SISTEMA DE VISIÓN ROBÓTICA PARA DISPOSITIVOS MÓVILES

Laboratorio de Instrumentación Virtual y Robótica Aplicada

Departamento de Ingeniería Electrónica y Computación

Facultad de Ingeniería

Universidad Nacional de Mar del Plata

Gelosi, Iván Exequiel

Contacto: egelosi@fi.mdp.edu.ar

Director: Fernández, Juana Graciela

Co-Director: Rivera, Raúl Rubén

Febrero 2017

Resumen

En este proyecto se realizó un diseño e implementación de un sistema robótico dotado de visión por computadora, capaz de detectar y seguir un objeto a elegir. El diseño del robot fue realizado mediante una placa Arduino.

Dado que el procesamiento digital de imágenes, requiere de una gran capacidad de memoria y potencia de cálculos, se optó por la implementación de dispositivos móviles con sistema operativo Android, en la cual se realiza todo el procesamiento y se entrega la información a la placa Arduino. Para el procesamiento se adoptó la librería para procesamiento de imágenes OpenCv4Android.

Como medio de comunicación entre dispositivos se optó utilizar la comunicación Bluetooth. Con este, el dispositivo móvil entrega la información de comandos, que luego la placa deberá interpretar y actuar.

Además de poseer visión, el robot fue dotado de un sensor de proximidad, para evitar obstáculos y encontrar el objeto deseado.

Luego, con fines de tener acceso a la cámara del dispositivo móvil y control del robot, se realizó una pagina web la cual adquiere imágenes y envía información de control mediante Wi-Fi.

Palabras clave: *Robótica, Arduino, Android, Páginas WEB, Sistemas autónomos*

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Continuamente se pueden observar los avances de la tecnología orientada al desarrollo de vehículos autónomos, y entre ellos se le ha dado mucha importancia a la visión por computador. Hoy en día, inclusive los coches poseen sensores y cámaras para la asistencia al estacionar, o para prevenir accidentes. Por lo cual la importancia de la optimización de la visión por computador incrementa cada día. Para que un vehículo sea autónomo, es imprescindible la detección de objetos u obstáculos, traking de objetos, y hasta reconocimiento de personas. La motivación principal que empuja este proyecto es la necesidad de unir la robótica y la visión por computador para crear sistemas autónomos más eficientes que puedan operar en determinados entornos, realizando tareas concretas.

En base a ello, se partió de la idea de crear un dispositivo capaz de realizar dichas tareas. Para la realización se debió seleccionar la plataforma adecuada, y dadas las características de la placa Arduino, se optó por su utilización, siendo una placa electrónica de código abierto que permite implementar de forma sencilla el control de motores y sensores. Esta tiene una gran versatilidad y con gran cantidad de librerías con mucho potencial.

Por otro lado, se necesitará una gran velocidad de procesamiento y memoria de almacenamiento para la realización de las tareas. Con dichas consideraciones y el conocimiento previamente adquirido sobre la programación para dispositivos Android se lo consideró la mejor opción. También se debe considerar que la programación en Android, siendo orientada a objetos, facilita enormemente las tareas a realizar por tener una gran cantidad de librerías. Por último, para aumentar el control y observar la tarea que se encuentra realizando el robot,

se decidió agregar una aplicación WEB que aumente la funcionalidad y permita observar a distancia el funcionamiento del robot. Dentro de dicha aplicación se permitirá iniciar y detener el movimiento y observar las imágenes que se ven en la aplicación Android.

1.2. Definición del problema

El Laboratorio de Instrumentación Virtual y Robótica Aplicada (L.I.V.R.A.), ámbito en el cual se desarrolla el trabajo, posee diversas líneas de investigación, dentro de las cuales se encuentran sensores, adquisición, procesamiento de señales e interconexión de redes. Las experiencia lograda por los integrantes del Grupo de Investigación y Desarrollo en Instrumentación Virtual (G.I.D.I.V.) en estas áreas durante los últimos 18 años han permitido proponer un Proyecto de Investigación, en el ámbito de la **UNMdP**, para el bienio 2015-2016 en el cual confluyen todas estas disciplinas: "Instrumentación Virtual: Estudio y desarrollo de interfases avanzadas orientadas a Sistemas de Robótica".

La Robótica es una de las áreas con mayor crecimiento en la ingeniería. Existen todo tipo de robots que operan en distintos ambientes con diferentes comportamientos y objetivos. Los desarrolladores son, entre otros, Ingenieros Electrónicos y en Computación, cuya actividad es el diseño de prototipos de robots, integrando tecnologías de sensores y actuadores, algoritmos de cálculo y comunicación, sistemas autónomos inteligentes y el hardware integrado. [[1] - [11]]

Por otra parte debe mencionarse el continuo desarrollo de los vehículos autónomos que ha otorgado una mayor importancia a los sistemas de visión por computadora. Muchos modelos de automóviles ya incluyen sensores y cámaras para la prevención de accidentes y choques, o incluso para asistencia al conductor en el estacionamiento. En síntesis, en los sistemas autónomos son imprescindibles los métodos de detección y seguimiento de objetos, ya que estos se utilizan para evitar accidentes, reconocer objetivos o efectuar tareas de monitoreo dinámicas. Paralelamente, ha surgido en los últimos años un creciente interés por la introducción de la robótica en la educación. Este fenómeno se debe en parte a la aparición de distintas plataformas robóticas de bajo costo que permite acercarla a un tipo de público que antes no tenía acceso a ella.

Este proyecto une la robótica y la visión por computadora para crear sistemas autónomos más efectivos que puedan operar en determinados entornos realizando tareas predeterminadas.

Se plantea diseñar un sistema que una la flexibidad de una plataforma robótica con la potencia de un dispositivo móvil. Esto implica trabajar con tecnologías diferentes en un mismo sistema: Dispositivos móviles con Android y Arduino. Por lo tanto, el robot realizará la detección y el seguimiento de un objetivo mediante algoritmos de visión por computador, y deberá ser capaz de alternar distintos comportamientos para desplazarse hacia su objetivo evitando los obstáculos cuando sea necesario.

1.3. Objetivos del proyecto

1.3.1. Objetivos Generales

El objetivo general es implementar un sistema robótico que sea capaz de utilizar la cámara de un dispositivo móvil equipado con un sistema operativo Android como sistema de visión artificial para guiar su movimiento hacia un objetivo (que puede o no hallarse en movimiento) en un determinado entorno, con el propósito de cumplir con diversas tareas en ambientes industriales y experimentación en ámbitos académicos.

1.3.2. Objetivos particulares

1. Estudio de las características del vehículo robótico.
2. Estudio de las técnicas que se emplean en la visión por computadora: procesamiento de imágenes, detección de características, reconocimiento de objetos y análisis de movimientos.
3. Desarrollar librerías para detección y seguimiento de objetos desde una señal de vídeo.
4. Desarrollar comandos para el control del vehículo robótico.
5. Estudio de requerimientos en aplicaciones de robótica industrial.
6. Desarrollo de algoritmos para la carga y procesamiento de mapas de navegación.
7. Estudio de interfaces de comunicación entre el vehículo robótico y el dispositivo móvil.
8. Pruebas de funcionamiento del sistema completo en ambientes industriales y académicos.

Capítulo 2

Estado del arte

2.1. Robótica

2.1.1. Introducción a la robótica

En 1979, el *Robot Institute of America* definía un robot como: *"Un manipulador reprogramable y multifuncional diseñado para trasladar materiales, piezas, herramientas o aparatos específicos a través de una serie de movimientos programados para llevar a cabo una variedad de tareas"*. Veinte años mas tarde, Microsoft dio la definición: *"Maquina controlada por computadora y programada para moverse, manipular objetos y realizar trabajos a la vez que interacciona con su entorno. Los robot son capaces de realizar tareas repetitivas de forma más rápidas, barata y precisa que los seres humanos"*. El cambio más sustancial incorporado al concepto en estos veinte años, es el hecho de realizar una interacción del robot con su entorno, que permite comportamientos adaptativos e inteligentes.
Se considera a un robot como un agente autónomo inteligente (AAI) cuando cumple los siguientes requisitos:

1. **Autonomía:** El sistema de navegación reside en la propia máquina, que debe operar sin conexión física a equipos externos.
2. **Inteligencia:** El robot posee capacidad de razonar hasta el punto de ser capaz de tomar sus propias decisiones y de seleccionar, fusionar e integrar las medidas de sus sensores.

2.1.2. Clasificación de robótica por cronología

1. *Manipuladores:* Son sistemas mecánicos multifuncionales con un sencillo sistema de control, bien manual, de secuencia fija o de secuencia variable.

Figura 2.1: Robot de primera generación

2. *Robots de aprendizaje:* Repiten una secuencia de movimientos que ha sido ejecutada previamente por un operador humano. El modo de hacerlo es a través de un dispositivo mecánico. El operador realiza los movimientos requeridos mientras el robot le sigue y los memoriza.

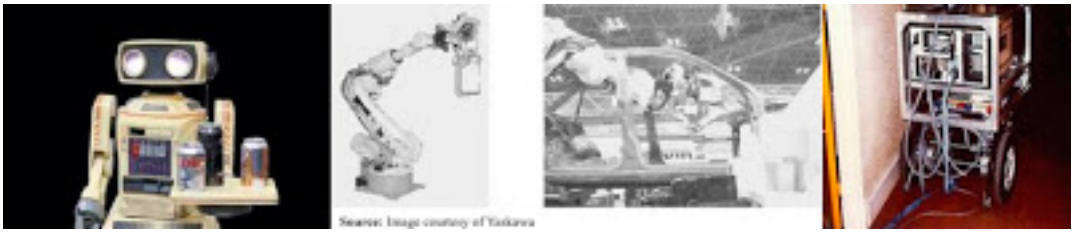

Figura 2.2: Robot de segunda generación

3. *Robots con control sensorizado:* El controlador es una computadora que ejecuta las órdenes de un programa y las envía al manipulador para que realice los movimientos necesarios.

Figura 2.3: Robot de tercera generación

4. *Robots inteligentes:* Son similares a los anteriores, pero además poseen sensores que envían información a la computadora de control sobre el estado del proceso. Esto permite una toma inteligente de decisiones y el control del proceso en tiempo real.

