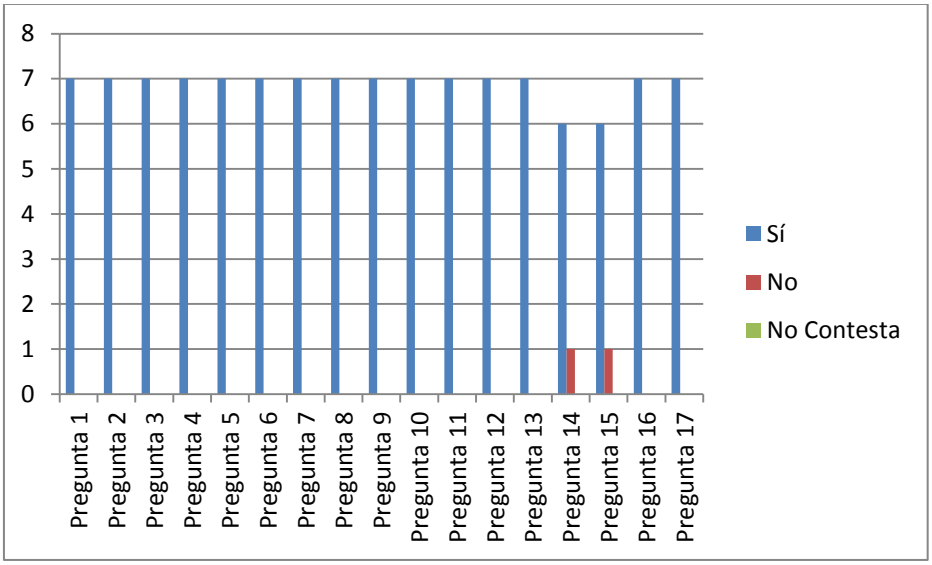


Figura 215.- Resultados de la Evaluación de la Interacción con el usuario mediante el Reconocimiento de Ordenes por voz

5.5.3.6 Resultados de la Evaluación del Servidor de Asistencia Remota.

En la Figura 216 se puede apreciar los resultados por pregunta para esta evaluación, se tiene un 97% de respuestas afirmativas y un 3% de respuestas en blanco.

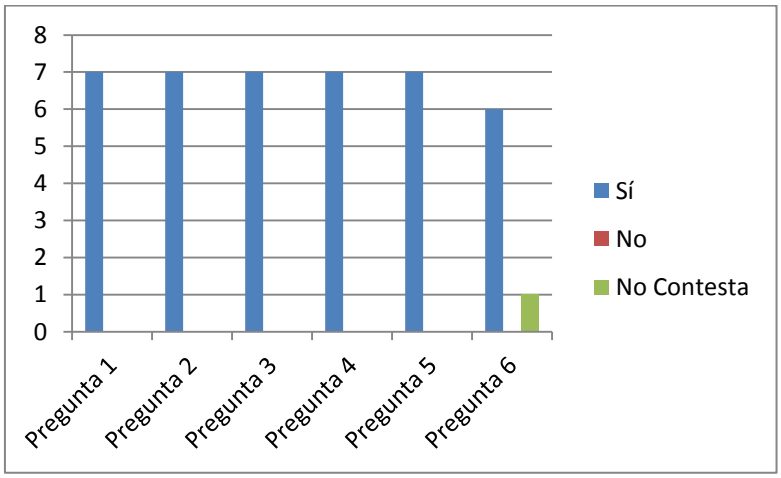


Figura 216.- Resultados de la Evaluación del Servidor de Asistencia Remota

En base a las evaluaciones realizadas, se determina que la gran mayoría responde afirmativamente, lo que indica que el funcionamiento de todo el sistema es correcto según lo esperado.

5.5.4 Resultados de las pruebas Teta

A continuación se detallan los resultados por pregunta de la evaluación aplicada a los terapistas de la Fundación "Jesús Para Los Niños" del cantón Cañar

1. **¿Piensa usted que el sistema de Asistencia Personal puede contribuir al proceso de rehabilitación de los niños con discapacidad?**

De seis terapistas las seis respondieron que "sí".

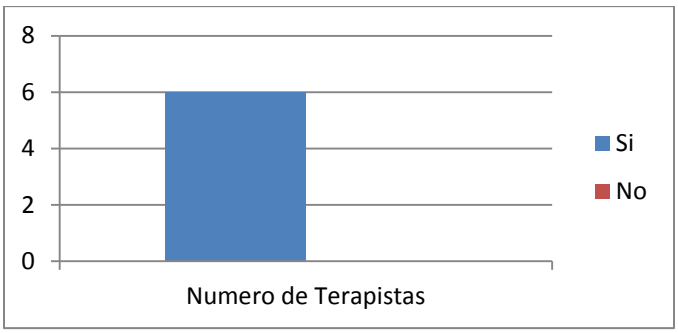


Figura 217.- Resultados de la pregunta número 1 de la evaluación aplicada en las Pruebas Zeta

El porqué: Es un sistema interactivo, a través del cual se puede facilitar la terapia durante el proceso de rehabilitación gracias a la gran cantidad de actividades interactivas que se pueden realizar aprovechando los últimos avances tecnológicos.

2. **¿La terapia con el Asistente Personal Autónomo se llevó con mayor rapidez?**

De seis terapistas las seis respondieron que "sí".

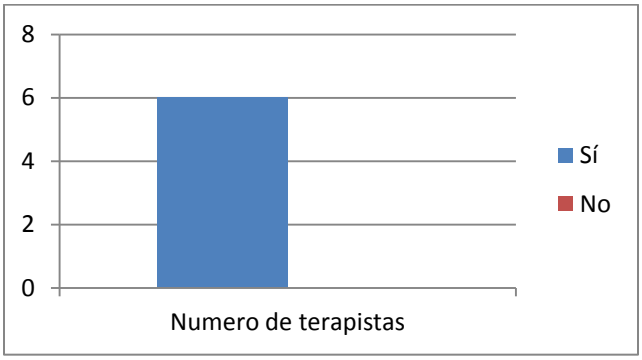


Figura 218.-Resultados de la pregunta número 2 de la evaluación aplicada en las Pruebas Zeta

El porqué: la atención de los niños se concentró en el autómata, permitiéndoles responder rápidamente a las actividades o interrogantes que realizaba el autómata.

3. **En el caso de que la respuesta anterior sea afirmativa, indique un porcentaje aproximado de 0 a 100% de tiempo que Usted considera que se ha ahorrado.**

El promedio de tiempo ahorrado según los terapistas es del: 49%

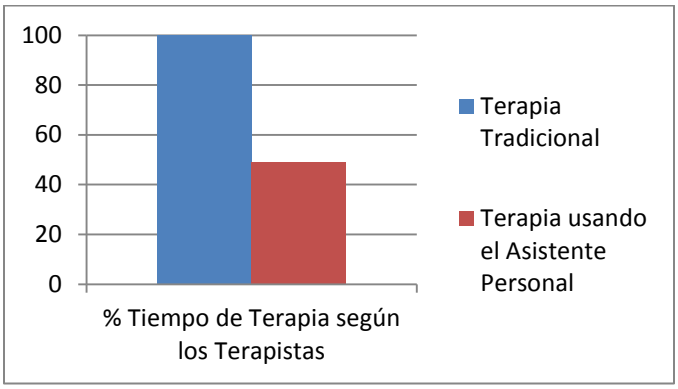


Figura 219.- Porcentaje de tiempo que los Terapistas piensa que se ahorró con respecto a la terapia tradicional

El tiempo de la terapia tradicional fue de 1 hora y la que se llevó con el autómata personal fue de 25 minutos, de forma que el porcentaje real en que se redujo el tiempo de terapia fue del 42%

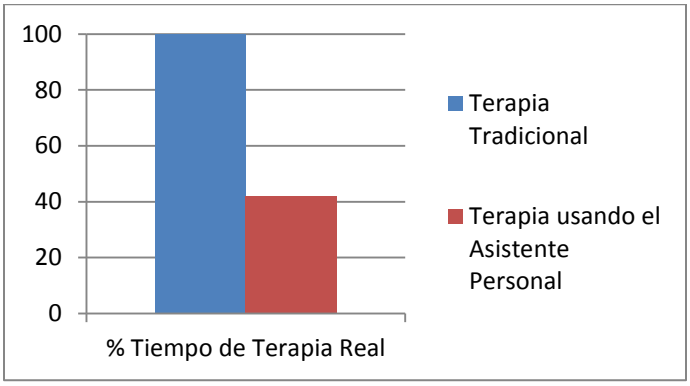


Figura 220.- Porcentaje real tiempo Ahorrado en la terapia con el Asistente Personal

4. **¿Considera usted que el Asistente personal fue factor que modificó cierto tipo de conductas de los niños durante la terapia?**

De seis terapistas las seis respondieron que “sí”.

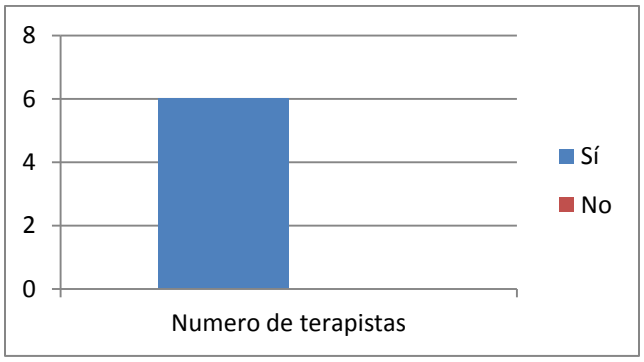


Figura 221.-Resultados de la pregunta número 4 de la evaluación aplicada en las Pruebas Zeta

El porqué: se vio un cambio en el nivel de atención de los niños con problemas conductuales y TDAH. Gracias a la forma y funcionalidad del autómata se despertó la curiosidad y la atención de los niños.

5. **¿El nivel de atención de los niños mejoró con la presencia del Asistente Personal Autónomo?**

De seis terapistas las seis respondieron que "sí".

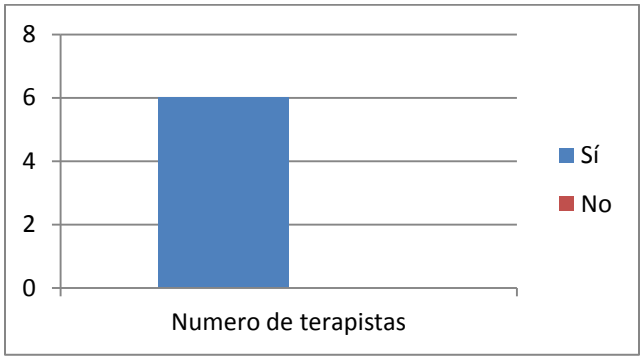


Figura 222.-Resultados de la pregunta número 5 de la evaluación aplicada en las Pruebas Zeta

El porqué: realizaron la terapia siguiendo las órdenes del asistente personal casi sin ayuda del terapista, manteniendo una atención sostenida sobre el mismo. Además el lenguaje que hablaba el asistente personal era claro y las actividades que presentaba eran entretenidas.

6. **¿Cree que la Asistencia Remota para el control del autómata es útil para mejorar las actividades con los niños con discapacidad?**

De seis terapistas las seis respondieron que "sí".

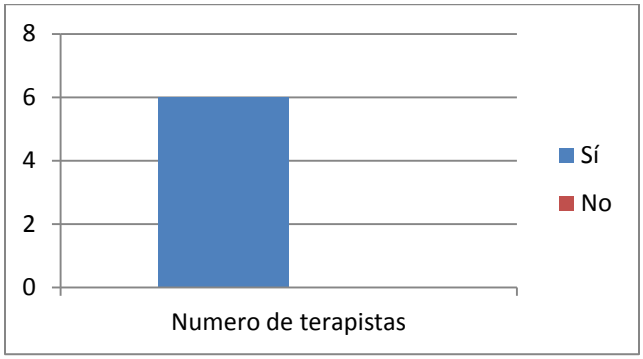


Figura 223.-Resultados de la pregunta número 6 de la evaluación aplicada en las Pruebas Zeta

El porqué: se puede crear, desarrollar, modificar o preparar una gran cantidad de terapias de acuerdo a las necesidades de cada niño.

7. **¿Las actividades que posee el autómata son correctas para trabajar con grupos de niños con diferentes necesidades educativas especiales asociadas a una discapacidad?**

De seis terapistas las seis respondieron que "sí".

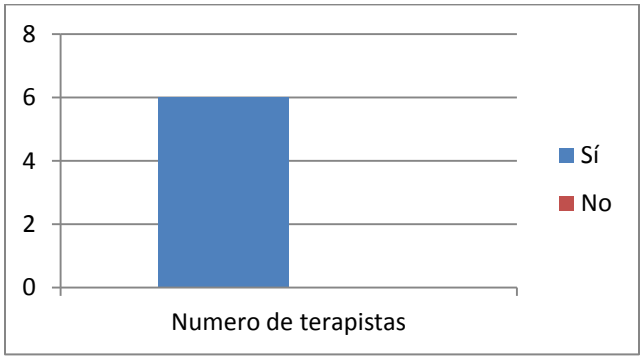


Figura 224.-Resultados de la pregunta número 7 de la evaluación aplicada en las Pruebas Zeta

El porqué: puesto que posee actividades lúdico-interactivas que son ideales para trabajar con grupos de niños con diferentes discapacidades, permitiéndoles desarrollar competencias o destrezas de acuerdo a la necesidad de cada uno. Finalmente, se pueden programar muchas actividades desde el monitoreo remoto.

8. ¿Considera que las actividades basadas en la realidad pueden contribuir al desarrollo psico-motriz de los niños?

De seis terapistas las seis respondieron que "sí".

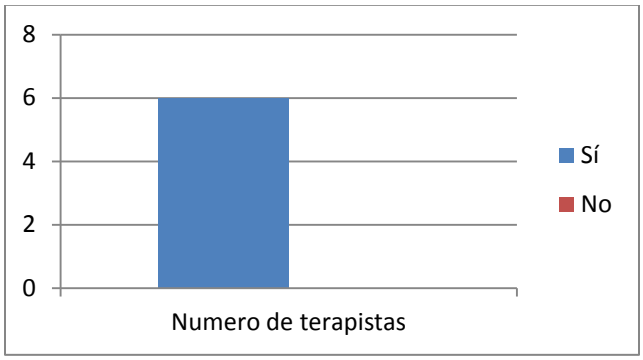


Figura 225.-Resultados de la pregunta número 8 de la evaluación aplicada en las Pruebas Zeta

El porqué: a través de la manipulación de objetos dentro de la realidad aumentada pueden mejorar sus destrezas motrices. Les incentiva a los niños a desarrollar todo su potencial.

9. ¿En qué niños se apreció un cambio en la conducta con la presencia del Asistente Personal Autónomo?

En los niños con trastornos conductuales, déficit de atención, TDAH, discapacidad intelectual, síndrome de Down y parálisis cerebral.

10. ¿En qué considera que se puede mejorar? O ¿Qué actividades se podrían implementar en el autómata?

- Actividades de percepción visual y auditiva
- Actividades de razonamiento lógico
- Actividades de atención
- Actividades para la modificación de conducta
- Fonemas
- Juegos que incluyan números y letras

11. ¿Qué actividades que posee el autómata considera usted que puede contribuir más al proceso de rehabilitación y para qué discapacidad?

- Las actividades relacionadas con percepción, motricidad y atención.
- Las actividades relacionadas con formas, tamaños, colores para niños con dificultad en el lenguaje, parálisis cerebral y déficit de atención.
- Creando niveles de comunicación en niños con PCI.

- Las actividades de atención, puesto que se vio un cambio en niños con trastornos conductuales y TDAH.
- Las actividades de lectoescritura para mejorar el proceso de aprendizaje principalmente en niños con PCI y TDAH.
- Las actividades de realidad aumentada ya que puede mejorar el desarrollo psico-motriz de la mayoría de niños.
- Todas, puesto que cada actividad puede ser aplicada para niños con distintas discapacidades.

El proceso de convertir un teléfono móvil en un asistente personal autónomo, fue un bastante arduo y a la vez complejo, ya que implicó el desarrollo e investigación de cada componente mecánico, cada elemento de hardware y cada elemento software que intervendría en el mismo.

El desarrollo de la estructura mecánica que soportaría todos los componentes electrónicos y el dispositivo inteligente, fue un proceso muy complejo en ciertas ocasiones, pues era necesario considerar los materiales que se usaría y los diseños mecánicos que se construirían para que la plataforma pudiese cumplir los requisitos que se habían planteado. Después de descartar varios diseños y fabricar algunos prototipos de plataforma, se logró construir un modelo que cumplía todos los requerimientos para convertir un dispositivo móvil inteligente en un asistente personal autónomo.

El diseño y construcción de la placa de control de la plataforma fue un proceso relativamente fácil cuyo costo es bastante bajo, dado que se usaron componentes electrónicos que se pueden encontrar fácilmente en el mercado actual. Todos los elementos de la placa son genéricos y están montados sobre sockets, aspecto que permite que sean fáciles de remplazar en caso de ser necesario.

Una de las partes más complejas fue el desarrollo y aplicación del protocolo de comunicación que manejaría la plataforma, pues era necesario que cumpla determinados criterios tales como la detección y corrección de errores. Luego de probar con varios lenguajes de programación para microcontroladores PIC sin obtener buenos resultados, se optó por escribir todo el código en C++, dando como resultado un protocolo de comunicación bastante robusto, que permite controlar cada componente de la plataforma mecánica en tiempo real sin que el microcontrolador se cuelgue. El protocolo de comunicación se estandarizó a fin de que las futuras generaciones puedan programar la plataforma a su conveniencia, según el uso que le quieran dar, al igual que cualquier dispositivo similar en el mercado.

La programación de aplicaciones Android es un proceso que lleva tiempo perfeccionar, para así hacer un uso más óptimo de todas las capacidades de los dispositivos que corren bajo esta plataforma. Gracias al sistema operativo Android se pudo crear un sistema robótico totalmente funcional con tareas de asistencia personal, cuyo costo es nulo gracias a que esta plataforma es totalmente libre.

Las actuales capacidades de los dispositivos inteligentes, como el reconocimiento de órdenes por voz mediante la API de Google o el sistema de conversiones texto a voz que posee incluye el sistema operativo Android, brindan la oportunidad de crear aplicaciones

con mucha funcionalidad que pueden ser aplicadas prácticamente en cualquier ámbito o campo social.

Dentro del procesamiento de imágenes, OpenCV es una librería muy potente que pone a nuestra disposición una gran cantidad de algoritmos que pueden se implementados y usados como se crea conveniente. La integración de esta librería con el sistema operativo Android fue un proceso algo complejo, que dependía mucho de qué características de la librería se querían usar, la manera más fácil fue usar la Java API, sin embargo, no integra todos los métodos que posee OpenCV, así que se optó por la segunda manera que es hacer uso de librerías nativas mediante la JNI (Java Native Interface), lo que hace posible disponer de todos los métodos que esta librería posee, así como improvisar el rendimiento de la aplicación en general.

El reconocimiento de gestos que se realizan con las manos fue un elemento que llevó mucho tiempo de desarrollo, pero gracias a él se logró crear un medio de interacción con el asistente personal autónomo, que además permitió la creación de actividades y juegos basados en la realidad aumentada. Para mejorar la tasa de reconocimientos se necesitó de la combinación tanto de los momentos de Hu para la región convexa como para los de la aproximación poligonal, pues cada una nos entrega información importante acerca de la geometría de la mano. La aproximación poligonal y la región convexa, son algoritmos muy interesantes para el reconocimiento de patrones, que consume mucho menos recursos que las técnicas habituales.

Para el caso de la detección y reconocimiento de rostros se emplearon técnicas muy usadas actualmente, con buenos resultados en condiciones de iluminación controladas. Es de gran importancia dentro del reconocimiento de rostros crear un buen corpus de imágenes de los usuarios que se desea identificar, el mismo debe incluir muestras de los rostros con una correcta iluminación y a su vez con distintas expresiones faciales.

Uno de los factores con los que se tuvo más inconvenientes dentro del procesamiento de imágenes, tanto para el reconocimiento de gestos, como para el reconocimiento de objetos y rostros, son los posibles cambios en iluminación que pueden existir en el ambiente. Estos cambios afectan al proceso de reconocimiento en general, por cual fue necesario aplicar varias técnicas e incluso se propusieron algunos algoritmos para medir el nivel de iluminación dentro de una escena para así realizar las correcciones necesarias.

Las actividades interactivas basadas en la realidad aumentada se constituyen en una herramienta innovadora que se implementó durante el desarrollo de esta Tesis en teléfonos móviles inteligentes. Dicha aplicación permite demostrar las capacidades actuales de los mismos y tener con ellos un producto similar a los que se pueden encontrar en el mercado, con la gran diferencia que no hace falta equipos costosos, sino simplemente instalar una aplicación como cualquier otra dentro de un teléfono Android.

Actualmente en el mercado global existen varios autómatas dedicados a la asistencia personal cuyo costo es relativamente elevado, con este proyecto se pudo desarrollar e implementar un asistente personal único en su tipo, con un costo muy por debajo, equivalente a 600$ dólares americanos. Dicho sistema posee una gran cantidad de funciones y provee a las futuras generaciones una plataforma sobre la cual seguir realizando estudios en inteligencia artificial, además con este proyecto se obtuvo una herramienta ideal de asistencia para todas las áreas sociales, enfocada principalmente a la asistencia a personas y niños con discapacidad. Con las actividades que posee el sistema actualmente se brinda una herramienta de soporte para el proceso de rehabilitación de personas con discapacidad y sobre todo una herramienta para el desarrollo psicomotriz de niños con y sin discapacidad, mediante el uso de actividades lúdico-interactivas.

El desarrollo de este proyecto de Tesis tuvo un impacto muy grande dentro de la rama investigativa, abriendo nuevos caminos para futuros estudios, además en base a los resultados obtenidos en las pruebas de campo se demostró que es una herramienta ideal capaz de contribuir en gran medida en el área de discapacidad, pues se notó que era una herramienta ideal para trabajar con niños que posean síndromes de Down, TDAH y PCI. Con el tiempo se espera que el prototipo pueda seguir mejorando y así contribuyendo de manera significativa al desarrollo tecnológico de nuestro país.

La construcción del asistente personal autónomo bautizado “Robodroid” y posteriormente “Anbaro” fue un proceso que tomó mucho tiempo, periodo durante el cual se pudieron desarrollar nuevos avances e innovar en técnicas de visión por computador, lo que brinda nuevas pautas para las futuras generaciones para el desarrollo de sistemas de soporte y ayuda a las personas.

La combinación de la plataforma con el servidor de asistencia remota, crean una herramienta muy poderosa que brinda gran libertad al usuario, permitiendo que todo el conjunto puede ser aplicado prácticamente en cualquier ámbito social.

Como conclusión final se puede decir que los actuales dispositivos móviles inteligentes son sistemas muy avanzados que poseen un gran cantidad de recursos en hardware, que muchas veces son desperdiciados, cuando en la realidad pueden ser aplicados para crear sistemas muy complejos y sobre los cuales se puede generar y desarrollar nuevas tecnologías que contribuyan al desarrollo de una sociedad más equitativa, tal es el caso de nuestro “Robodroid”, que se constituye en un claro ejemplo de desarrollo en innovación tecnológica aplicada a la contribución de una mejor sociedad.

La plataforma mecánica que se desarrolló durante el transcurso de esta tesis posee todos los elementos necesarios tanto electrónicos como mecánicos, además de un protocolo de comunicación robusto, por lo que es recomendable utilizar dicha plataforma para desarrollar cualquier aplicación en lo que concierne a robótica según la necesidad o el problema que quiera solucionar el desarrollador.

Para mejorar las capacidades de asistencia personal y que pueda ser incluido en más campos, es factible desarrollar más funciones sobre las que ya posee actualmente, así como se podría mejorar las tareas de autonomía a fin de darle más naturalidad al autómata como tal.

Se podría seguir desarrollando en el ámbito de la reconstrucción bidimensional y combinarlo con técnicas de reconstrucción tridimensional, para que el asistente personal autónomo pueda desplacer sobre cualquier ambiente detectando los obstáculos que existan en su camino. Este punto también podría ser aplicado para que el asistente personal autónomo busque y captures objetos con su pinza en determinados lugares o ambientes.

Para el caso del reconocimiento de gestos u objetos se podrían reemplazar los momentos invariables de HU por los descriptores de Fourier que es una técnica más precisa, además es posible sustituir la distancia euclidiana por cualquier otra distancia que sea más sensible a pequeñas variaciones. Adicionalmente existen buenas técnicas que también podrían ser usadas para las distintas etapas de reconocimiento, tales como los filtros de HAAR, Sift y Surf, teniendo en consideración el consumo de recursos que implica el uso de cada uno de estos algoritmos.

Dentro del reconocimiento facial se podría utilizar algoritmos como LBP, que permite reconocer rostros en distintas condiciones de iluminación, así como utilizar un corpus más pequeño durante la fase de entrenamiento. Una de las desventajas que se debe considerar es que los filtros entrenados para reconocer rostros utilizando este algoritmo, implican un gran consumo de recursos de hardware.