Figura 2.4: Robot de cuarta generación

2.1.3. Clasificación de robótica por arquitectura

1. **Poliarticulados:**
   En este grupo se encuentran los Robots de muy diversa forma y configuración, cuya característica común es la de ser básicamente sedentarios (aunque excepcionalmente pueden ser guiados para efectuar desplazamientos limitados) y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas, y con un número limitado de grados de libertad. En este grupo, se encuentran los manipuladores, los Robots industriales, los Robots cartesianos y se emplean cuando es preciso abarcar una zona de trabajo relativamente amplia o alargada, actuar sobre objetos con un plano de simetría vertical o reducir el espacio ocupado en el suelo.

Figura 2.5: Ejemplo de robot poliarticulado

2. **Móviles**:
   Son Robots con gran capacidad de desplazamiento, basados en carros o plataformas y dotados de un sistema locomotor de tipo rodante. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus sensores. Estos Robots aseguran el transporte de piezas de un punto a otro de una cadena de fabricación. Guiados mediante pistas materializadas a través de la radiación electromagnética de circuitos empotrados en el suelo, o a través de bandas detectadas fotoeléctricamente, pueden incluso llegar a sortear obstáculos y están dotados de un nivel relativamente elevado de inteligencia.

Figura 2.6: Ejemplo de robot móvil

3. **Androides:**

Son Robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano. Actualmente, los androides son todavía dispositivos muy poco evolucionados y sin utilidad práctica, y destinados, fundamentalmente, al estudio y experimentación. Uno de los aspectos más complejos de estos Robots, y sobre el que se centra la mayoría de los trabajos, es el de la locomoción bípeda. En este caso, el principal problema es controlar dinámica y coordinadamente en el tiempo real el proceso y mantener simultáneamente el equilibrio del Robot.

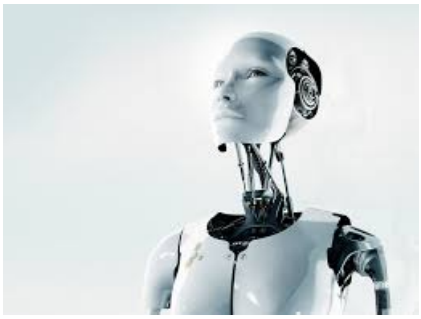

Figura 2.7: Ejemplo de robot androide

4. **Zoomórficos:**

Los Robots zoomórficos, que considerados en sentido no restrictivo podrían incluir también a los androides, constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos. A pesar de la disparidad morfológica de sus posibles sistemas de locomoción es conveniente agrupar a los Robots zoomórficos en dos categorías principales: caminadores y no caminadores. El grupo de los Robots zoomórficos no caminadores está muy poco evolucionado. Los experimentos efectuados en Japón basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de un movimiento relativo de rotación. Los Robots zoomórficos caminadores multípedos son muy numerosos y están siendo objeto de experimentos en diversos laboratorios con vistas al desarrollo posterior de verdaderos vehículos terrenos, pilotados o autónomos, capaces de evolucionar en superficies muy accidentadas. Las aplicaciones de estos Robots serán interesantes en el campo de la exploración espacial y en el estudio de los volcanes.

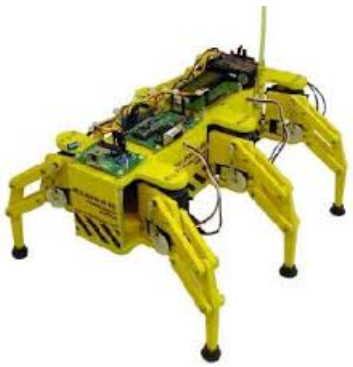

Figura 2.8: Ejemplo de robot zoomórfico

5. **Híbridos:**

Corresponden a aquellos de difícil clasificación, cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas, bien sea por conjunción o por yuxtaposición. Por ejemplo, un dispositivo segmentado articulado y con ruedas, es al mismo tiempo, uno de los atributos de los Robots móviles y de los Robots zoomórficos.

Figura 2.9: Ejemplo de robot híbrido

2.2. Visión artificial

2.2.1. Introducción a la visión por computador

Uno de los sentidos más importantes de los seres humanos es la visión. Ésta es empleada para obtener la información visual del entorno físico. Se calcula que más de 70 % de las tareas del cerebro son empleadas en el análisis de la información visual. Por ejemplo, en Ingeniería Electrónica se emplean esquemas de circuitos, a modo gráfico, para describirlos. Se podría hacerlo mediante texto, pero para la especie humana resulta mucho más eficiente procesar imágenes que procesar texto. La visión humana es el sentido más desarrollado y el que menos se conoce debido a su gran complejidad. Es una actividad inconsciente y difícil de saber cómo se produce. De hecho, hoy en día, se carece de una teoría que explique cómo los humanos perciben el exterior a través de la vista.

La revolución de la Electrónica, con las cámaras de vídeo CCD y los microprocesadores, junto con la evolución de las ciencias de la Computación hace que sea factible la Visión Artificial.

La visión artificial o visión por computador es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un computador. Tal y como los humanos usamos nuestros ojos y cerebros para comprender el mundo que nos rodea, la visión por computador trata de producir el mismo efecto para que las computadoras puedan percibir y comprender una imagen o secuencia de imágenes y actuar según convenga en una determinada situación. Esta comprensión se consigue gracias a distintos campos como la geometría, la estadística, la física y otras disciplinas. La adquisición de los datos se consigue por varios medios como secuencias de imágenes, vistas desde varias cámaras de vídeo o datos multidimensionales desde un escáner médico.

Hay muchas tecnologías que utilizan la visión por computador, entre las cuáles: reconocimiento de objetos, detección de eventos, reconstrucción de una escena (mapping) y restauración de imágenes.

2.2.2. Aprendizaje automático

Las técnicas de aprendizaje automático tienen como objetivo conseguir diferenciar automáticamente patrones usando algoritmos matemáticos. Estas técnicas son comúnmente usadas

para clasificar imágenes, para tomar decisiones dentro del mundo empresarial (por ejemplo, para decidir qué clientes de un banco pueden recibir un préstamo o cuánto ha de pagar cada cliente por un seguro dependiendo de sus antecedentes), así como dentro de muchos otros ámbitos de la ciencia y la tecnología. Principalmente se pueden distinguir dos tipos de técnicas: supervisadas y no supervisadas.

En el aprendizaje supervisado se entrena al computadora proporcionando patrones previamente etiquetados, de forma que algoritmo usado debe encontrar las fronteras que separan los posibles diferentes tipos de patrones. Adaboost y algunas redes neuronales forman parte de este grupo.

En el aprendizaje no supervisado se entrena al computadora con patrones que no han sido previamente clasificados y la misma la que debe agrupar los distintos patrones en diferentes clases. K-means y algunas redes neuronales forman parte de este grupo.

2.2.3. Detección de objetos

La detección de objetos es la parte de la visión artificial que estudia cómo detectar la presencia de objetos en una imagen sobre la base de su apariencia visual, bien sea atendiendo al tipo de objeto (una persona, un coche) o a la instancia del objeto (mi coche, el coche del vecino). Generalmente se pueden distinguir dos partes en el proceso de detección: la extracción de características del contenido de una imagen y la búsqueda de objetos basada en dichas características.

La extracción de características consiste en la obtención de modelos matemáticos compactos que resuman el contenido de la imagen con el fin de simplificar el proceso de aprendizaje de los objetos a reconocer. Dichas características son comúnmente llamadas descriptores.

Los mayores retos tanto de la extracción de características como la clasificación es encontrar descriptores y clasificadores que sean invariantes a los cambios que pueda tener un objeto, como su posición o iluminación.

2.3. Sistemas Operativos en dispositivos móviles

A continuación realizaremos un breve estudio de los principales sistemas operativos que existen en el mercado, analizándolos tanto desde el punto de vista comercial como de desarrollo de aplicaciones. Esto nos servirá para justificar la elección tomada para la implementación de nuestra aplicación del proyecto.

2.3.1. iOS

Es el sistema operativo de Apple Inc. y que fue introducido en su primer iPhone en enero de 2007. Este sistema no es más que una versión reducida del sistema Mac OSX para PC pero aplicada a su Smartphone y que también puede funcionar en otros dispositivos de Apple como son iPod Touch, iPad o Apple Tv.

Figura 2.10: Logo de iOS - Apple

Destaca por ser un sistema muy estable, intuitivo y fácil de usar.
El gran inconveniente del iPhone es su elevado coste y la exclusividad de sus productos. No podemos instalar iOS en ningún terminal que no sea Apple (ni Samsung, ni Nokia). Depende de una computadora con iTunes instalado para poder realizar acciones como la configuración inicial, pasar contenido multimedia al móvil o actualizaciones.
Las aplicaciones para iPhone se programan en xCode a través de un SDK (Software Developement Kit) fácil de utilizar que dispone de un emulador de teléfono integrado. Para ello se necesita suscripción en *iPhone Developement Program*, licencia de desarrollador con un coste de U$S 99 anuales. Además sólo podremos diseñarla a través de un Mac. Tampoco podemos instalar ninguna aplicación que no sea compilada en código nativo.

2.3.2. Android

Android es el sistema operativo de Google, lo cual permite la gestión en los smartphones que utilizan este sistema de productos muy útiles de la misma firma como Gmail, calendarios, gestión de contactos o Google Maps Navigator de forma sencilla.

Figura 2.11: Logo de Android

Principalmente destaca por tratarse de un sistema abierto lo cual permite que cualquier fabricante pueda amoldarlo a sus terminales y desarrollar en él sus productos, de ahí que las actualizaciones del software dependan de los mismos (Samsung, Lg, HTC, etc) y no del propio Google.

Visualmente es un sistema que guarda un cierto parecido de estilo con iOS pero mucho más personalizable.
Android, es líder del mercado en cuanto a descarga de aplicaciones.
La facilidad con la que se pueden transferir archivos entre la computadora y el terminal móvil es otra de sus ventajas. A diferencia de otros sistemas en los que se requiere de un software de sincronización específico, con la simple conexión del terminal al puerto USB del pc es suficiente para poder acceder a todos los recursos y contenidos del dispositivo.
Desde el punto de vista de desarrollo, programar para Android resulta mucho más sencillo. Dispone de una máquina virtual, *Dalvik* (SDK) , fácil de instalar y utilizar en cualquier plataforma : Windows, Linux o Mac. Esta máquina está basada en la actual *JavaRunTimeMachine*, que es la encargada de ejecutar aplicaciones Java, por lo que el lenguaje de programación para las aplicaciones se asemeja bastante a Java. De hecho se dispone nativamente de todas las librerías de Java más las librerías que Google incorpora de su parte.
También se dispone de una web oficial para Android (*developer.android.com*) donde los desarrolladores pueden encontrar abundante documentación, software, tutoriales y ejemplos de diferentes tipos de funcionalidades que permiten la rápida familiarización con este lenguaje, además de infinidad de ejemplos en la red, foros de discusión y páginas web especializadas en desarrollo que sitúan a este sistema operativo como el más utilizado y preferido por los desarrolladores de aplicaciones móviles. Tampoco es necesario invertir en licencias para programar y cargar aplicaciones en el Google Play.

2.3.3. Windows Phone

Es el sistema operativo para móviles de Microsoft que antiguamente se llamaba Windows Mobile. Dentro de los frabricantes que utilizan este sistema operativo el principal es Nokia. Esta alianza entre Windows y Nokia busca penetrar en el mercado para lograr competir con Android e iOS.
Destaca por haber apostado en versión, Windows Phone Mango 7.5, por la sincronización de todos sus productos en el llamada Nube o Cloud-Computing como ha hecho también Apple. Esto va a permitir compartir recursos con los computadoras personales de forma fácil a los usuarios de Windows. Su tienda de aplicaciones llamada Market Place es una de las nuevas pero más logradas, por la facilidad de navegación y detalles de las aplicaciones, pero se queda corta en cuanto a número de programas.

Uno de los principales inconvenientes de esta aplicación es la incompatibilidad con Outlook y la necesidad de instalar un software específico en el computadora (Zune) para poder transferir archivos en el dispositivo móvil.

2.4. Sistemas de comunicaciones de corta distancia

En este apartado se van a describir y analizar algunas de las principales tecnologías existentes para realizar comunicaciones inalámbricas entre dispositivos en corta distancia. La mayoría de estas están implementadas en la banda ISM (Industrial, Scientific and Medical) del espectro radioeléctrico y facilitan las comunicaciones, entre dispositivos móviles y fijos eliminando los cables y conectores. Se destaca la coexistencia en la actualidad de todas estas tecnologías ya que cada una tiene su implementación en un campo determinado dependiendo de las necesidades tecnológicas.

2.4.1. Bluetooth

Viene a ser el nombre común por el que se conoce a la especificación IEEE 802.15.1 que define un estándar global de comunicaciones inalámbricas que permite la transmisión de voz y datos entre diferentes dispositivos mediante un radioenlace a 2,4 GHz.

Figura 2.13: Logo de Bluetooth

Los dispositivos que lo implementan pueden comunicarse entre ellos y transmitir voz y datos siempre que se encuentren dentro de un rango de alcance. Una de las ventajas de este sistema es que lo dispositivos no requieren estar alineados, incluso pueden estar en habitaciones diferentes. Los dispositivos pueden clasificarse en clases en referencia a su potencia de transmisión y a su rango de cobertura, siendo todos compatibles entre si.

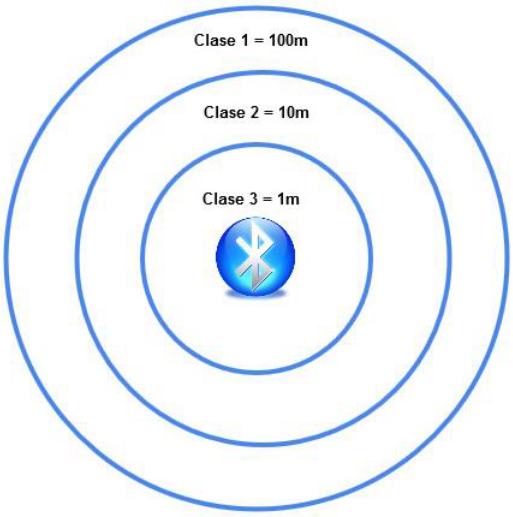

Figura 2.14: Clases de bluetooth según el alcance

Además, los dispositivos Bluetooth también pueden clasificarse según su ancho de banda definiéndose así las distintas versiones existentes:

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0	+ EDR 3 Mbit/s
Versión 3.0	+ HS 24 Mbit/s
Versión 4.0	N/D

Como se indican en las especificaciones de Bluetooth, se define un canal un canal de comunicaciones de máximo 720 Kb/s para un rango óptimo de 10 m. La frecuencia de trabajo se encuentra en el rango de 2.4 GHz a 2.48 GHz con amplio espectro que permite una trasmisión Full-Duplex, síncrona o asíncrona dependiendo de los dispositivos que estén interconectados. La potencia de salida que se requiere para transmitir a una distancia máxima de 10 m es de 0dBm (1 mW) mientras que la versión de largo alcance transmite entre los 20 y 30 dBm (entre 100 mW y 1 W).

Con los sistemas Bluetooth se puede crear una red de área personal (PAN), o también llamada Piconet, dentro de su rango de alcance de hasta 8 dispositivos donde uno de ellos es designado como máster y los otros 7 esclavos. Existe la posibilidad de aumentar el alcance de la red si se asigna a un dispositivo esclavo otra Piconet ya que un dispositivo Bluetooth puede servir a más de un maestro. La técnica que utiliza Bluetooth para evitar interferencias entre dispositivos es el Spread-Spectrum frecuency hopping.

El objetivo de bajo consumo y bajo coste se ha conseguido integrando toda la estructura del sistema en un solo chip elaborado con tecnología CMOS. De esta manera se consiguió elaborar una solución de tamaño 9 x 9 mm y que consume aproximadamente un 97 % menos de energía que un teléfono celular común. Todo ello a permitido que todos los dispositivos móviles actuales del mercado incorporen esta tecno—logía.

2.4.2. NFC

Son las siglas de Near Field Communication y es una de las tecnologías que en la actualidad está teniendo más popularidad. Sus características técnicas basadas en la norma ISO 14443 y el estándar NFCIP-1 lo hacen especialmente apropiado para aplicaciones de seguridad e implementación en el campo del comercio electrónico. Es un sistema de muy corto alcance, menos de 10 cm, que necesita por lo tanto que los terminales estén prácticamente en contacto.

A diferencia de las otras tecnologías, esta trabaja en la banda de los 13,56 MHz pudiendo alcanzar velocidades de 106, 212, 424 0 848 Kbit/s según del entorno en el que se trabaje.

NFC no está pensada para utilizarse en una transmisión masiva de datos, como puede darse con las tecnologías wifi o Bluetooth. Solo sirve para el intercambio rápido de unos pocos bits de información, lo justo para que identifique y valide al usuario.

La implementación de NFC en los dispositivos móviles requiere, como en todas, de la integración de un chip que soporte esta tecnología.

Figura 2.15: Logo de NFC

2.4.3. ZigBee

En una tecnología diseñada por ZigBee Alliance basada en el estándar IEEE 802.15.4 (de redes inalámbricas de área personal, WPAN) y que se desarrolló en 1998 a la par de las tecnologías Bluetooth y Wifi para cubrir el hueco de necesidades que dejan estas tecnologías. La implementación de este protocolo de red, lo hace idóneo para conexiones punto a punto y punto a multipunto con una baja transferencia de datos y que requieran de un bajo consumo energético.

Figura 2.16: Logo de ZigBee

En concreto tiene un consumo de 30 mA transmitiendo y de 3 uA en reposo frente a los 40 mA transmitiendo y 0,2 mA en reposo que tiene el Bluetooth. Además destaca por la sencillez de diseño e implementación. También transmite en la banda de los 2.4 GHz y su rango de acción oscila entre los 10 y 75 metros dependiendo del entorno donde se utilicen pudiendo transmitir datos a velocidades de 250 Kbps.

Como ya se ha comentado, los protocolos ZigBee están definidos para su uso en aplicaciones con requerimientos muy bajos de transmisión de datos y consumo energético. Su ámbito de uso es variado como en sistemas domóticos, redes de seguridad, control industrial, control de sensores empotrados, etc. Pero este no se encuentra disponible en dispositivos móviles.

2.4.4. Infrarrojos

La tecnología infrarroja (IR) fue de las pioneras en la comunicación inalámbrica de dispositivos a corta distancia. Basada en radiocomunicaciones en la zona del infrarrojo, por encima del espectro visible, permite una comunicación bidireccional a velocidades que oscilan entre 9600 bps y 4 Mbps Esta tecnología fue muy extendida a finales de los años 90, principios del 2000, en distintos dispositivos como teléfono móviles, controladores remotos de televisores,

DVD, teclados de computadora, etc..

A diferencia de los anteriores sistemas, se requiere una visión directa entre el dispositivo emisor y receptor para poder realizar la comunicación y que esta no supere el metro de distancia.

La utilización de esta tecnología ha caído en desuso a favor también de Bluetooth.

Capítulo 3

Análisis del problema

3.1. Planteo del problema

El proyecto a desarrollar está basado en un robot móvil programable, que permita cumplir con los objetivos planteados inicialmente, para ello, se necesita que tenga acceso a sensores (en nuestro caso, se utilizará un sensor de ultrasonido para la detección de objetos). Dado que el robot debe tener visión, la captura y procesamiento de la imagen se desarrollará mediante la ejecución de una aplicación en el dispositivo móvil.
El procesamiento debe ser el más rápido posible, dado que si tarda mucho tiempo, la reacción del robot en la situación de un cambio de dirección no será la deseada. En este caso, no solo podría no seguir correctamente a su objetivo, sino inclusive perderlo. Por este motivo, el tiempo de procesamiento debe ser el menor posible, para ello, se necesita un código lo mas simple, rápido y eficiente posible.
Esta problemática y tipo de solución, se encuentra realizada en otros ámbitos. Lo que se buscó en este trabajo, fue simplificar las soluciones existentes, y realizarlo sin necesidad de recursos de un costo tan alto, dado que no se utilizan ni equipos o materiales costosos, y la aplicación puede ser instalada en cualquier dispositivo con Android. Los robots empleados en este tipo de tareas suelen ser muy complejos, y de grandes requerimientos.

3.2. Descripción del sistema desarrollado

Se propone de un conjunto compuesto de un robot, que recibirá comandos de un dispositivo móvil, el cual hará el procesamiento necesario con las capturas obtenidas. Para su sincroni-

zación e intercambio de datos, se utilizará un sistema de comunicación basado en Bluetooth (Véase A.1). Este será el encargado de transmitir las ordenes del dispositivo móvil al robot, para realizar la búsqueda. Se puede observar un diagrama en bloques del sistema propuesto en la Figura 3.1.