Durante la aplicación de los algoritmos de aproximación poligonal y región convexa, se pudo observar que son técnicas muy interesantes y sobre las cuales se pueden realizar más estudios. Como se vio la combinación de estas dos técnicas da buenos resultados por lo cual se pueden crear algoritmos de reconocimiento más robustos, por tanto sería interesante aplicar dichas técnicas a otros conceptos.

Se pueden seguir realizando estudios para medir iluminación de un ambiente en base a los algoritmos que ya se han aplicado a lo largo de esta tesis, con el fin de poder crear un sistema más robusto ante los distintos cambios de iluminación.

Cuando se trabaje en el desarrollo de aplicaciones Android que involucren procesamiento de imágenes se debe considerar el consumo de CPU que implica cada algoritmo que se implemente, debido a que esto puede evocar en un mal funcionamiento de la aplicación final. Es recomendado escribir todo el código que corresponda al procesamiento de imágenes en código nativo(C/C++) con el fin de optimizar recursos lo máximo posible.

Finalmente el prototipo desarrollado en el transcurso de esta tesis es un sistema muy complejo, que brinda las herramientas para que futuros desarrolladores puedan continuar sus estudios en inteligencia artificial y en técnicas de visión por computador. Por tanto se recomienda que se siga trabajando e investigando en base al sistema ya desarrollado, con el fin de conseguir un sistema más completo, con más funciones y aplicabilidad que pueda contribuir de manera más significativa a la sociedad.

BIBLIOGRAFÍA

[1] J. R. Alejandro Rodríguez-Picavea, «Asociación Iniciativas y Estudios Sociales,» Mayo 2006. [En línea]. Available: http://www.asoc-ies.org/vidaindepen/docs/la_%20figura_del_asistente_personal_v1-1.pdf. [Último acceso: 4 3 2014].

[2] E. M. Mellado, «El Taller de mis Memorias,» 12 Diciembre 2013. [En línea]. Available: http://www.eltallerdemismemorias.com/2013/12/12/asistente-personal/. [Último acceso: 20 Enero 2014].

[3] Kioskea, «Kioskea,» Marzo 2014. [En línea]. Available: http://es.kioskea.net/contents/394-pda-organizador. [Último acceso: 1 Abril 2014].

[4] F. L. M. González, «Aplicaciones Para Dispositivos Móviles,» 2011. [En línea]. Available: http://riunet.upv.es/bitstream/handle/10251/11538/Memoria.pdf?sequence=1. [Último acceso: 2 4 2014].

[5] C. Tardáguila Moro, «Universidad Oberta de Catalunya,» 20 Noviembre 2009. [En línea]. Available: http://hdl.handle.net/10609/9164. [Último acceso: 13 Noviembre 2013].

[6] Área de Ingeniería de Sistemas y Automática, «Escuela Universitaria de Ingeniería Técnica Industrial de Zaragoza,» [En línea]. Available: http://automata.cps.unizar.es/Historia/Webs/automatas_en_la_historia.htm. [Último acceso: 15 Noviembre 2013].

[7] D. M. Chércoles, «Universidad de Valladolid,» Febrero 2001. [En línea]. Available: http://isa.uniovi.es/~gojea/funding/documentos/historia%20automatica.pdf. [Último acceso: 1 Marzo 2014].

[8] L. R. Noguez, «Micromegas,» Febrero 2007. [En línea]. Available: http://micromegas2.files.wordpress.com/2007/02/ruiz02.pdf. [Último acceso: 1 Abril 2014].

[9] J. D. G. Cobo, «Universidad Carlos III de Madrid,» [En línea]. Available: http://e-archivo.uc3m.es/bitstream/handle/10016/16336/TFG_Juan_Domingo_Galvez_Cobo.pdf?sequence=1. [Último acceso: 4 Enero 2014].

[10] X. Barandiaran, «Sin Dominio,» 16 Octubre 2003. [En línea]. Available: http://sindominio.net/~xabier/textos/biorob/biorob.pdf. [Último acceso: 1 Marzo 2014].

[11] C. D. V. Á. G. G. F. J. P. M. M. Juan Jesús Romero Cardalda, «Inteligencia Artificial y Computación Avanzada,» de *Inteligencia Artificial y Computación Avanzada*, Santiago de Compostela, Fundación Alfredo Brañas, 2007.

[12] I. F. A. M. Á. R. R. G. B. Arturo Baz Alonso, «Dispositivos móviles,» [En línea]. [Último acceso: 3 Abril 2014].

[13] «Index.es,» [En línea]. Available: http://motorola.index.es/subpaginas.php?a=813. [Último acceso: 2 Abril 2014].

[14] C. D. Falcón, «Universidad de la Palmas de Gran Canaria,» [En línea]. Available: http://www.iuma.ulpgc.es/~nunez/clases-micros-para-com/mpc1011-trabajos/mpc1011-Cassandra-Microprocesadores%20en%20SMARTPHONES.pdf. [Último acceso: 7 Abril 2014].

[15] Peter Rogers, Rhianna Brien , «Kantar_Worldpanel,» Kantar_Worldpanel, 2013. [En línea]. Available: http://www.kantarworldpanel.com/. [Último acceso: 8 Abril 2014].

[16] «Support Google,» Google, 2014. [En línea]. Available: https://support.google.com/websearch/answer/2819496?hl=es&ref_topic=4440405. [Último acceso: 8 Abril 2014].

[17] «Apple,» Apple, 2014. [En línea]. Available: http://www.apple.com/es/ios/siri/. [Último acceso: 7 Abril 2014].

[18] «Hello Indigo,» Artificial Solutions, 2013. [En línea]. Available: http://www.hello-indigo.com/. [Último acceso: 7 Abril 2014].

[19] «Intuitive Automata,» 2012. [En línea]. Available: http://www.intuitiveautomata.com/products.html. [Último acceso: 8 Abril 2014].

[20] «Robots for Humanity,» 24 Octubre 2013. [En línea]. Available: http://r4h.org/2013/10/24/body-surrogate-overview/. [Último acceso: 3 Abril 2014].

[21] «International Press Digital,» 2012 Octubre 2014. [En línea]. Available: http://espanol.ipcdigital.com/2012/10/06/docomo-desarrolla-robot-que-actua-como-asistente-personal/. [Último acceso: 10 Junio 2013].

[22] «Asimo Honda,» Honda, 2014. [En línea]. Available: http://asimo.honda.com/asimo-history/. [Último acceso: 2014 Abril 7].

[23] «Aldebaran,» [En línea]. Available: http://www.aldebaran.com/en/humanoid-robot/nao-robot. [Último acceso: 7 Abril 2014].

[24] «Aldebaran,» [En línea]. Available: http://www.aldebaran.com/en/humanoid-robot/nao-robot-working. [Último acceso: 8 Abril 2014].

[25] «Toyota,» Toyota, [En línea]. Available: http://www.toyota-global.com/innovation/partner_robot/family_2.html. [Último acceso: 2104 Abril 9].

[26] M. N. Marko Gargenta, Learning Android, United States of America: O'Reilly Media, Inc, 2014.

[27] «Dashboards,» Google, [En línea]. Available: http://developer.android.com/about/dashboards/index.html?utm_source=ausdroid.net. [Último acceso: 19 Abril 2014].

[28] M. Wolfson, Android Developers Tools Essential, California: O'Reilly Media, Inc., 2013.

[29] L. D. G. B. M. M. N. Zigurd Mednieks, Programing Android, California: O'Reilly Media, Inc., 2011.

[30] R. S. C. K. C. E. O. W. FRANK ABLESON, Android in Action, New York: Manning Publications Co., 2012.

[31] J. F. Dave Smith, Android Recipes, Apress, 2011.

[32] L. D. G. B. M. M. N. Zigurd Mednieks, Programming Android, California: O'Reilly Media, Inc, 2011.

[33] «Activity,» Google, 25 Abril 2014. [En línea]. Available: http://developer.android.com/reference/android/app/Activity.html. [Último acceso: 27 Abril 2014].

[34] G. Allen, Beginning Android 4, New York: Springer Science+Business Media, LLC., 2012.

[35] «Manifest Intro,» Google, [En línea]. Available: http://developer.android.com/guide/topics/manifest/manifest-intro.html. [Último acceso: 6 Mayo 2014].

[36] «Widget, LinearLayout,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/widget/LinearLayout.html. [Último acceso: 8 Mayo 2014].

[37] «Widget, FrameLayout,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/widget/FrameLayout.html. [Último acceso: 8 Mayo 2014].

[38] «Widget, Relative Layout,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/widget/RelativeLayout.html. [Último acceso: 8 mayo 2014].

[39] «View,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/view/View.html. [Último acceso: 9 Mayo 2014].

[40] «OS, Asyctask,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/os/AsyncTask.html. [Último acceso: 9 Mayo 2014].

[41] «View, SurfaceView,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/view/SurfaceView.html. [Último acceso: 9 Mayo 2014].

[42] «Graphics, Canvas,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/graphics/Canvas.html. [Último

acceso: 9 Mayo 2014].

[43] «Graphics, BitmapFactory,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/graphics/BitmapFactory.html. [Último acceso: 9 Mayo 2014].

[44] «Media, Media Player,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/media/MediaPlayer.html. [Último acceso: 12 Mayo 2014].

[45] «Media, SoundPool,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/media/SoundPool.html. [Último acceso: 12 Mayo 2014].

[46] «Speech, RecognizerIntent,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/speech/RecognizerIntent.html#ACTION_RECOGNIZE_SPEECH. [Último acceso: 13 Mayo 2014].

[47] «Media, FaceDetectos.Face,» Google, 2 Mayo 2014. [En línea]. Available: http://developer.android.com/reference/android/media/FaceDetector.Face.html. [Último acceso: 13 mayo 2014].

[48] R. Khatko, «Implementing Face Detection in Android,» Intel, 10 Octubre 2013. [En línea]. Available: https://software.intel.com/en-us/blogs/2013/10/28/implementing-face-detection-in-android. [Último acceso: 13 Mayo 2014].

[49] «Conectivity, Bluetooth,» Google, [En línea]. Available: http://developer.android.com/guide/topics/connectivity/bluetooth.html#ManagingAConnection. [Último acceso: 13 Mayo 2014].

[50] R. Szeliski, Computer Vision:Algorithms and Applications, Springer, 2010.

[51] A. K. Gary Bradski, Lerning OpenCV: Computer Vision with OpenCV Library, California: O'Reilly Media, Inc, 2008.

[52] D. Mery, Visi¶on por Computador, Santiago de Chile: Departamento de Ciencia de la Computación Universidad Católica de Chile, 2004.

[53] D. G. M. Gómez, «El Ojo Humano,» Horus, 29 Noviembre 2011. [En línea]. Available: http://www.horusgo.com/elojo.htm. [Último acceso: 1 Julio 2014].

[54] P. A. Alegre, «Asociación Pro Ayuda a No Videntes,» 2012. [En línea]. Available: http://www.apanovi.org.ar/iusaludparyfun.html. [Último acceso: 1 Julio 2014].

[55] OpenCV, «About,» OpenCV, 2014. [En línea]. Available: http://opencv.org/about.html. [Último acceso: 1 Julio 2014].

[56] J. Howse, Android Application Programming with OpenCV, Birmingham: Packt Publishing Ltd., 2013.

[57] O. Cinar, Pro Android C++ with the NDK, New York: Springer Science+Business Media, 2012.

[58] «OpenCV4Android SDK,» OpenCV, 21 Abril 2014. [En línea]. Available: http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/O4A_SDK.html. [Último acceso: 10 Junio 2014].

[59] «Miscellaneous Image Transformations,» OpenCV, 2014. [En línea]. Available: http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html. [Último acceso: 5 Julio 2014].

[60] «Changing Colorspaces,» OpenCV, 2014. [En línea]. Available: http://docs.opencv.org/trunk/doc/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html. [Último acceso: 5 Julio 2014].

[61] J. Jones, «Quick Color Guide,» NetSource technologies, 2011. [En línea]. Available: http://www.netsourceinc.com/blog/quick-color-guide. [Último acceso: 7 Julio 2014].

[62] «Adaptive Coloring for Syntax Highlighting,» Nokia, 2008. [En línea]. Available: http://doc.qt.digia.com/qq/qq26-adaptivecoloring.html. [Último acceso: 5 Julio 2014].

[63] A. Mozas, «La Bitácora Industrial,» 20 Marzo 2012. [En línea]. Available: http://disenoypreimpresionmozadr.wordpress.com/2012/03/20/traductor-universal-de-color-espacio-cielab/. [Último acceso: 6 Julio 2014].

[64] «Erosion an Dilating,» OpenCV, 2014. [En línea]. Available: http://docs.opencv.org/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html. [Último acceso: 4 Julio 2014].

[65] «Basic Thresholding Operations,» OpenCV, 2014. [En línea]. Available: http://docs.opencv.org/doc/tutorials/imgproc/threshold/threshold.html. [Último acceso: 5 Julio 2014].