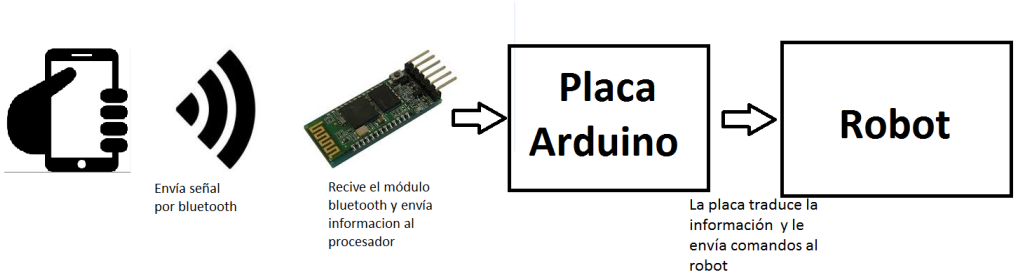

Figura 3.1: Diagrama en bloques del sistema propuesto

Del sistema planteado, se seleccionó la plataforma Arduino (Véase B.1) para el robot, dada su amplia variedad de librerías para el control de motores, dispositivos de comunicación, y en aplicaciones de robótica. En lo que hace referencia al dispositivo móvil, se encuentra implementado en Android (Véase C.1), dado que es un lenguaje de programación orientado a objetos y con una extensa cantidad de librerías que serán de gran ayuda para la solución de la problemática. Por último para la comunicación, se utilizó un módulo de comunicación bluetooth HC-06, para el Arduino, dado que permite la comunicación con el sistema Bluetooth que los dispositivos móviles poseen integrados.

Se utilizó una librería especializada en procesamiento de imágenes, *OpenCv4Android* (Véase C.5). En esta librería no solamente permite realizar la captura de frames con una velocidad adecuada para el procesamiento, sino que también permite realizar el procesamiento de imágenes de un modo simple y eficiente.

Por último, se dotó al robot de sensores de proximidad por ultrasonido, estos fueron utilizados para prevenir situaciones en las cuales el robot pudiera colisionar. Es necesario aclarar, que dado que se utiliza una sola cámara, no es posible calcular la distancia del objetivo ya que se tiene una visión en 2D solamente.

Como agregado al sistema, para permitirle al usuario tener un mayor control sobre el dispositivo en movimiento, se diseñó una página web. En dicha página, se reciben los frames que observa el robot y los muestra en pantalla. Además, cuenta con las tres funciones principales de la aplicación (iniciar y detener el movimiento, y iniciar la comunicación por Bluetooth).

Para realizar dicha aplicación web serán necesario utilizar los lenguajes *PHP* (Véase D.1), *HTML* (Véase D.2), *JavaScript* (Véase D.3) y *CSS* (Véase D.4) que se explicarán luego.

Capítulo 4

Diseño e implementación

4.1. Diseño del robot

En este proyecto se utilizó uno de los robots móviles que posee el laboratorio, el cual se puede observar en la Figura 4.1.

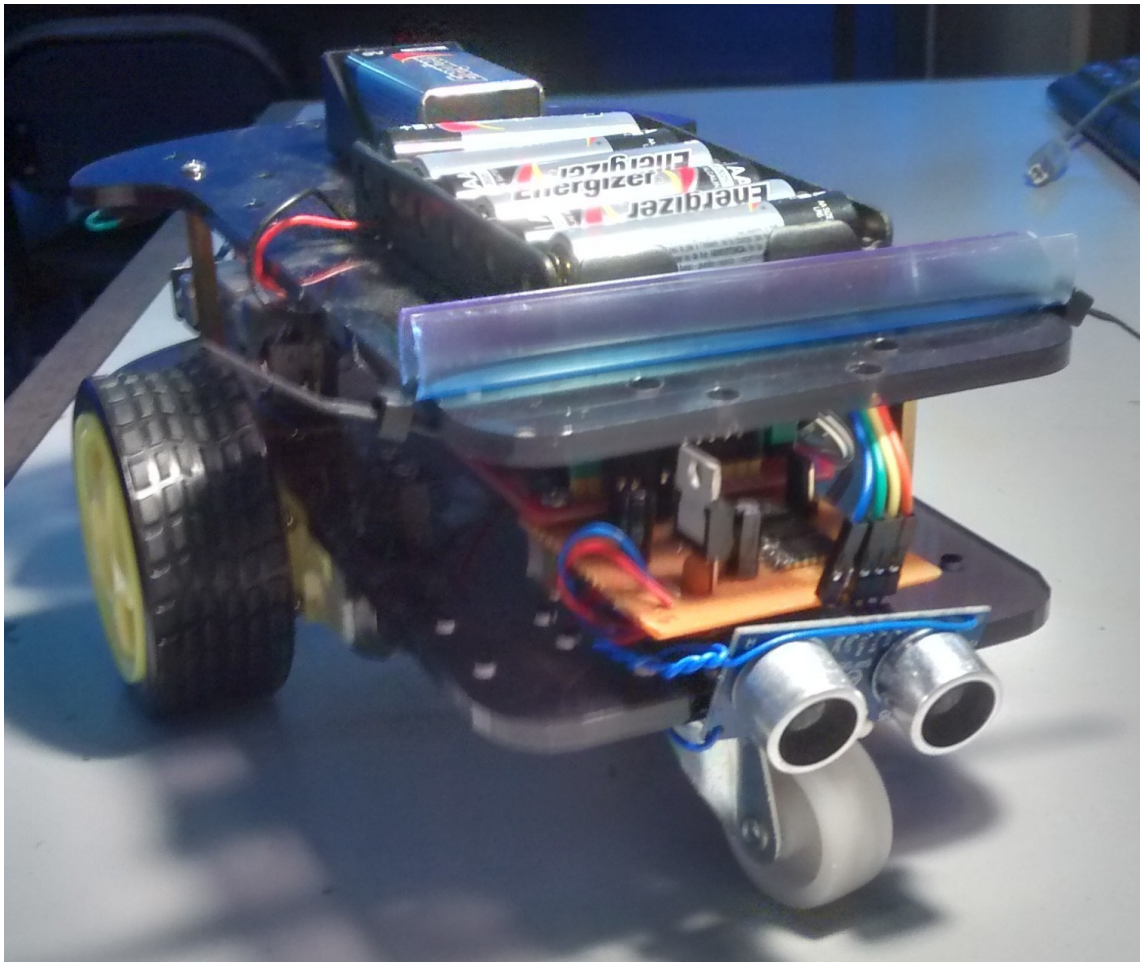

Figura 4.1: Robot utilizado

Se le agregó un soporte para que pueda incorporar el dispositivo móvil que lo dotará de adquisición y procesamiento de la imagen de vídeo.

Dicho robot está basado en un módulo Arduino y una placa de control de los motores de corriente continua. El Arduino entrega a la placa de control una señal en formato PWM para controlar la velocidad de los motores de continua (Véase E.2) y esta convierte dicha información para los motores.

El robot además cuenta con un sensor de proximidad (Véase E.1) mediante el cual sensa y controla para evitar colisiones. Dicho sensor tiene prioridad dentro del código, por lo tanto en caso de aproximarse al objeto, el robot tiene programada una rutina para evadirlo.

Además posee tracción independiente en dos ruedas únicamente como se muestra en la Figura 4.2, y una tercer rueda de apoyo. Entonces, para realizar movimientos hacia adelante, se debe configurar el PWM para la misma velocidad en ambas ruedas. Luego, para movimientos a la izquierda o derecha, deberá tener un PWM mayor en las ruedas externas al giro, como se puede observar en la Figura 4.2 .

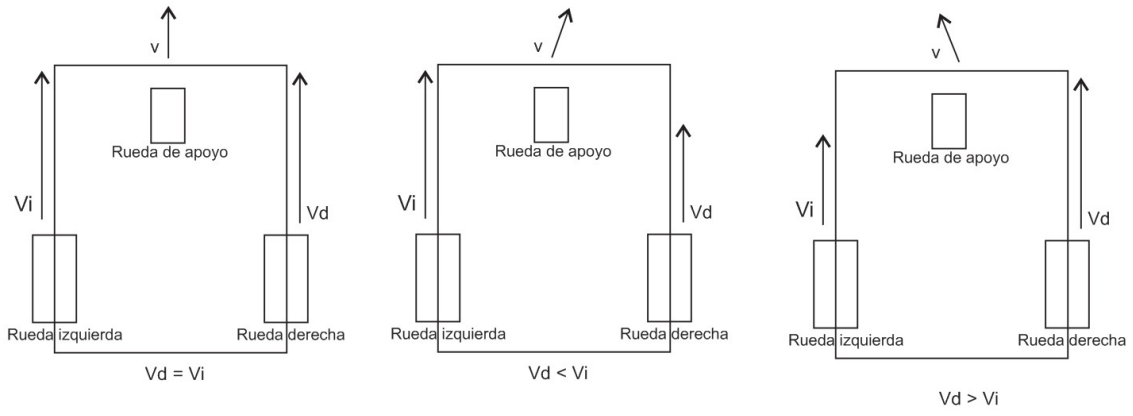

Figura 4.2: Posibles movimientos. De izquierda a derecha, hacia adelante, hacia la derecha, hacia la izquierda

4.2. Diseño de la aplicación Android

4.2.1. Organización del código

El código propuesto se encuentra dividido en cuatro principales grupos. Entre los cuales se encuentra, *inicialización de la aplicación y librerías*, *comunicación Bluetooth*, *cámara y obtención de frames*, y por último *procesamiento de imágenes*. A continuación se realizará una breve explicación del funcionamiento de cada uno de las funciones anteriormente menciona-

das. El funcionamiento se explica con mayor detalle más adelante (incluyendo fragmentos de código y funciones), aquí se explicará cual es el objetivo de cada función. [12] [13] [14]
En el siguiente diagrama se pueden observar las clases que se encuentran dentro de la aplicación y la relación que posee cada una. En el diagrama, no se incluyen las funciones para la visualización web, dado que se generan en un *Thread* (Véase C.7) aparte y serán explicadas más adelante en la sección 4.5.