[66] «Canny Edge Detector,» OpenCV, 2014. [En línea]. Available: http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html. [Último acceso: 4 Julio 2014].

[67] «Métrica Eucledian,» EcuRed, [En línea]. Available: http://www.ecured.cu/index.php/M%C3%A9trica_euclideana. [Último acceso: 8 Julio 2014].

[68] «Face Detection using Haar Cascades,» OpenCV, 25 Junio 2014. [En línea]. Available: http://docs.opencv.org/trunk/doc/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html. [Último acceso: 26 Mayo 2014].

[69] «Face Recognition with OpenCV,» OpenCV, 21 Abril 2014. [En línea]. Available: http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html. [Último acceso: 27 Mayo 2014].

[70] J. P. H. J. K. Peter N. Belhumeur, «UCSD Computer Vision,» Julio 1997. [En línea]. Available: http://vision.ucsd.edu/kriegman-grp/papers/pami97.pdf. [Último acceso: 10 Julio 2014].

[71] «FaceRecognizer,» OpenCV, 27 Mayo 2014. [En línea]. Available: http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec_api.html. [Último acceso: 27 Mayo 2014].

[72] «Cascade Classifier Training,» OpenCV, 21 Abril 2014. [En línea]. Available: http://docs.opencv.org/doc/user_guide/ug_traincascade.html. [Último acceso: 28 Mayo 2014].

[73] «Huawei Ideos X1 U8180,» Smart GSM, [En línea]. Available: http://www.smart-gsm.com/moviles/huawei-ideos-x1-u8180. [Último acceso: 25 Septiembre 2014].

[74] «Samsung Galaxy S III mini,» Smart GSM, [En línea]. Available: http://www.smart-gsm.com/moviles/samsung-galaxy-s3-mini-value-edition. [Último acceso: 25 Septiembre 2014].

[75] «Samsung Galaxy S III,» Smart GSM, [En línea]. Available: http://www.smart-gsm.com/moviles/samsung-galaxy-s3. [Último acceso: 25 Septiembre 2014].

[76] «Samsung Galaxy Note 3,» Smart GSM, [En línea]. Available: http://www.smart-gsm.com/moviles/samsung-galaxy-note-3. [Último acceso: 25 Septiembre 2014].

[77] «Samsung GALAXY Tab 2 (7.0),» Smart GSM, [En línea]. Available: http://www.smart-gsm.com/moviles/samsung-galaxy-tab-2. [Último acceso: 2014 Septiembre 2014].

[78] «Samsung Galaxy Tab 2 (10.1),» Smart GSM, [En línea]. Available: http://www.smart-gsm.com/moviles/samsung-galaxy-tab-2-101. [Último acceso: 25 Septiembre 2014].

[79] «Samnsung Galaxy Tab pro 10.1,» Smart Gsm, [En línea]. Available: http://www.smart-gsm.com/moviles/samsung-galaxy-tabpro-101#. [Último acceso: 4 Octubre 2014].

[80] «LG Nexus 4,» Smart Gsm, [En línea]. Available: http://www.smart-gsm.com/moviles/lg-nexus-4. [Último acceso: 4 Octubre 2014].

[81] «EXP TECH,» [En línea]. Available: http://www.exp-tech.de/service/datasheet/HC-Serial-Bluetooth-Products.pdf. [Último acceso: 6 Agosto 2014].

[82] «Wavesen,» [En línea]. Available: http://www.wavesen.com/probig.asp?id=24. [Último acceso: 6 Agosto 2014].

[83] J. S. O. Zambrano, «Bluetooth Terminal,» Google Play, [En línea]. Available: https://play.google.com/store/apps/details?id=ptah.apps.bluetoothterminal. [Último acceso: 2014 Noviembre 25].

[84] «Power HD Standard Servo 3001HB,» Pololu Corporation, [En línea]. Available: http://www.pololu.com/product/1058/specs. [Último acceso: 23 Septiembre 2014].

[85] ServoDatabase, «Servomotores de rotación continua GWS S35 STD,» ServoDatbase, [En línea]. Available: http://www.servodatabase.com/servo/hitec/hs-755hb. [Último acceso: 2014 Septiembre 24].

[86] «Power HD Continuous Rotation Servo AR-3606HB,» Pololu, [En línea]. Available: http://www.pololu.com/product/2149. [Último acceso: 24 Sepetiembre 2014].

[87] «Pololu,» [En línea]. Available: http://www.pololu.com/file/0J37/ssc03a_guide.pdf. [Último acceso: 24 Septiembre 2014].

[88] «Elec-Freaks,» [En línea]. Available: http://www.elecfreaks.com/store/infrared-proximity-sensor-long-range-sharp-gp2y0a02yk0f-p-451.html?zenid=72a12747e6880f84a0170dfcbb8b9608. [Último acceso: 2014 Septiembre 30].

[89] «Erasme,» [En línea]. Available: http://www.erasme.org/IMG/gp2y0a02_e.pdf. [Último acceso: 24 Septiempre 2014].

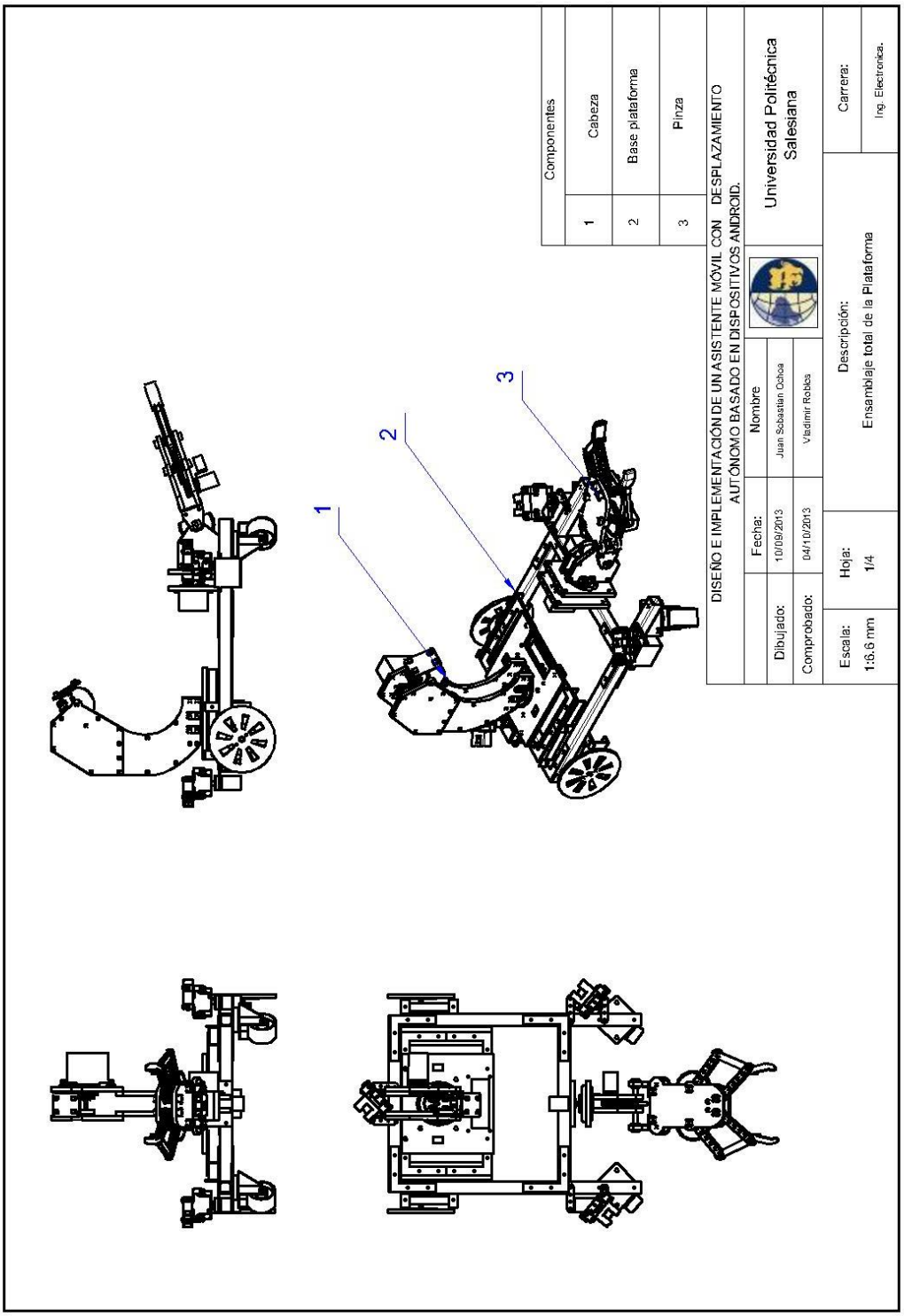
[90] «Hoskin,» [En línea]. Available: http://www.hoskin.qc.ca/uploadpdf/Instrumentation/Tekscan/hoskin_FlexiForce4120f35db4439.pdf. [Último acceso: 24 Septiembre 2014].

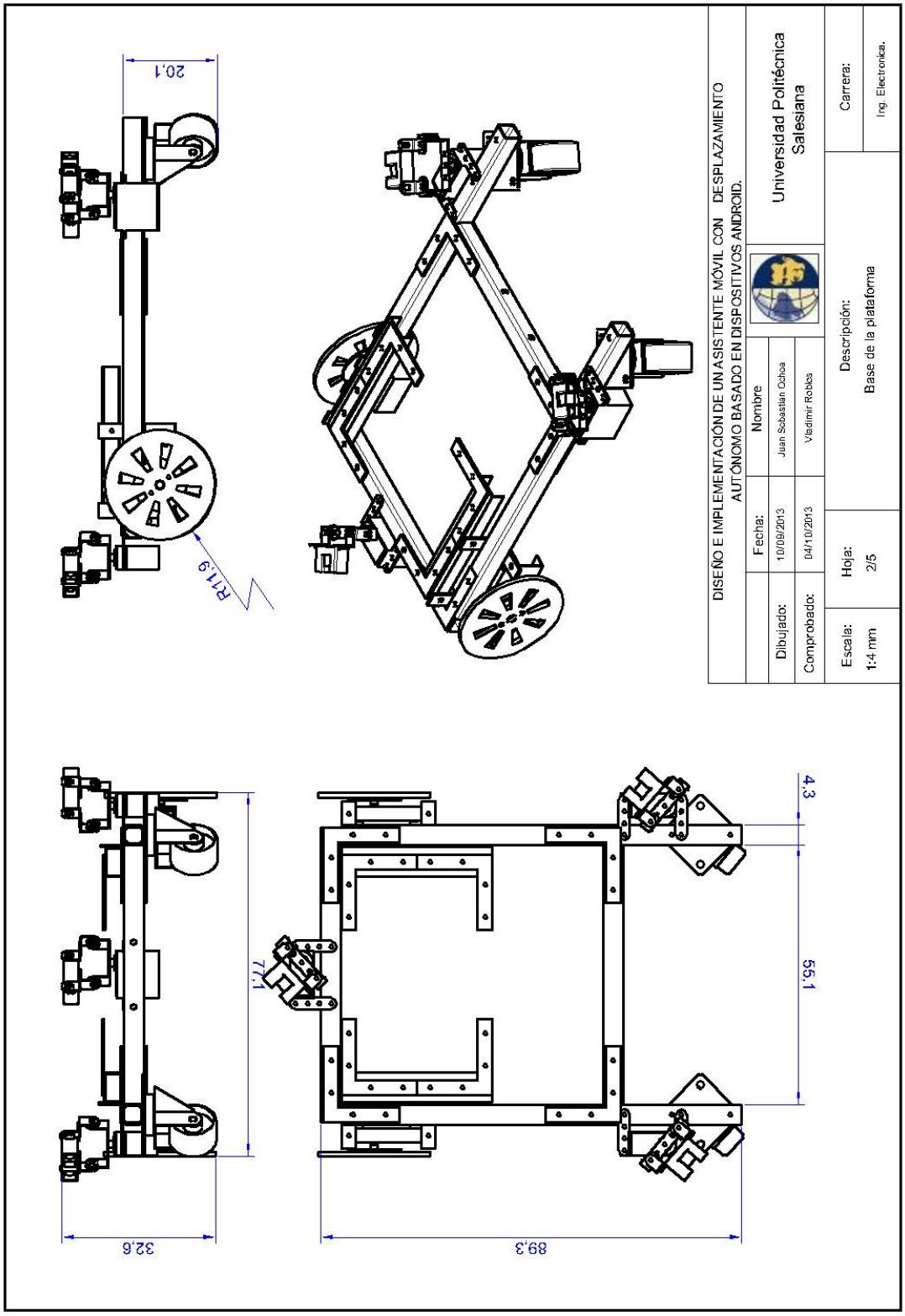
[91] «Microchip,» [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf. [Último acceso: 24 Septiembre 2014].

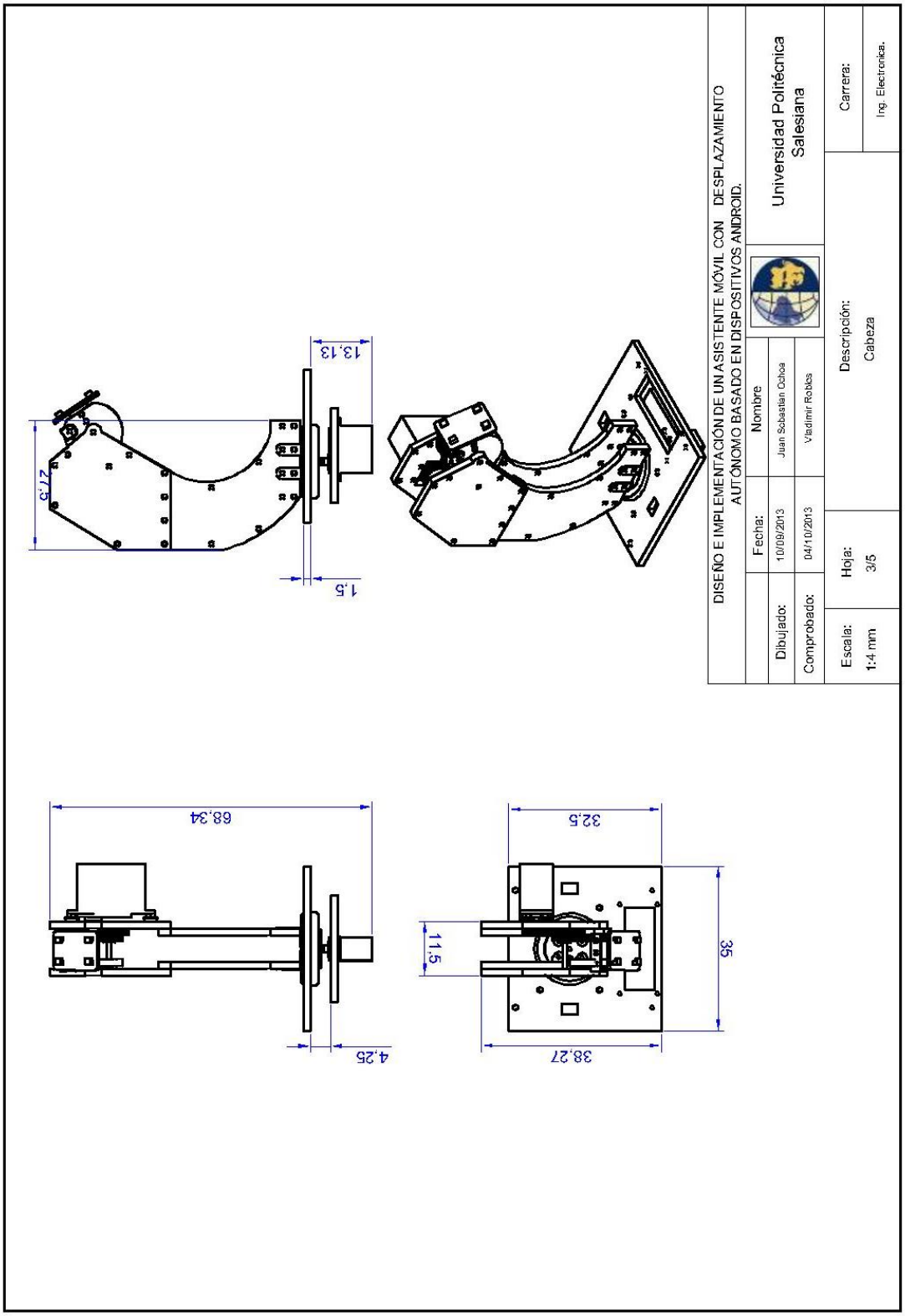
[92] E. G. Breijo, Compilador C CCS y Simulador Proteus para Microcontroladores PIC, Mexico D.F: Alfaomega, 2008.

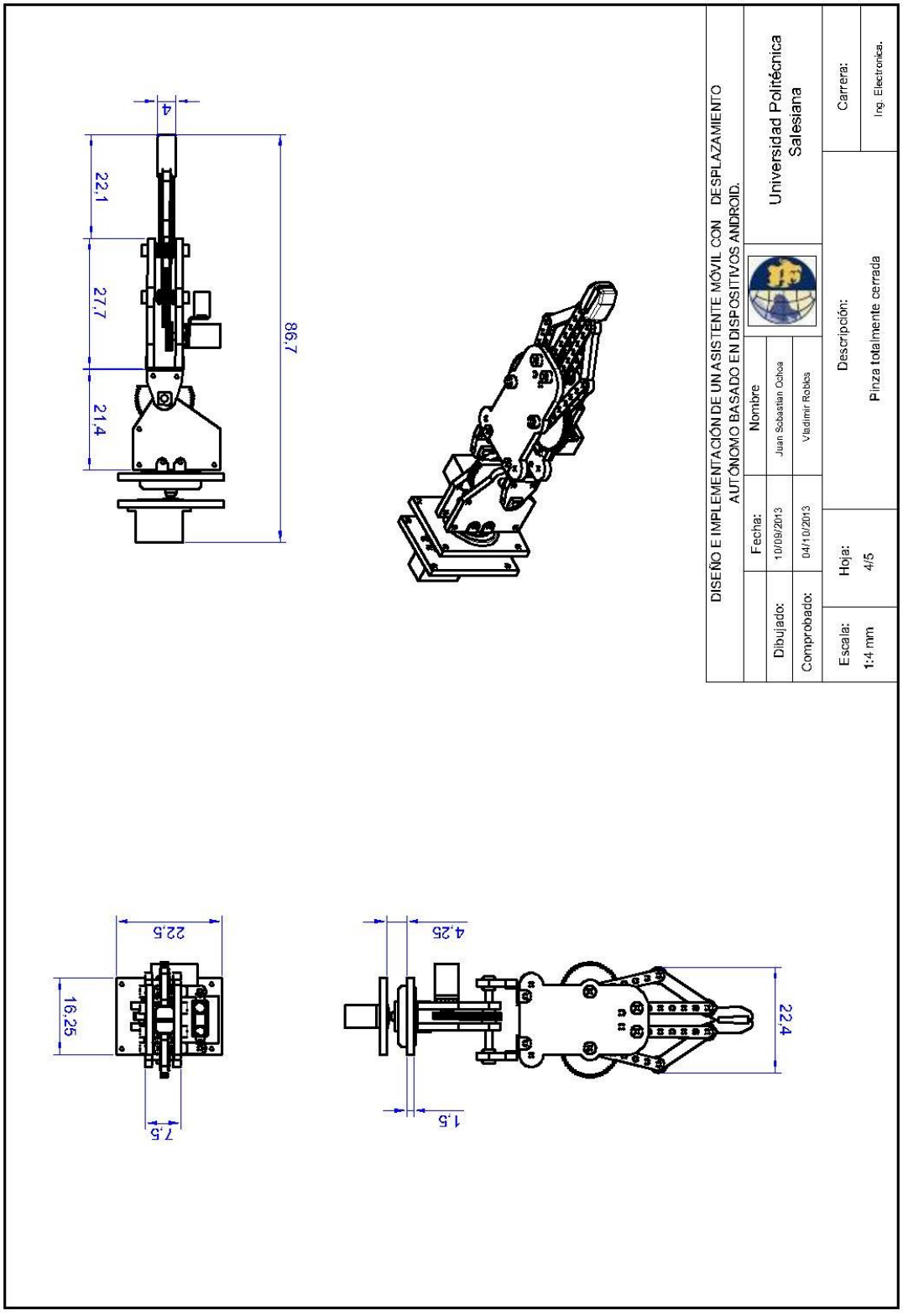
[93] V. E. L. N. M. P. A. J. J. G. D. J. &. O.-Z. J. Robles Bykbaev, «Modelling domain knowledge of Speech and Language Therapy with an OWL ontology and OpenEHR archetypes,» de *8th International Conference on Health Informatics (HEALTHINF)*, Lisbon, Portugal, To apper in 2015.

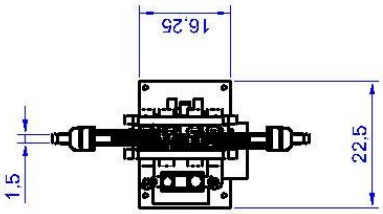
Anexo 1.- Diseños de la plataforma

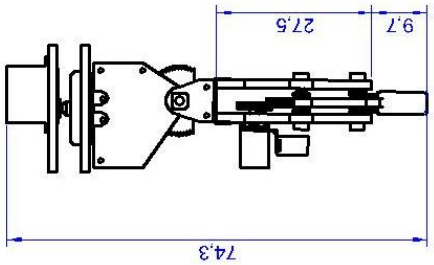


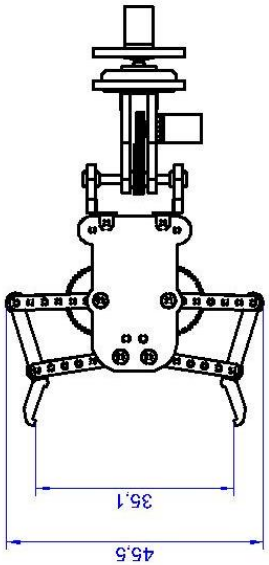

20.1
R11.9
32.6
77.1
89.3
4.3
55.1
DISEÑO E IMPLEMENTACIÓN DE UN ASISTENTE MÓVIL CON DESPLAZAMIENTO AUTÓNOMO BASADO EN DISPOSITIVOS ANDROID.
Universidad Politécnica Salesiana
Nombre
Fecha:
Dibujado:
10/09/2013
Juan Sebastian Cohoa
Comprobado:
04/10/2013
Vladimir Robles
Descripción:
Base de la plataforma
Carrera:
Ing. Electronica.
Escala:
1:4 mm
Hoja:
2/5

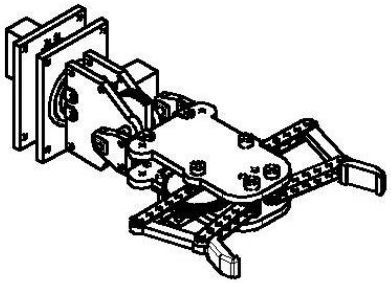

DISEÑO E IMPLEMENTACIÓN DE UN ASISTENTE MÓVIL CON DESPLAZAMIENTO AUTÓNOMO BASADO EN DISPOSITIVOS ANDROID.
Fecha:
Nombre
Dibujado:
10/09/2013
Juan Sebastian Ochoa
Comprobado:
04/10/2013
Vladimir Robles
Universidad Politécnica Salesiana
Escala:
1:4 mm
Hoja:
3/5
Descripción:
Cabeza
Carrera:
Ing. Electronica.
13.13
27.3
1.5
68.34
4.25
32.5
11.5
35
38.27


4
22.1
27.7
21.4
86.7
22.5
16.25
7.5
4.25
1.5
22.4
DISEÑO E IMPLEMENTACIÓN DE UN ASISTENTE MÓVIL CON DESPLAZAMIENTO AUTÓNOMO BASADO EN DISPOSITIVOS ANDROID.
Fecha:
Nombre
Dibujado:
10/09/2013
Juan Sebastian Ochoa
Comprobado:
04/10/2013
Vladimir Robles
Universidad Politécnica Salesiana
Escala:
1:4 mm
Hoja:
4/5
Descripción:
Pinza totalmente cerrada
Carrera:
Ing. Electronica.