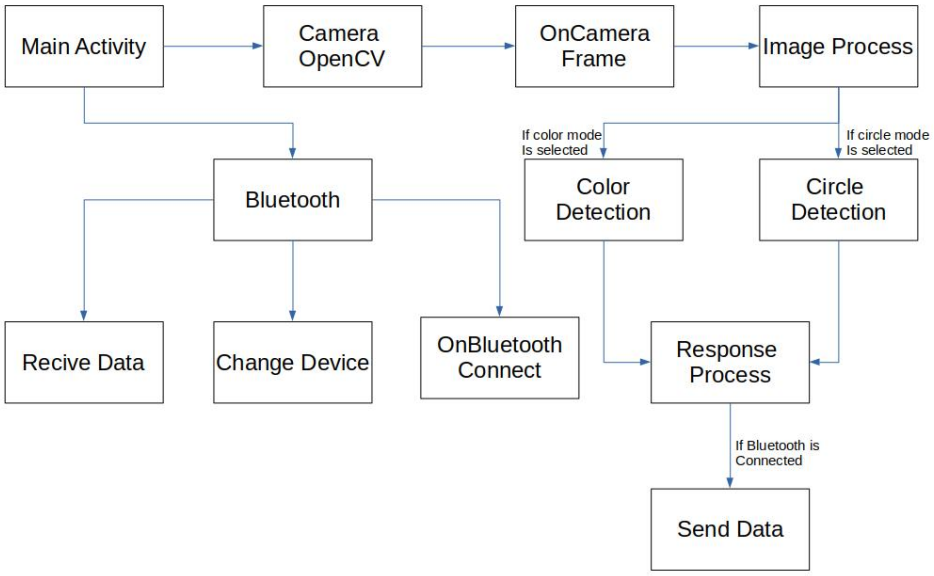

Figura 4.3: Esquema de las clases utilizadas

4.2.2. Inicialización

La actividad principal en el funcionamiento de la aplicación es la denominada *MainActivity* en la Figura 4.3. En esta actividad se encuentra la inicialización de las librerías predeterminadas por el sistema operativo Android para su funcionamiento. Pero también, se puede hallar en el mismo la inicialización para la cámara y la obtención de frames, dado que la librería OpenCv requiere de una inicialización especial para poder ser utilizada. Esta actividad cuenta con las funciones principales que se explican a continuación:

1. **Función onCreate:** Dentro de esta función se realiza la inicialización de variables, se inicia el denominado Layout. El layout es el encargado de mostrar la interfaz gráfica de la aplicación, botones, imágenes, opciones, etc. Dentro de esta misma función se

coloca la inicialización de la cámara por OpenCv, dado que al iniciar la aplicación se debe informar que no se utilizará la librería por defecto de la aplicación.

2. **Función onClick:** Aquí se realiza la definición de todas las utilidades que se realizaran en caso de ser presionado alguno de los botones dentro de la pantalla. En el caso de que sea presionado alguno, se ingresa a este evento y realiza el codigo correspondiente.

Si bién dentro del código se pueden encontrar otras funciones además de las mencionadas anteriormente, dichas funciones no hacen al funcionamiento del programa. Estas son las encargadas de realizar la creación del menu de opciones, agregan procedimientos en caso de que la aplicación se cierre de manera inesperada, se suspenda la aplicación, entre otras. El funcionamiento de estas opciones no es escencial, pero deben ser agregadas para mayor fluidez.

4.2.3. Comunicación

Dentro de la sección de comunicación Bluetooth se encuentran los bloques mostrados en la Figura 4.3 referentes a Bluetooth, Change Device, OnBluetooth Connect/Disconnect, Send data y Recive Data. Dado que la comunicación debe realizar distintas funciones, se listará a continucación el funcionamiento y la utilidad de cada uno de esos bloques. Antes de proceder a la explicación de los bloques mencionados, se debe tener en cuenta, que la transmisión de datos por este medio, se encuentra especialmente orientada a la conexión. Esto quiere decir, que para poder enviar y recibir datos por este medio, primero se debe conectar al dispositivo y ambos deben estar listos para comunicarse entre sí. Por este motivo, en el caso de que se intenten envíar datos sin haberse conectado, se produce un error.

1. **Bluetooth:** Dentro de este grupo de funciones, se encuentran las encargadas de definir los denominados socket de comunicación, en estos socket se hace la definición de donde se conectará, el tipo de información a enviar (dado que los dispositivos móviles pueden enviar datos, audio, etc). Por otro lado, en este grpo se encuentran funciones que preparan todos los aspectos de la comunicación listos para iniciar la conexión. Además se encuentran definidas funciones, para cuando la comunicación es cerrada. En esta, se liberan los recursos para en el caso de desear una nueva comunicación no se produzca error.

2. **Change Device:** En estas funciones, se permite el cambio de dispositivo con el que se realizará la comunicación. Este grupo se encuentra dividido en dos sub-grupos principales, por un lado se encuentra la opción de realizar el cambio de dispositivo con los dispositivos que se encuentran apareados con el equipo. Aquí simplemente se selecciona el deseado y se procede a iniciar la comunicación. El otro grupo posee una complejidad mayor, dado que permite el tipeado del nombre del dispositivo a realizar la conexión y realiza el procedimiento de apareamiento (*paring*). Una vez realizado este procedimiento realiza la conexión en caso de ser posible, o retorna un mensaje de error en caso contrario.

3. **OnBluetooth Connect/Disconnect:** Tal como el nombre del grupo lo denomina, este grupo se encuentra definido en dos funciones principales, realizar la conexión y desconexión de los dispositivos. Si bién solo realizan este proceso, las funciones cuentan con la complejidad de realizar la comunicación de modo correcto para que se encuentren conectados durante todo el proceso. Luego, al realizar la desconexión del dispositivo, se debe preparar las variables para luego se proceda a liberar los recursos y se pueda preparar para una nueva conexión.

4. **Send data:** Luego, en esta función se encarga de realizar el envío de datos a través de la comunicación, tanto de envío como de recepción de información. En esta función se envían los comandos al robot para que realice de modo correcto su búsqueda. En la misma se le realiza el agregado de los caracteres especiales fin de linea y retorno de carro antes de realizar el envío de datos.

5. **Recive data:** La última función de comunicación es la encargada de recibir información del robot y procesarla. Esta información tiene dos utilidades, la principal es conocer la distancia al objetivo. Esta distancia se utiliza para hacer un control más efectivo del robot. Por otro lado se encuentra información básica sobre el estado del robot, principalmente asegurar la correcta comunicación entre dispositivos.

4.2.4. Cámara y obtención de frames

En este grupo se destacan principalmente las funciones de Camera OpenCv y OnCamera Frame. Realmente este grupo debería incluir así también lo referente a los recursos de la

cámara utilizados, y el seteo de la cámara. Se mencionarán y explicarán solo las dos funciones mencionadas, dado que son las que hacen al correcto funcionamiento de la aplicación.
La primer función es la encargada de realizar la inicialización de la cámara. En la cual, se abre la cámara y comienza a mostrar la imagen en la pantalla, mediante la cual se podrá ver lo que se esta buscando (una vez inicializada la búsqueda). También inicia las funciones de la librerías de OpenCv.
La segunda función es por la cual se obtienen los frames que serán procesados, dichos frames se obtienen directamente con la función proveniente de la librería de OpenCv. En dicha librería se puede setear un determinado numero de FPS para que se obtengan, en este trabajo se dejará que se utilicen la cantidad de frames que permita el dispositivo según la utilización del mismo. De este modo, aseguramos que se trabaje a la mayor velocidad posible y el seguimiento sea el mejor posible.

4.2.5. Procesamiento

Los bloques que se incluyen en este grupo son: Image Process, Color Detection, Circle Detection, Response Process. Además de los bloques que se encuentran actualmente funcionando, el codigo se encuentra preparado para agregarle distintas funcionalidades. Dado que posee una actividad para cambiar el modo en el que se desea trabajar, y posee las funciones separadas del cuerpo principal del programa, de este modo se puede agregar o quitar de una forma muy simple.
La funciones principales de este grupo son las de Image Process y Respose Process. La primera es la encargada de tomar los frames que se obtienen de OnCamera Frame, y prepararlas para su procesamiento, es decir, deja todo preparado para que luego se realice la búsqueda de patrones, ya sea por color o por formas. En esta misma función se colocan todas las conversiones de formatos necesarios para ingresarlos luego a las funciones de procesamiento.
Por otro lado, la segunda función es la que toma los parámetros de ubicación del objeto, es decir, las posiciones X Y, y con las mismas ubica donde se encuentra ubicado en la pantalla. Con dicha ubicación realiza el procesamiento necesario para ubicar en que sentido se debe mover el robot. En la sección 4.3 se detallará el modo en el que se analiza el movimiento.
Por otro lado, se encuentran las funciones implementadas de detección de patrones. En las mismas, mediante el uso de OpenCv se realiza una detección de bordes, hasta conseguir el conjunto de propiedades que describen al objeto , como por ejemplo su color y/o forma.

Dichas librerías se encuentran optimizadas, dado que llevan años siendo mejoradas, por este motivo se decidió utilizar librerías ya establecidas.

4.3. Procesamiento de Imágenes

La librería *OpenCV4Android* se utiliza para la captura y procesamiento de las imagenes desde la camara del dispositivo portatil mediante codigo en Android, lo que permite trabajar desde el codigo Java.

4.3.1. Cargar librería de OpenCv en Android

Antes de comenzar con el método de procesamiento, se debe incluir la librería en el proyecto, y realizar la inicialización de la misma. Para hacerlo se utilizará la activity principal.
El proceso de carga de librería es asíncrono, por lo tanto, es de utilidad tener una función callback que se ejecute cuando el proceso de carga de la librería se haya completado.
Por lo tanto, lo primero que hacemos es definir la función:

```
1   private BaseLoaderCallback  mLoaderCallback = new BaseLoaderCallback(this)
    {
2       @Override
3       public void onManagerConnected(int status) {
4           switch (status) {
5                 //Si se pudo conectar correctamente inicia la cámara.
6               case LoaderCallbackInterface.SUCCESS: {
7                   Log.i(TAG, "OpenCv loaded successfully");
8                   //Inicia la visualización de la cámara
9                   mOpenCvCameraView.enableView();
10                  //Setea el listener para cuando se presiona la imagen
11                  mOpenCvCameraView.setOnTouchListener(MainActivity.this);
12                  break;
13              }
14              default: {
15              //Si no logra conectarse, vuelve a intentarlo.
16                  super.onManagerConnected(status);
17              }
18          }
19      }
20  };
```

Luego, se debe colocar dentro del método onResume() la llamada a la inicialización de la librería mediante la función:

```
1  public void onResume(){
2    super.onResume();
3    //Inicialización de librería de OpenCv
4      OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_0_0, this,
     mLoaderCallback);
5
6    }
```

En este caso se utilizará la librería 3.0.0 de OpenCv.