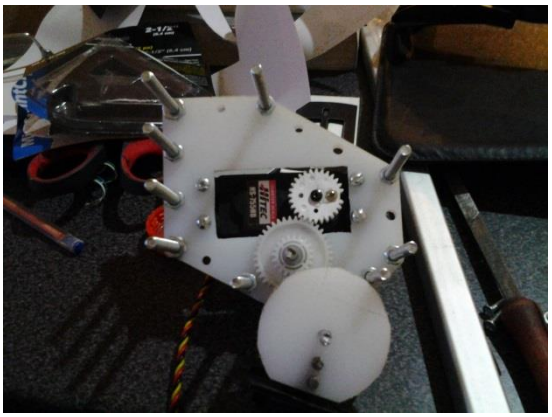


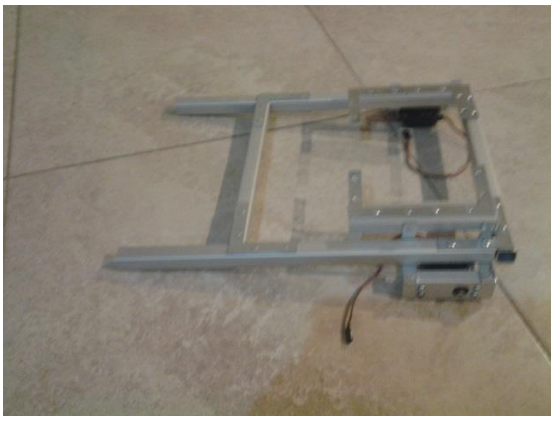


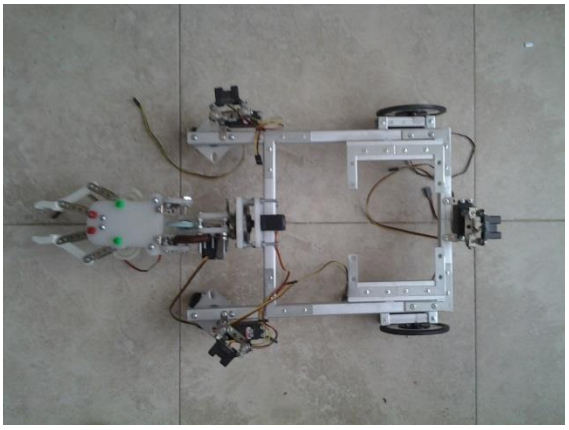


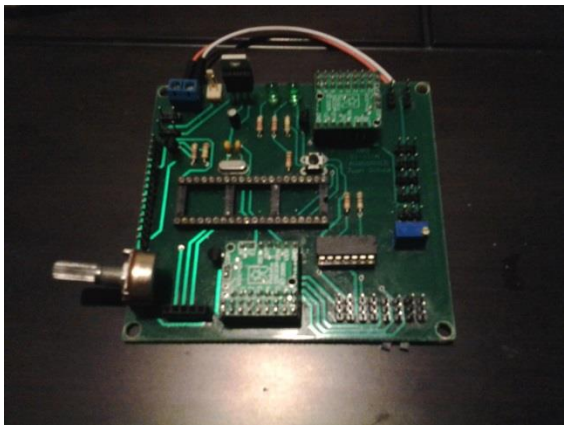
DISEÑO E IMPLEMENTACIÓN DE UN ASISTENTE MÓVIL CON DESPLAZAMIENTO AUTÓNOMO BASADO EN DISPOSITIVOS ANDROID.				
	Fecha:	Nombre		Universidad Politécnica Salesiana
Dibujado:	10/09/2013	Juan Sebastian Ochoa		
Comprobado:	04/10/2013	Vladimir Robles		
Escala: 1:4 mm	Hoja: 5/5	Descripción: Pinza totalmente abierta		Carrera: Ing. Electronica.

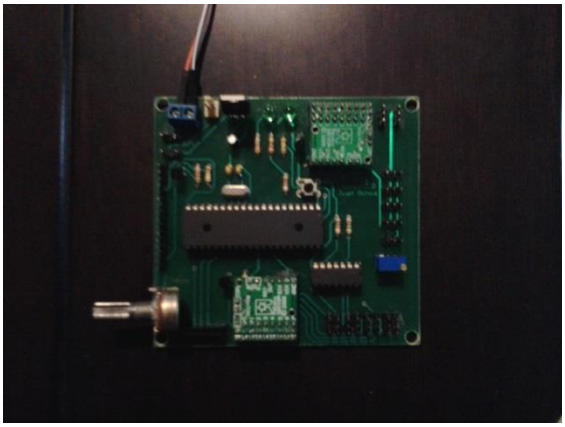
Anexo 2.- Fotografías de la construcción de la plataforma

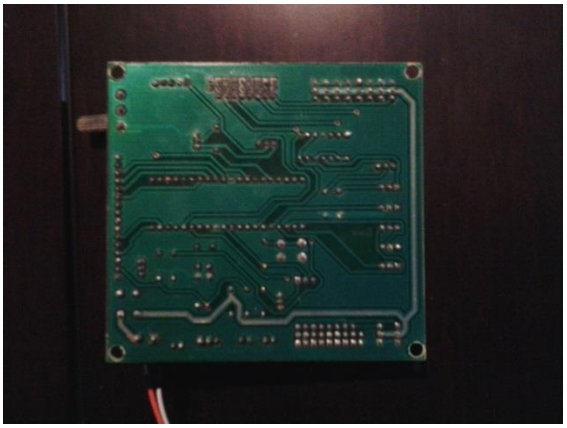


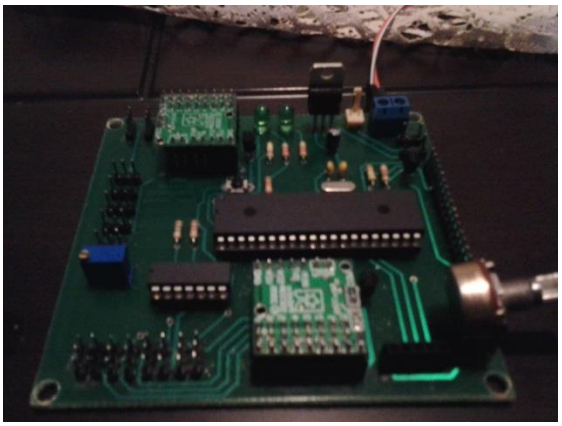












Anexo 3.- Tablas de las pruebas realizadas para verificar protocolo de comunicación y el funcionamiento de toda la plataforma

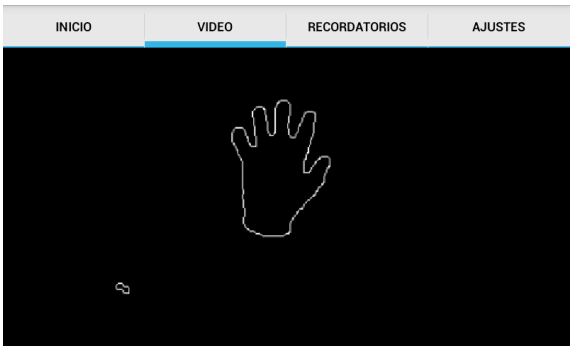
Evaluación del protocolo de comunicación y de la plataforma usando la aplicación "TestProtocolo"											
Comandos enviados(Bytes)										Respuesta Esperada	
Sincro	Comando	Data									
1	2	3	4	5	6	7	8	9	10	Correcto	Incorrecto
100	1	1	0							X	
255	1	1	0							X	
255	1	2	0							X	
255	1	3	0							X	
255	1	1	1							X	
255	1	1	2							X	
255	1	1	3							X	
255	1	1	4							X	
255	1	1	5							X	
100	4	3	7							X	
255	1	100	1							X	
255	4	1	1							X	
255	3	1	1							X	
255	1	2	1							X	
255	1	1								X	
255	1	2								X	
100	1	1								X	
255	1	10								X	
255	3	1								X	
255	4	1								X	
255	6	1								X	
255	2	20	20	20	20	20	20	20	20	X	
255	2	140	30	40	50	60	70	80	90	X	
255	2	20	50	50	60	40	30	20	50	X	
255	2	20	50	50	60	40	30	20	50	X	
255	2	200	200	240	255	1	2	3	6	X	
255	2	35	45	55	65	75	85	95	105	X	
100	2	60	30	40	50	60	70	80	90	X	
255	2	60	30	40	50	60	70	80	90	X	
255	2	60	30	40	50	60	70	80	90	X	
255	2	127	127	20	127	20	127	127	20	X	
255	2	128	127	30	128	127	30	128	128	X	
255	3	10	10	10	10	10	10	10	10	X	
255	3	10	10	60	10	10	10	10	10	X	
255	3	60	10	60	10	10	10	10	10	X	
252	3	60	90	60	10	90	180	10	10	X	

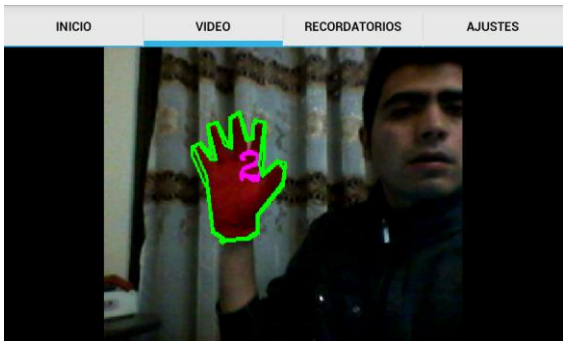
255	3	60	90	60	10	90	180	10	10	X	
255	3	10	10	180	30					X	
155	200	10	20							X	
200	3	150	50	200	10	90	10	10	160	X	
255	3	180	50	60	20	30	60	70	180	X	
255	4	1	50	1	50					X	
255	4	0	50	0	50					X	
255	4	2	100	1	100					X	
255	4	5	100	1	100					X	
255	4	3	100	3	100					X	
255	4	1	100	1						X	
255	4	2	50							X	
255	4	1	100	1	100					X	
255	4	1	100	1	100	90	10			X	
255	4	2	100	1	50					X	
155	4	1	100	1	100					X	
255	4	1	100	1	100					X	
255	4	120	1	1	50					X	
255	4	2	100	1	50					X	
255	4	3	100	1	50					X	
100	4	1	100	1	100					X	
255	4	1	100	1	100					X	
255	4	5	50	5	50					X	
255	4	1	200	1	50					X	
255	5	1								X	
255	5	2								X	
255	5	3								X	
255	5	4								X	
255	5	5								X	
255	5	4	1							X	
255	5	3								X	
155	5	4								X	
255	5	5								X	
255	5	3								X	
100	5	3								X	
255	5									X	
255	5	3	1	1						X	
100	5	7								X	
255	6									X	
255	6	1								X	
155	6									X	
255	6	50								X	
255	7									X	

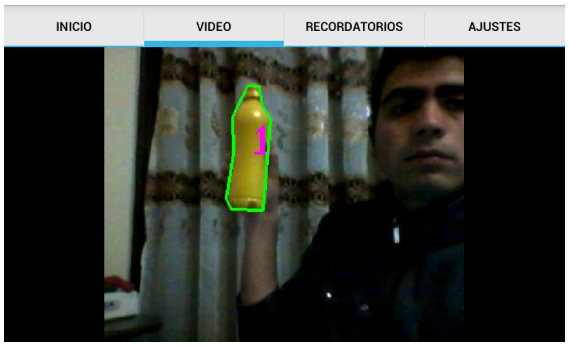
Evaluación del protocolo de comunicación y de la plataforma usando la aplicación "Robodroid"		
Elemento a evaluar	**Respuesta**	
	Si	**No**
¿Al presionar el botón "Encender", de la sección de comandos, la LCD de la plataforma se encendió?	X	
¿Al presionar el botón "Apagar" de la sección de comandos, la LCD se apagó?	X	
¿Al presionar el botón "Conf. Servos" en la sección de comandos, la plataforma mostro las configuraciones de velocidad de los 8 servomotores en la LCD?	X	
¿Al presionar el botón "Pos. Servos" en la sección de comandos, la plataforma mostro las posiciones de los 8 servomotores en la LCD?	X	
¿Al presionar el botón "Motores" en la sección de comandos, la plataforma mostro las configuraciones de velocidad y dirección de los 2 servomotores de rotación continua en la LCD?	X	
¿Al presionar el botón "Sensores" en la sección de comandos, la plataforma mostro las lecturas de los 5 sensores en la LCD?	X	
¿Cuándo se presiona el botón "Automove" cuando el *Automove* está inactivo, este se activa?	X	
¿Cuándo se presiona el botón "Automove" cuando el *Automove* está activo, este se desactiva?	X	
¿Al presionar el botón "Lec. Sensores" de la sección Solicitar Datos, la plataforma devuelve la lectura correcta de los sensores?	X	
¿Al presionar el botón "Pos. Servos" de la sección Solicitar Datos, la plataforma devuelve la posición actual de los 8 servomotores?	X	
¿Al presionar el botón "Conf. Servos" de la sección Solicitar Datos, la plataforma devuelve la configuración de velocidad de los 8 servomotores?	X	
¿Al presionar el botón "Conf. Motores" se la sección Solicitar Datos, la plataforma devuelve las configuraciones de velocidad y dirección de los dos servomotores de rotación continua?	x	
¿Al presionar el botón "Limpiar" limpia los datos que recibe de la plataforma?	X	
¿Cuándo se presiona el botón "Configurar Servos", se configura la velocidad ingresada para los 8 servomotores?	X	
¿Cuándo cambia cualquiera de los valores de las 8 barras de progreso, las posiciones de cada uno de los 8 servomotores correspondientes cambia acorde a dichos valores?	X	
¿Cuándo el Automove está activo los valores de posición enviados por las 3 barras de progreso de los servos 6, 7 y 8, son inhabilitados por la plataforma?	X	
¿Al presionar el botón "Adelante" la plataforma se desplaza hacia adelante con la velocidad ingresada?	X	
¿Al presionar el botón "reversa" la plataforma se desplaza hacia atrás con la velocidad ingresada?	X	
¿Al presionar el botón "izquierda" la plataforma gira hacia la izquierda con la velocidad ingresada para cada motor?	X	
¿Al presionar el botón "Derecha" la plataforma gira hacia la derecha con la	X	

velocidad ingresada para cada motor?		
¿Al presionar el botón “Parar” la plataforma se detiene?	X	
¿Cuándo el automove está activo, las lecturas y la gráfica del barrido que realizan los 3 sensores infrarrojos de distancia es la correcta?	X	
¿Cuándo la plataforma pierde la conexión Bluetooth con el Smartphone la plataforma detiene el proceso que esté realizando?	X	