4.3.2. Obtener frames

Para lograr obtener el flujo de imágenes de la cámara, la Activity que se encargará de esta tarea debe implementar el método *CvCameraViewListener2*. La definición de esta clase quedará de la siguiente forma:

```
1  public class MainActivity extends AppCompatActivity implements
     CvCameraViewListener2, View.OnClickListener,View.OnTouchListener {
2
3  }
```

La definición significa, que la clase *MainActivity* que extiende a *AppCompatActivity*, es decir, será una actividad con una pantalla de aplicación y funciones. La misma implementa a los tres métodos *CvCameraViewListener2, View.OnClickListener, View.OnTouchListener*
Las dos implementaciones restantes son necesarias para utilizar los listeners de eventos de presionar botones y presionar la pantalla, dado que más adelante serán necesarios.
La implementación de *CvCameraViewListener2*, obliga a la implementación de las siguientes tres funciones:

```
1  void onCameraViewStarted(){
2  }
```

```
3
4   void onCameraViewStopped(){
5   }
6
7   Mat onCameraFrame( Mat inputFrame ){
8     return inputFrame;
9   }
```

Las primeras dos funciones, como su nombre lo indica, son utilizadas para las definiciónes y seteos de la camara al iniciar y cerrar la camara.

En la última función *onCameraFrame* es en la cual se recibirá la matriz correspondiente a cada frame de la cámara, y debe retornar la matriz que será visualizada en la pantalla. En este método se realizará el procesamiento del frame y retornar el resultado.

A continuación, se puede observar el modo en el que fue utilizada dentro de este proyecto, en el orden mostrado anteriormente:

```
1     @Override
2     public void onCameraViewStarted(int width, int height) {
3         mRgba = new Mat(height, width, CvType.CV_8UC4); //Matriz con
    información RGB
4         mGray = new Mat(height, width, CvType.CV_8UC4); //Matriz con
    información en escala de grises
5         mDetector = new ColorBlobDetector();  //Información para detección de
    colores
6         mSpectrum = new Mat();     //Matriz con información del espectro
7         mBlobColorRgba = new Scalar(255);
8         mBlobColorHsv = new Scalar(255);
9         SPECTRUM_SIZE = new Size(200, 64);
10        CONTOUR_COLOR = new Scalar(255,0,0,255);
11    }
12
13    @Override
14    public void onCameraViewStopped() {
15        mRgba.release();
16        mGray.release();
17    }
18
```

```
19      @Override
20       public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame
        ) {
21           mRgba = inputFrame.rgba();
22           mGray = inputFrame.gray();
23           if (colorDetect) {
24                procesamientoColor();
25           }else
26           if (faceDetect){
27                procesamientoFace();
28           }else
29           if (circleDetect){
30                procesamientoCircle();
31           }
32           return mRgba;
33       }
```

Puede observarse que al iniciar la cámara se hace el seteo de las variables necesarias. Mientras que al cerrar la cámara es necesario liberar los recursos para que no se produzca error al momento de volver a iniciarla.

Luego, para lograr la visualización en la aplicación web, se agregó el llamado a una función para enviar al servidor las imágenes. En forma completa, la función *onCameraFrame* quedaría como se observa a continuación.

```
1       @Override
2       public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame
        ) {
3           mRgba = inputFrame.rgba();
4           procesamientoColor();
5           //Para enviar la imagen
6           //Creo un mapa de bits a partir de la imágen original
7           bm = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(), Bitmap.Config.
        ARGB_8888);
8           Utils.matToBitmap(mRgba,bm);
9           //Si se encuentra listo para enviar la siguiente imagen, inicia la
        comunicación y envia la imagen
10          if (ready) new ServerUpdate().execute();
```

```
11          return mRgba;
12      }
```

En la sección 4.5 se explicará en detalle como enviar la imagen al servidor.

4.3.3. Algoritmo de detección del objetivo

El algoritmo de detección consiste en la búsqueda de objetos de un color predeterminado para lo cual se utilizo la clase denominada *ColorBlobDetector* que permitió realizar la detección del color deseado dentro de la imagen. Dicho código, a su vez, almacena en memoria los contornos de todas las figuras que cumplan con los requisitos. A continuación se puede observar las funciones que posee código utilizado para lograrlo:

```
1  public class ColorBlobDetector {
2
3       public void setColorRadius(Scalar radius)
4
5       public void setHsvColor(Scalar hsvColor)
6
7       public Mat getSpectrum()
8
9       public void setMinContourArea(double area)
10
11     public void process(Mat rgbaImage)
12
13      public List<MatOfPoint> getContours()
14  }
```

Las seis funciones mostradas en la clase anterior, cumplen las funciones de detectar un color dentro del frame ingresado. La primera, *setColorRadius* simplemente almacena dentro de una variable local de la clase, el radio de color que se desea setear, dado que se trabaja con distintas clases, antes de utilizarla se debe guardar los valores con los que se trabajará.

```
1      public void setColorRadius(Scalar radius) {
2          mColorRadius = radius;
3      }
```

Luego, *setHsvColor* se encarga de delimitar los rangos con los que se trabajará para realizar el procesamiento.

```
1   public void setHsvColor(Scalar hsvColor) {
2       double minH = (hsvColor.val[0] >= mColorRadius.val[0]) ? hsvColor.val
    [0]-mColorRadius.val[0] : 0;
3       double maxH = (hsvColor.val[0]+mColorRadius.val[0] <= 255) ? hsvColor.
    val[0]+mColorRadius.val[0] : 255;
4
5       mLowerBound.val[0] = minH;
6       mUpperBound.val[0] = maxH;
7
8       mLowerBound.val[1] = hsvColor.val[1] - mColorRadius.val[1];
9       mUpperBound.val[1] = hsvColor.val[1] + mColorRadius.val[1];
10
11      mLowerBound.val[2] = hsvColor.val[2] - mColorRadius.val[2];
12      mUpperBound.val[2] = hsvColor.val[2] + mColorRadius.val[2];
13
14      mLowerBound.val[3] = 0;
15      mUpperBound.val[3] = 255;
16
17      Mat spectrumHsv = new Mat(1, (int)(maxH-minH), CvType.CV_8UC3);
18
19      for (int j = 0; j < maxH-minH; j++) {
20          byte[] tmp = {(byte)(minH+j), (byte)255, (byte)255};
21          spectrumHsv.put(0, j, tmp);
22      }
23
24      Imgproc.cvtColor(spectrumHsv, mSpectrum, Imgproc.COLOR_HSV2RGB_FULL,
    4);
25  }
```

Continuando en el orden anterior, se encuentra una función para obtener el espectro de colores con los que se está trabajando, para ello simplemente se debe entregar el espectro que se tiene almacenado dentro de la clase.

```
    public Mat getSpectrum() {
        return mSpectrum;
    }
```

Al igual que las funciones de seteo anteriores, la función *setMinContourArea*, setea el área mínima que tendrá el objeto encontrado.

```
    public void setMinContourArea(double area) {
        mMinContourArea = area;
    }
```

La función *process* es una de las más importantes dentro de este código, dado que es la función que procesa la imagen buscando los objetivos. Para ello, utiliza funciones de OpenCv dentro de las cuales realiza búsqueda mediante filtros y detección de contornos.

```
  public void process(Mat rgbaImage) {

        Imgproc.pyrDown(rgbaImage, mPyrDownMat);
        Imgproc.pyrDown(mPyrDownMat, mPyrDownMat);

        Imgproc.cvtColor(mPyrDownMat, mHsvMat, Imgproc.COLOR_RGB2HSV_FULL);

        Core.inRange(mHsvMat, mLowerBound, mUpperBound, mMask);
        Imgproc.dilate(mMask, mDilatedMask, new Mat());

        List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
        try {
          Imgproc.drawContours(rgbaImage, contours, -1, CONTOUR_COLOR, -1);
      //Warna full merah smw >> Terblock
            Imgproc.findContours(mDilatedMask, contours, mHierarchy, Imgproc.
    RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);
    } catch (Exception e) {
      // TODO: handle exception
    }

        double maxArea = 1000;
        Iterator<MatOfPoint> each = contours.iterator();
```

```
21          while (each.hasNext()) {
22              MatOfPoint wrapper = each.next();
23              double area = Imgproc.contourArea(wrapper);
24              if (area > maxArea){
25                  maxArea = area;
26              }
27          }
28          mContours.clear();
29          each = contours.iterator();
30          while (each.hasNext()) {
31              MatOfPoint contour = each.next();
32              if (Imgproc.contourArea(contour) > mMinContourArea*maxArea) {
33                  Core.multiply(contour, new Scalar(4,4), contour);
34                  mContours.add(contour);
35              }
36          }
37      }
```

Por último, la función *getContours*() es donde se guardan los contornos de todas los objetos encontrados durante el procesamiento. Esta función entrega una lista de contornos, que luego serán dibujados en la pantalla en otras funciones.

```
1       public List<MatOfPoint> getContours() {
2           return mContours;
3       }
```

Dentro de la clase *ColorBlobDetector* se puede observar que posee las funciones necesarias para la realización y detección de un espectro de colores, como así realizar la detección de bordes que permite ubicar las regiones a detectar.

Para determinar el color del objeto a utilizar, se desarrolló una interfaz gráfica, que permite, presionando sobre la pantalla, tomar una muestra del color que se desea seguir.