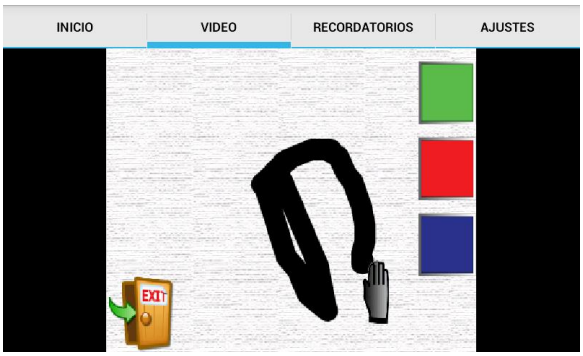
Anexo 4.- Capturas de Aplicación en funcionamiento

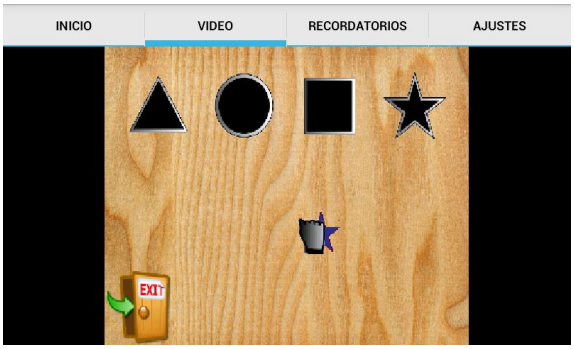


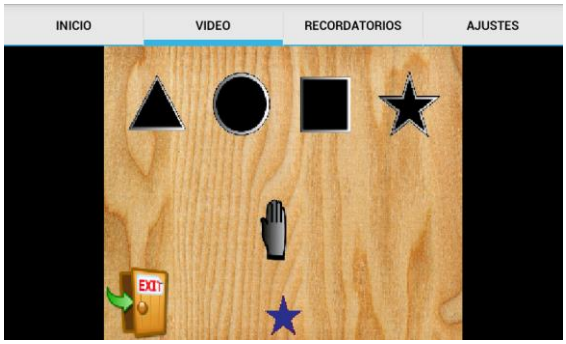


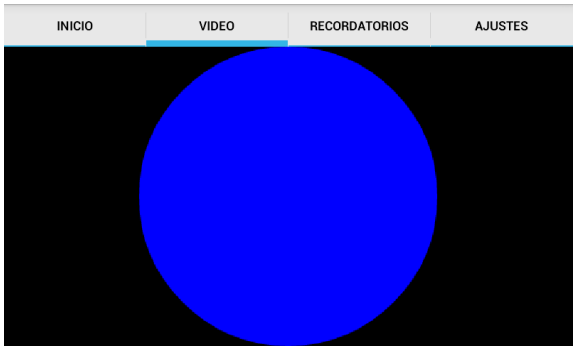


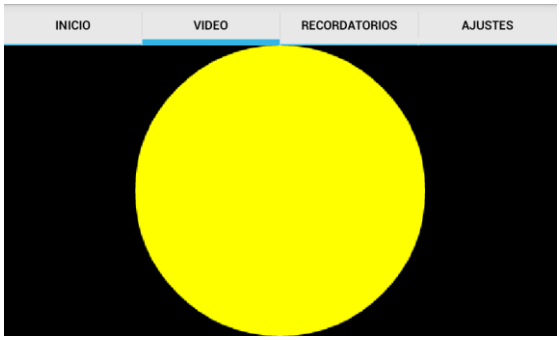

INICIO
VIDEO
RECORDATORIOS
AJUSTES
EXIT

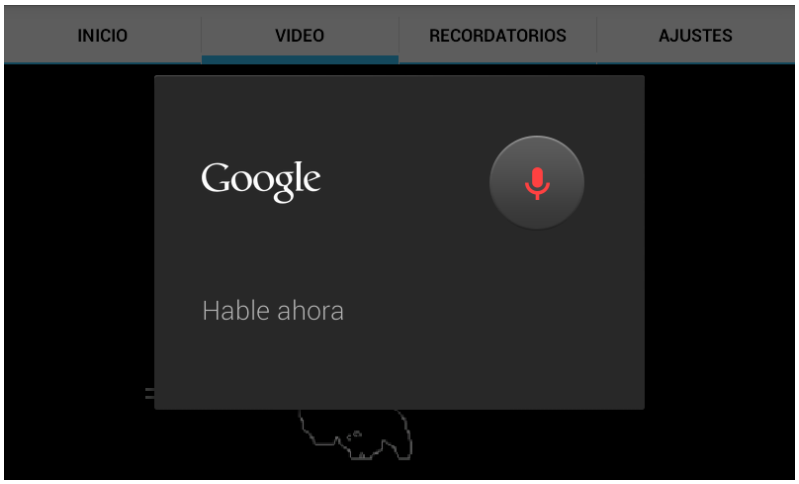

INICIO
VIDEO
RECORDATORIOS
AJUSTES
EXIT

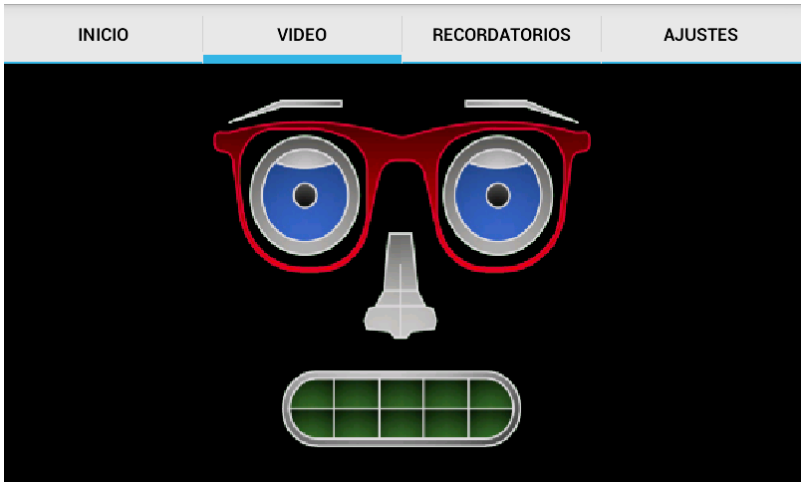

INICIO
VIDEO
RECORDATORIOS
AJUSTES
EXIT


INICIO
VIDEO
RECORDATORIOS
AJUSTES
EXIT


INICIO
VIDEO
RECORDATORIOS
AJUSTES


INICIO
VIDEO
RECORDATORIOS
AJUSTES


INICIO
VIDEO
RECORDATORIOS
AJUSTES
Google
Hable ahora


INICIO
VIDEO
RECORDATORIOS
AJUSTES

Anexo 5.- Evaluaciones tomadas en las pruebas Teta

Formulario de Evaluación y Validación del Asistente Personal Autónomo				
Evaluación del Aspecto estético de la Aplicación				
N.-	Elemento a evaluar	Valoración		Observaciones
		Sí	No	
1	¿El icono de la aplicación le parece adecuado?			
2	¿Las combinaciones de colores de la aplicación le parece adecuada?			
3	¿Los elementos se encuentran bien distribuidos?			
4	¿La distribución de los elementos de la aplicación en las pestañas le parece óptima?			
5	¿Le parece fácil de Manejar?			
6	¿Es fácil desplazarse a lo largo de los componentes de la aplicación?			
7	¿Es fácil comprender la función de cada componente?			

Formulario de Evaluación y Validación del Asistente Personal Autónomo				
Evaluación del Avatar Animado				
N.-	Elemento a evaluar	Valoración		Observaciones
		Sí	No	
1	¿El Avatar le parece Amigable?			
2	¿Los gráficos usados para la creación del Avatar le parecen Adecuados?			
3	¿Los colores del Avatar le parecen adecuados?			
4	¿Cree que las expresiones faciales que posee el Avatar son suficientes?			
5	¿El parpadeo y el movimiento de la cejas le brinda mayor naturalidad?			
6	¿Cuándo el Avatar lee un texto, da una verdadera impresión de que este se encuentra hablando?			

Formulario de Evaluación y Validación del Asistente Personal Autónomo				
Evaluación de Funcionamiento				
N.-	Elemento a evaluar	Valoración		Observaciones
		Sí	No	
1	¿Al seleccionar una pestaña cualquiera, la pantalla muestra la ventana correspondiente?			
2	¿Los dispositivos Bluetooth vinculados se muestran listados			

	correctamente en el Spinner correspondiente?			
3	¿Al intentar conectarse a un dispositivo Bluetooth, lo hace correctamente?			
4	¿Se conecta correctamente al servidor de Asistencia Remota con la IP indicada?			
5	¿El correo electrónico y la contraseña del usuario se almacenan correctamente?			
6	¿Los recordatorios se almacenan correctamente, con la hora y fecha indicada?			
7	¿Los recordatorios son leídos en la hora y fecha oportuna?			
8	¿La lectura de correos electrónicos se efectúa exitosamente?			
9	¿Se puede ajustar la segmentación por color, para cada etapa de reconocimiento de manera adecuada?			
10	¿Las Actividades de Autonomía se ejecutan de acuerdo a lo previsto?			
11	¿Al activar el botón “S. gestos” se muestra la segmentación por color correspondiente?			
12	¿Al activar el botón “I. Gestos” se muestra la identificación de gestos que realiza?			
13	¿Al activar el botón “S. Objetos” se muestra la segmentación por color correspondiente?			
14	¿Al activar el botón “I. Objetos” se muestra la identificación de objetos que realiza?			
15	¿Al Activar el botón “Avatar” la aplicación muestra el Avatar Interactivo?			
16	¿Al Activar el botón “Juego” se muestra el juego correspondiente?			
17	¿Al Activar el botón “Pintar” se muestra el juego correspondiente?			
18	¿Al Activar el botón “Círculos” se muestra el juego correspondiente?			
19	¿Al Activar el botón “Rec. Rostros” se muestra el proceso de identificación y reconocimiento de rostros?			
20	¿Al activar el botón “Seguir Mano”, el asistente personal sigue la mano del usuario?			
21	¿Al activar el botón “Seguir Persona”, el asistente personal sigue el rostro el rostro de la persona que se indica en el Spinner correspondiente?			
22	¿Al activar el botón “Seguir Objeto”, el asistente personal sigue el objeto que esta seleccionado en el Spinner correspondiente?			
23	¿Al desactivar el botón “Autonomía” las actividades de autonomía dejan de estar activas?			

Formulario de Evaluación y Validación del Asistente Personal Autónomo				
Evaluación de interacción con el Usuario mediante la Visión Artificial				
N. -	Elemento a evaluar	Valoración		Observaciones
		Sí	No	
1	¿El Asistente Personal sigue correctamente la mano del usuario?			
2	¿El Asistente Personal sigue correctamente el rostro del usuario?			
3	¿El Asistente personal sigue correctamente el objeto que muestra el usuario?			
4	¿Al realizar con la mano el gesto 1 inmediatamente seguido por el 4, se lanza el reconocimiento por voz?			
5	¿Se puede jugar de manera intuitiva dentro de los juegos interactivos, utilizando los gestos que se realizan con la mano?			
6	¿El desplazamiento del cursor dentro de los juegos interactivos es proporcional al desplazamiento de la mano del usuario en el aire?			
7	¿Se puede salir de los juegos realizando el gesto número 4 sobre el icono correspondiente?			
8	¿Mientras el Asistente personal, reproduce un sonido o se encuentra hablando, es posible detener dichos proceso realizando el gesto numero 1 inmediatamente seguido por el gesto numero 4?			