Para lograrlo se utilizó la función *onTouch* (función que es implementada con *View. OnTouchListener* en la definición de *MainActivity*), la cual se ejecuta en el momento en el que hay un evento de tocar la pantalla. Esta función utiliza las funciones de *ColorBlobDetector* y determina el color seleccionado. Para que se pueda observar el color seleccionado, esta misma

función crea un rectangulo en la parte superor izquierda de la pantalla en la cual indica el color seleccionado y la gama de colores que serán detectados. A continuación se muestra la función mencionada:

```
//Setea el listener para cuando se clickea la pantalla.
    @SuppressLint("ClickableViewAccessibility")
    public boolean onTouch(View v, MotionEvent event) {
      //Realiza el analisis de un nuevo color solo si no hay color seleccionado.
        if (!isColorSelected){
            isColorSelected =true;
            int cols = mRgba.cols();
            int rows = mRgba.rows();

            int xOffset = (mOpenCvCameraView.getWidth() - cols) / 2;
            int yOffset = (mOpenCvCameraView.getHeight() - rows) / 2;

            int x = (int) event.getX() - xOffset;
            int y = (int) event.getY() - yOffset;
            //Punto seleccionado, (x,y)
            Log.i(TAG, "Touch in: " + x + " " + y);
    //Si el valor elegido no se encuentra dentro de la imagen, sale de la función.
            if ((x < 0) || (y < 0) || (x > cols) || (y > rows)) returnfalse;
            showMessage("Color is selected");

    //Si es correctamente seleccionado, selecciona un color promedio de la zona en la que se toco.
            Rect touchedRect = new Rect();

            touchedRect.x = (x>4) ? x-4 : 0;
            touchedRect.y = (y>4) ? y-4 : 0;

            touchedRect.width = (x+4 < cols) ? x + 4 - touchedRect.x : cols - touchedRect.x;
            touchedRect.height = (y+4 < rows) ? y + 4 - touchedRect.y : rows - touchedRect.y;
            Mat touchedRegionRgba = mRgba.submat(touchedRect);

```

```
31              Mat touchedRegionHsv = new Mat();
32              Imgproc.cvtColor(touchedRegionRgba, touchedRegionHsv, Imgproc.
    COLOR_RGB2HSV_FULL);
33
34              // Calcula el color promedio de la región que se tocó
35              mBlobColorHsv = Core.sumElems(touchedRegionHsv);
36              int pointCount = touchedRect.width*touchedRect.height;
37              for (int i = 0; i < mBlobColorHsv.val.length; i++)
38                  mBlobColorHsv.val[i] /= pointCount;
39
40              mDetector.setHsvColor(mBlobColorHsv);
41
42              Imgproc.resize(mDetector.getSpectrum(), mSpectrum, SPECTRUM_SIZE);
43
44              touchedRegionRgba.release();
45              touchedRegionHsv.release();
46          } else {
47              showMessage("Color is already selected");
48          }
49          returntrue;
50      }
```

Por último, se encuentra la función que se ejecuta en el momento que se recibe un frame. Esta función es la que le da la funcionalidad de busqueda a la aplicación, dado que se trata de la función principal, que se encuentra dentro de *onCameraFrame*. Una vez que ingresa a dicha función, hace utilización de las dos funciones mencionadas anteriormente y es la que le entrega a la función de envío de mensaje para controlar al robot. También es la que determina en que posición se encuentra y cual es el siguiente paso a seguir. La funcion se muestra a continuación:

```
1       private void procesamientoColor() {
2          // Llama a la función process para encontrar los contornos
    correspondientes.
3          mDetector.process(mRgba);
4          //Almaceno los contornos dentro de una variable para visualizarlos
    luego.
5          List<MatOfPoint> contours = mDetector.getContours();
```

```
6         if (isColorSelected) {
7           //Dibujo los contornos hallados
8             Imgproc.drawContours(mRgba, contours, -1, CONTOUR_COLOR);
9             for (int i = 0; i < contours.size(); i++) {
10                if (Imgproc.contourArea(contours.get(i)) > 20) {
11                    Rect rect = Imgproc.boundingRect(contours.get(i));
12                    if (contours.size() == 1) {
13                        moveTo(rect.x);
14                        lastValue = rect.x;
15                    } else {
16                        moveTo(lastValue);
17                    }
18                }
19            }
20            if (contours.size() == 0 && startSearch) {
21                moveTo(-100);
22            }
23        }
24        Mat colorLabel = mRgba.submat(4, 68, 4, 68);
25        colorLabel.setTo(mBlobColorRgba);
26        Mat spectrumLabel = mRgba.submat(4, 4 + mSpectrum.rows(), 70, 70 +
    mSpectrum.cols());
27        mSpectrum.copyTo(spectrumLabel);
28    }
```

Dentro de este código se encuentra utlizada la función *moveTo()*, esta función se explicará luego. Dentro de la misma, se toma el punto donde se encuentra el objeto y se decide hacia que lugar debe moverse el robot. Por último, hace el llamado a la función que se encarga de enviar los mensajes por bluetooth al robot.

4.3.4. Función de cálculo y accionamiento de los motores

Como se mencionó anteriormente, dentro de la función *moveTo()* se toma la decisión final y se determina hacia donde se debe mover el robot para luego envíar el mensaje al mismo para que ejecute su orden. La función puede observarse a continuación:

```
1  public void moveTo(int place){
```

```
        // La pantalla va de 0 a 800 (imagen completa sin bordes)
        //
        // |-----------------|-------|----------------|---> x
        // 0                 300     500              800
        //
        if (startSearch && distancia>15) {
            if (place > 0 && place < 300) {
                setPot(mDerechaF, mIzquierdaR);
                last = 1;
            }else if (place < 500 && place > 300) {
                setPot(mDerechaF, mIzquierdaF);;
                last = 2;
            }else if (place > 500 && place < 800) {
                setPot(mDerechaR,mIzquierdaF);
                last = 3;
            }else if (place == -1){
                sendData("BACK \n");
            }else if (place == -100){
                lookEverywhere();
            }
        }else if(distancia<15 && place < 500 && place > 300){
            this.runOnUiThread(new Runnable() {
                public void run() {
                    sendData("WAIT \n");
                    Toast.makeText(getApplication().getApplicationContext(), "
Se encontró el objetivo! en: " +
                            Double.toString((System.currentTimeMillis() -
timeZ)/1000) + "segundos", Toast.LENGTH_SHORT).show();
                }
            });
        } else if (distancia<15 && startSearch){
            sendData("BACK \n");
        }
    }
```

Se puede observar que la función es simple, dado que se tienen la cantidad de pixeles totales y el punto en el cual se encuentra el centro del objeto. Como se trabaja con una gran cantidad

de FPS, se decidió realizar movimientos pequeños, en ángulos de giro pequeño.
La función utilizada *setPot()* se encarga de enviar la potencia de los motores en un control proporcional a la distancia del objetivo. Para lograr el funcionamiento se necesita además de una función para setear los valores de *mDerecha* y *mIzquierda* dependiendo de la distancia.

Función procDistancia()

```
1  private void procDistancia() {
2          int factor = 15;
3          if (distancia < 50){
4              mDerechaF = 150;
5              mIzquierdaF = 150+factor;
6              mDerechaR = 130;
7              mIzquierdaR = 130+factor;
8          }else if (distancia < 100) {
9              mDerechaF = 165;
10             mIzquierdaF = 165+factor;
11             mDerechaR = 130;
12             mIzquierdaR = 130+factor;
13         }else{
14             mDerechaF = 180;
15             mIzquierdaF = 180+factor;
16             mDerechaR = 150;
17             mIzquierdaR = 150+factor;
18         }
19     }
```

Puede observarse que se toman solo tres valores posibles. Otra posible forma de hacer el controlador más proporcional, sería hacerlo linealmente dependiente con la distancia. No se realizó de dicho modo, ya que aumenta el tiempo de procesamiento y no se logra observar diferencia entre un PWM de n y uno de $n+1$

Función setPot()

```
1  private void setPot(int der,int iz){
2    //Envio con formato predefinido
3          sendData("MOTOR"+iz+der+"\n");
```

```
4     }
```

Esta función simplemente envía un mensaje por bluetooth hacia el robot con el formato predeterminado que el robot espera recibir.
Este método de procesamiento, basado en la detección por color trabaja a un promedio de $15FPS$, que cumple con los requisitos de tiempo real buscado en este trabajo. Esto determina la velocidad de trabajo, dado que por cada frame que es procesado se envía una orden al robot.
La cantidad de frames por segundo esta determinada principalmente por la velocidad de procesamiento del dispositivo móvil, ya que la aplicación se encuentra preparada para trabajar a la velocidad máxima que se permita. Los valores presentados anteriormente están medidos con el dispositivo de prueba (Véase C.9).

4.4. Comunicación Bluetooth

4.4.1. Proceso de paring

Cada dispositivo Bluetooth se identifica con una dirección única de 12 dígitos hexadecimales, y se acompaña de un "friendly name"para poder referenciar los dispositivos fácilmente.
Para que se pueda realizar el intercambio de mensajes entre los dispositivos Bluetooth, estos deben estar previamente sincronizados. En el proceso de sincronización (llamado paring), ambos dispositivos intercambian mensajes para compartir sus direcciones Bluetooth, sus "friendly name", y los perfiles soportados. Una vez que los dispositivos se han sincronizado, pueden realizar sucesivas conexiones usando sus nombres ya que recuerdan las passwords.
Dado que el robot se encuentra funcionando con un módulo Bluetooth que permite solo trabajar en modo esclavo, el que deberá iniciar la comunicación y transferencia de información inicial será el dispositivo móvil.

4.4.2. Comunicación Bluetooth en Android

La API de comunicación Bluetooth de Android está basada en el protocolo RFCOMM. Mediante dicha API es posible buscar dispositivos y conectarse a ellos siempre que se encuentren

dentro de rango. Mediante el Bluetooth Socket se obtiene un enlace de comunicación que permite a las aplicaciones recibir y transmitir flujos de datos.
Nótese que solo se admite comunicación cifrada entre los dispositivos, y por lo tanto deben estar previamente sincronizados tal como se ha expuesto en el apartado anterior.

1 - Local Bluetooth Device Adapter

La clase BluetoothAdapter nos permite controlar el dispositivo bluetooth local del teléfono. Para tener acceso al Bluetooth Adapter hemos de llamar al método getDefaultAdapter como se muestra a continuación.

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

Mediante este controlador podemos activar o desactivar el bluetooth, buscar dispositivos, o leer y modificar propiedades del dispositivo local. Para tener acceso a los privilegios de lectura, tenemos que incluir el permiso BLUETOOTH en el archivo *Manifest* de nuestra aplicación. Si además queremos modificar propiedades del dispositivo, necesitamos incluir también el permiso $BLUETOOTH\_ADMIN$.