Formulario de Evaluación y Validación del Asistente Personal Autónomo				
Evaluación de interacción con el Usuario mediante Reconocimiento de Ordenes por Voz				
N. -	Elemento a evaluar	Valoración		Observaciones
		Sí	No	
1	¿Al pronunciar la orden “Saluda” el Asistente Personal se presenta?			
2	¿Al pronunciar la orden “Funciones” el Asistente Personal pronuncia algunas de sus funciones?			
3	¿Al pronunciar la orden “Futuro” el Asistente Personal habla acerca de su futuro?			
4	¿Al pronunciar la orden “Cantar” el Asistente Personal reproduce una canción?			
5	¿Al pronunciar la orden “Juego” el Asistente Personal lanza el juego interactivo numero 1?			
6	¿Al pronunciar la orden “Pintar” el Asistente Personal lanza el juego interactivo numero 2?			
7	¿Al pronunciar la orden “Juego” el Asistente Personal lanza el juego interactivo numero 3?			
8	¿Al pronunciar la orden “Cuéntame un Cuento” el Asistente Personal procede a la lectura de un cuento almacenado en su			

	base de datos?			
9	¿Al pronunciar la orden “Seguir Mano” el Asistente Personal procede a seguir la mano del usuario?			
10	¿Al pronunciar la orden “Seguir Cara” el Asistente Personal procede a identificar y seguir el rostro de cualquier usuario?			
11	¿Al pronunciar la orden “Seguir Juan” el Asistente Personal procede a identificar y seguir el rostro del usuario Juan?			
12	¿Al pronunciar la orden “Seguir Vladimir” el Asistente Personal procede a identificar y seguir el rostro del usuario Vladimir?			
13	¿Al pronunciar la orden “Seguir Luis” el Asistente Personal procede a identificar y seguir el rostro del usuario Luis?			
14	¿Al pronunciar la orden “Seguir Salomé” el Asistente Personal procede a identificar y seguir el rostro del usuario Salomé?			
15	¿Al pronunciar la orden “Seguir Botella” el Asistente Personal procede a seguir el objeto correspondiente?			
16	¿Al pronunciar la orden “Seguir Lata” el Asistente Personal procede a seguir el objeto correspondiente?			
17	¿Al pronunciar la orden “Seguir Pelota” el Asistente Personal procede a seguir el objeto correspondiente?			

Formulario de Evaluación y Validación del Asistente Personal Autónomo				
Evaluación del Servidor de Asistencia Remota				
N.-	Elemento a evaluar	Valoración		Observaciones
		Sí	No	
1	¿Los parámetros de conexión del servidor de Asistencia Remota se muestran correctamente?			
2	¿Se puede controlar adecuadamente todos los motores y servomotores desde el servidor de Asistencia Remota?			
3	¿Se puede controlar todas las funciones y actividades que posee el Asistente Personal desde el servidor de Asistencia Remota?			
4	¿Se puede ejecutar la lectura de cualquier texto desde el servidor de Asistencia Remota?			
5	¿El servidor de Asistencia remota muestra el texto recuperado del Reconocimiento de órdenes por voz lanzado previamente en el Asistente Personal?			

Anexo 6.- Evaluación realizada en las Pruebas Zeta

FORMULARIO DE EVALUACION DE DESEMPEÑO DEL ASISTENTE PERSONAL AUTONOMO "ROBODROID" EN EL PROCESO DE TERAPIA.

Por favor responda las siguientes preguntas, en base a lo observado durante la terapia con y sin el Asistente Personal Autónomo.

1. **¿Piensa usted que el sistema de Asistencia Personal puede contribuir al proceso de rehabilitación de los niños con discapacidad?**

☐ Sí ☐ No

¿Por qué?_____
_____
_____

2. **¿La terapia con el Asistente Personal Autónomo se llevó con mayor rapidez?**

☐ Sí ☐ No

¿Por qué?_____
_____
_____

3. **En el caso de que la respuesta anterior sea afirmativa, indique un porcentaje aproximado de 0 a 100% de tiempo que Usted considera que se ha ahorrado.**

_____
_____

4. **¿Considera usted que el Asistente personal fue factor que modificó cierto tipo de conductas de los niños durante la terapia?**

☐ Sí ☐ No

¿Por qué?_____
_____
_____

5. **¿El nivel de atención de los niños mejoró con la presencia del Asistente Personal Autónomo?**

☐ Sí ☐ No

¿Por qué?_____
_____
_____

6. **¿Cree que la Asistencia Remota para el control del autómata es útil para mejorar las actividades con los niños con discapacidad?**

☐ Sí ☐ No

¿Por qué?_____
_____
_____

7. **¿Las actividades que posee el autómata son correctas para trabajar con grupos de**

niños con diferentes necesidades educativas especiales asociadas a una discapacidad?

☐ Sí ☐ No

¿Por qué?_____
_____
_____

8. **¿Considera que las actividades basadas en la realidad pueden contribuir al desarrollo psico-motriz de los niños?**

☐ Sí ☐ No

¿Por qué?_____
_____
_____

9. **¿En qué niños se apreció un cambio en la conducta con la presencia del Asistente Personal Autónomo?**

_____
_____
_____

10. **¿En que considera que se puede mejorar? O ¿Qué actividades se podrían implementar en el autómata?**

_____
_____
_____

11. **¿Qué actividades que posee el autómata considera usted que puede contribuir más al proceso de rehabilitación y para qué discapacidad?**

_____
_____
_____

Anexo 7.- Terapia aplicada a los niños de la Fundación "Jesús para los Niños", durante las pruebas Zeta

Hola niños, mi nombre es Robodroid y el día de hoy van a jugar conmigo.
¿Pero primero cuéntenme como estuvieron sus vacaciones?
¿Se fueron de paseo?
¿Durmieron Bastante?
¿Están contentos de regresar a la escuelita?
Ahora vamos a ver si tenemos nuevos compañeros!! Vamos a saludar a los compañeros que teníamos y a los nuevos compañeros
#canción 8
Muy bien..!! Ahora vamos a ver si se acuerdan de algunos juegos, porque yo les dije que hoy vamos a jugar. Cuando yo muestre el círculo rojo levantan las manos, si muestro el circulo azul bajan las manos, y finalmente con el amarillo todos a correr.
#juego3
Muy bien..!! Hemos jugado y vamos a descansar, quiero que duerman, quiero que todos ronquen...
#sonido 4
Qué lindo que he dormido vamos a despertar..
¿Selena estas despierta?
¿María José estas despierta?
¿Juan José estas despierto?
¿Ana Luisa estas despierta?
¿Johnny estas despierto?
¿Jhuliana estas despierta?
¿Carolina estas despierta?
¿Kevin estas despierto?
Todos a despertar, todos a despertar que les voy a contar un cuento.!!
¿Quieren que les cuente un cuento? Muy bien, pero todos vamos a estar atentos y a escuchar el cuento
#cuento
¿Les gusto el cuento?
#Preguntas sobre el cuento
Muy Bien..!!, Ahora vamos a bailar todos al ritmo de la música y hacemos un ruedo alrededor de mí.
#canción 1
Muy bien un aplauso para ustedes, ha sido un gusto trabajar con ustedes Ahora vengan para que se despidan de mí, denme la mano por favor.

Anexo 8.- Fotografías de la pruebas realizadas en la Fundación “Jesús para los Niños”

Anexo 9.- Certificados de las pruebas realizadas en la Fundación "Jesús para los Niños"

ESCUELA ESPECIAL
"JESUS PARA LOS NIÑOS"

Ministerio de Educac

Cañar 20 de noviembre de 2014

La Escuela de Educación Básica Especial Jesús para los Niños del cantón Cañar certifica que el estudiante Juan Sebastián Ochoa de la Universidad Politécnica Salesiana realizo sus prácticas con el "Asistente Personal Autónomo basado en dispositivos móviles Android" con los niños de la institución, mimas que se realizaron en primera instancia con terapias de actividades iníciales utilizando los métodos tradicionales, para luego realizar el abordaje utilizando el asistente personal autónomo, pudiendo evidenciar el nivel de participación y de atención de los niños en las diferentes actividades que se utilizó el asistente, el personal docente y técnico de la institución participo y, observo los beneficios que implica el trabajar en el área de discapacidad con este tipo de herramientas, las prácticas fueron realizadas el 9 de septiembre del 2014. Es necesario anotar la importancia que tiene el desarrollar este tipo de herramientas que van a facilitar a las personas con discapacidad actividades que promuevan su funcionalidad.

Mgst. Ma. Augusta Zambrano C.
LEDER DE LA E.E.E.B.J.P.L.N.

ESCUELA ESPECIAL JESÚS PARA LOS NIÑOS MIRAME SOY PARTE DE TU MUNDO CAÑAR

Cañar, Capital Arqueológica y Cultural del Ecuador
Dirección: Panamericana Norte y Paseo de los Cañarís
Telf: 072236796 E-mail: fundacionjpn@hotmail.com

ESCUELA ESPECIAL
"JESUS PARA LOS NIÑOS"

Ministerio de Educac

Cañar 20 de noviembre de 2014

Previa convocatoria realizada por pare de la Líder de la Escuela de Educación Básica Especial Jesús para los Niños del cantón Cañar, a los padres de familia a una reunión el día 17 de septiembre del 2014, entre uno de sus puntos consta "Proyecto Universidad Politécnica Salesiana", se da a conocer sobre las actividades que el Señor Juan Sebastián Ochoa realizo el día 9 de septiembre en donde se observo como los niños y jóvenes reaccionaron con las actividades propuestas con el Asistente Personal Autónomo basado en dispositivos móviles Android" y los beneficios que implico al trabajar utilizando este tipo de herramientas, autorizando utilizar las fotos y el video realizado en las diferentes actividades.

Mgst. Ma. Augusta Zambrano O
LIDER DE LA E.E.B.J.P.L.N.

ESCUELA ESPECIAL JESÚS PARA LOS NIÑOS
MIRAME SOY PARTE DE TU MUNDO
CAÑAR

ministerio de educación

ESCUELA ESPECIAL "JESÚS PARA LOS NIÑOS

Dirección: Panamericana S/n y Paseo de los cañarís

TELF: 2 236 796

FUNDACION JESUS PARA LOS NIÑOS MIRAME SOY PARTE DE TU MUNDO CAÑAR

CONVOCATORIA

Se cita a los padres de familia de la escuela especial "Jesús para los Niños" a una reunión a efectuarse el día 17 de septiembre del 2014 a partir de las 7h30 en el local de la Institución con el siguiente orden del día:

- Lectura y aprobación del acta anterior
- Elección de la nueva directiva de padres de familia
- Informe de la dirección
- Elección del CAE
- Proyecto Universidad Politécnica Salesiana
- Asuntos varios

NOMBRES Y APELLIDOS /ALUMNO	FIRMA DEL REPRESENTANTE	Nº DE CEDULA/REPRESENTANTE
Acero Buñay Lizbeth Noemí		0302716089
Ávila Dután Silvia Juliana		
Buñay Guaman Juan José		
Caguana Tenelema Paola Azucena		0301196762
Castillo Mayllazhungo Ana Cristina		0302146691
Cunin Santander Janeth Carolina		0301992012
Dután Guasco Segundo Juan		0302857719
Dután Guasco Karla Marisol		0302857719
Dután Sumba Emilio Josué		
Encalada Bernal Bryan Esteban		0302600622
Guaman Guaman Tannya Alexandra		
Juncal Sarmiento Jostin Alexis		
Loja Guaman Estafania Abigail		
Mainato Tenesaca Fanny Alexandra		0302709035
Mayancela Loja William Gregorio		0301038287
Murudumbay Tenelema Joselyn Domenica		
Narváez Ojeda Jackson Patricio		
Nivelo Ayavaca Ana Luisa		

ministerio de educación

ESCUELA ESPECIAL "JESÚS PARA LOS NIÑOS

Dirección: Panamericana S/n y Paseo de los cañarís

TELF: 2 236 796

Ojeda Tomas Roxana Maribel		030 2388533.
Pallchizaca Tenezaca Luis Alfredo		0301178711
Padilla Paramo Ligia Elizabeth		0301735247
Patiño Neira Blanca Johanna		
Patiño Siguencia Kevin Andrés		
Pichizaca Muñoz María José		0302604871
Pillaga Verdugo Johnny Franklin		0302567052
Quishpe Guaman Juan Jesús		030268906 -2
Seguin Chuqui Natali Silvana		0301152724
Serrano Serrano Omar Mauricio		0301212486
Siguencia Muñoz Jessica Alexandra		030279 4292
Siguencia Narváez Jennifer Carolina		0300856689
Simbaña Muyulema Selena Angeles		0301950788
Tenezaca Acero Kevin José		0302904990
Tenesaca Paucar Pedro Freddy		030260632 -2
Tenezaca Pino Cristopher Leonardo		0303085245
Valdés Andrade Diego Eduardo		030254 6042.
Verdugo Cadena Cesar Rafael		
Villavicencio Rojas Cristina Michell		
Zhirvi Méndez Juan Diego		

Lcda. Raquel Bernal

SECRETARIA DEL COMITÉ CENTRAL

DE PADRES DE FAMILIA

ESCUELA ESPECIAL JESÚS PARA LOS NIÑOS MIRAME SOY PARTE DE TU MUNDO CAÑAR