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

Para comprobar si el dispositivo está habilitado podemos usar el método isEnabled. Una vez activado podemos acceder a propiedades de nuestro dispositivo, como el nombre o la dirección.

```
if (bluetooth.isEnabled()) {
  String address = bluetoothAdapter.getAddress();
  String name = bluetoothAdapter.getName();
}
```

Para activar el Bluetooth Adapter se puede iniciar una Preference Activity de sistema de la siguiente manera:

```
if (!bluetooth.isEnabled()) {
  //Controla que el bluetooth no este ya habilitado
```

```
    blueTooth.enable();
}
```

Para obtener información mas detallada del estado actual del adaptador, se puede usar el método getState, el cual retorna una de las siguientes constantes:

```
STATE_TURNING_ON
STATE_ON
STATE_TURNING_OFF
STATE_OFF
```

2 - Obteniendo el dispositivo remoto

Para establecer la conexión con un dispositivo se deben cumplir las siguientes condiciones:

1. El dispositivo debe estar visible.
2. El dispositivo remoto debe aceptar conexiones a través de Bluetooth Server Socket.
3. Los dispositivos local y remoto deben estar sincronizados.

Si se cumplen esta serie de requisitos, se puede empezar a buscar un dispositivo. El objeto Bluetooth Device se utiliza para representar dispositivos remotos, a los cuales se pueden realizar peticiones para obtener sus propiedades e iniciar una conexión.
Antes de realizar la conexión al dispositivo, es recomendable comprobar que éste se encuentra visible y que los dos están sincronizados. Se puede realizar una búsqueda de los dispositivos visibles o, en el caso de que conozcamos la dirección del dispositivo con el que vamos a realizar la conexión, es posible especificarla en el método getRemoteDevice.

```
BluetoothDevice device = bluetooth.getRemoteDevice("01:23:77:35:2F:AA");
```

Si se quiere obtener una lista con los dispositivos que están actualmente sincronizados, se puede obtener a través del BluetoothAdapter.

```
Set<BluetoothDevice> bondedDevices = bluetooth.getBondedDevices();
```

3 - Estableciendo conexión

Para iniciar un canal de comunicación con un dispositivo remoto, hay que crear un objeto Bluetooth Socket mediante el Bluetooth Device que hemos obtenido en el paso anterior. Para crear una nueva conexión, se ha de llamar primero al método createRfcommSocketToServiceRecord a través del Bluetooth Device de la siguiente manera.

```
String uuid = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB") );
BluetoothSocket mmSocket = mmDevice.createRfcommSocketToServiceRecord( uuid );
```

Otra manera de crear la conexión que es utilizada es la siguiente:

```
Method m = mmDevice.getClass().getMethod("createRfcommSocket", new Class[] {
    int.class });
mmSocket = (BluetoothSocket)m.invoke(device, Integer.valueOf(1));
```

Una vez se ha obtenido el socket, ya se puede iniciar la conexión mediante la llamada al método connect.

```
mmSocket.connect()
```

4 - Transmisión de datos

Una vez se ha establecido la conexión, es posible transmitir y recibir datos usando los sockets de los dos dispositivos. La transferencia de datos sobre Bluetooth Sockets se gestiona a través de los objetos de Java InputStream y OutputStream, los cuales se obtienen mediante el socket con los métodos getInputStream y getOutputStream. A continuación se puede observar la función utilizada para el envío de información:

```
    private void sendData(String message) {
        byte[] msgBuffer = message.getBytes();

        Log.d(TAG, "...Send data: " + message + "...");

        try {
            mmSocket.getOutputStream().write(msgBuffer);
```

```
        } catch (Exception e) {
            // TODO Auto-generated catch block
            Log.e(TAG,"Error output func");
            e.printStackTrace();
        }
    }
```

Una vez invocada la función se le debe pasar como argumento un String y esta misma función la transforma a Bytes para realizar el envío de datos. Esto debe hacerse dado que la comunicación se hace mediante la transmisión de Bytes.
Luego, para recibir datos por bluetooth, es necesario primero crear un *Handler* para manipular la información que ingresa. Se puede crear como se muestra a continuación.

```
    Handler                         bluetoothIn;
    bluetoothIn = new Handler() {
        public void handleMessage(android.os.Message msg) {
            if (msg.what == handlerState) {
                //Si el mensaje es lo que queriamos recibir
                String readMessage = (String) msg.obj;
                parcialMsg += readMessage;
                processMsg();
            }
        }
    };
```

Una vez que recibimos la información, se procede a procesarla para obtener la información deseada del mensaje.

```
    private void processMsg() {
        if (parcialMsg.contains("\r")||parcialMsg.contains("\n")){
            try {
                String datoString;
                datoString = (String)          parcialMsg.subSequence(
    parcialMsg.length() - 5, parcialMsg.length() - 2);
                distancia = Integer.parseInt(datoString);
                /* Llamo a la función procDistancia() para poder
                 setear los valores de PWM en el que se moverá
```

```
                el robot*/
                procDistancia();
                Long tWrite;
                if (startSearch) {
                    tWrite = (System.currentTimeMillis() - timeZ) / 1000;
                } else {
                    tWrite = Long.valueOf(0);
                }
                /*Seteo el nombre del dispotivijo,
                junto con el estado de la conexión,
                la distancia con el objetivo,
                y el tiempo que procesa*/
                nombreBluetooth.setText("Bt name = " + name + "(Connect) -
    Tiempo: " + Long.toString(tWrite) + " seg \n" + "State (cm) = " +
    parcialMsg);
            }catch (Exception e){
                e.printStackTrace();
            }
            //Limpio el mensaje parcial para recibir nuevamente otro
            parcialMsg="";
        }
    }
```

5 - Conexión y desconexión del bluetooth

Por último, a continuación se muestran las funciones completas de conexión y desconexión del bluetooth. Se puede observar, que como se mencionó anteriormente, se realizó la ejecución en forma asincrona para que la aplicación funcione con mayor fluidez.

```
@SuppressLint("NewApi")
    private class ConnectBT extends AsyncTask<Void, Void, Void> {
        private boolean mConnectSuccessful =true;

        @Override
        protected void onPreExecute() {
            progressDialog = ProgressDialog.show(MainActivity.this, "Loading
    ....", "Connecting");
        }
```

```
9
10          @Override
11          protected Void doInBackground(Void... devices) {
12              if (!connectBt) {
13                  try {
14                      blueTooth = BluetoothAdapter.getDefaultAdapter();
15                      if (!blueTooth.isEnabled()) {        //Controla que el
    bluetooth no este ya habilitado
16                          blueTooth.enable();
17                      }
18                      Set<BluetoothDevice> pairedDevices = blueTooth.
    getBondedDevices();
19                      if (pairedDevices.size() > 0) {
20                          for (BluetoothDevice device : pairedDevices) {
21                              if (device.getName().equals(name)) {
22                                  mmDevice = device;
23                                  //Log.e(TAG, "Device Found");
24                                  break;
25                              }
26                          }
27                      }
28                      if (mmSocket == null || !connectBt) {
29          //Method m = mmDevice.getClass().getMethod("createRfcommSocket", new
    Class[] { int.class });
30          //mmSocket = (BluetoothSocket)m.invoke(device, Integer.valueOf(1));
31          //probar
32                          mmSocket = mmDevice.
    createInsecureRfcommSocketToServiceRecord(mDeviceUUID);
33                          BluetoothAdapter.getDefaultAdapter().cancelDiscovery()
    ;
34                          mmSocket.connect();
35                      }
36                  } catch (Exception e) {
37                      // Unable to connect to device
38                      e.printStackTrace();
39                      mConnectSuccessful =false;
40                  }
41              }
42              return null;
```

```
43          }
44
45          @Override
46          protected void onPostExecute(Void result) {
47              super.onPostExecute(result);
48
49              if (!mConnectSuccessful) {
50                  //finish();
51                  nombreBluetooth.setText("Bt name = " + name + "(No Connect)");
52                  if (intentos < 2) {
53                      intentos++;
54                      new ConnectBT().execute();
55                  }
56                  else{
57                      Toast.makeText(getApplicationContext(), "Failed, Try Again
    ...", Toast.LENGTH_SHORT).show();
58                  }
59                  connectBt =false;
60              } else {
61                  nombreBluetooth.setText("Bt name = " + name + "(Connect)");
62                  sendData("BT-DONE \n");
63                  showMessage("Connected to device");
64                  connectBt =true;
65
66              }
67
68              progressDialog.dismiss();
69          }
```

```
1 private class DisConnectBT extends AsyncTask<Void, Void, Void> {
2
3           @Override
4           protected void onPreExecute() {
5               progressDialog = ProgressDialog.show(MainActivity.this, "Loading
    ....", "Disconnecting");
6               if (mmSocket != null) {
7                   sendData("BT-DISCONNECT \n");
8               }
```

```
9           }
10
11          @Override
12          protected Void doInBackground(Void... params) {
13
14              try {
15                  mmSocket.close();
16                  blueTooth.disable();
17              } catch (Exception e) {
18                  // TODO Auto-generated catch block
19                  e.printStackTrace();
20              }
21
22              return null;
23          }
24
25          @Override
26          protected void onPostExecute(Void result) {
27              super.onPostExecute(result);
28              connectBt =false;
29              nombreBluetooth.setText("Bt name = " + name + "(No Connect)");
30              progressDialog.dismiss();
31          }
32
33      }
```

En ambos casos, las funciones *onPreExecute() y onPostExecute()* muestran cuadros de aviso para informar que el dispositivo está realizando la conexión.

4.5. Programación WEB

A lo largo de esta sección se explicará el modo en que se realizó la página para monitorear los movimientos del robot. Como el robot debe ser autónomo, implica que el usuario no tiene acceso directo sobre el robot mientras se encuentra en funcionamiento.
Un esquema del funcionamiento se puede observar a continuación en la Figura 4.4.

Figura 4.4: Esquema de funcionamiento de la página web

Esta sección se dividirá en dos grupos, por un lado se explicará la comunicación, envío y recepción entre el servidor y el dispositivo móvil. Luego, se analizará la comunicación entre la página web y el servidor.

4.5.1. Comunicación Servidor-Dispositivo móvil

Aqui, se tienen dos funciones principales, por un lado, el envío de datos al servidor, y por otro, la recepción de datos por parte del servidor. Se explicará por separado dado que tienen dos funcionalidades distintas.
Antes de hacer la explicación, es importante destacar, que para poder utilizar un ciertas funciones en Android, se deberá descargar el paquete *apache-httpcomponents-httpclient-cache*. Una vez agregada la librería, es posible hacer utilización de los servicios de la red.

Enviar datos al servidor desde Android

Por cada frame que captura la cámara, Android evalúa el estado de la variable *ready*. En caso de ser verdadero, ejecuta en segundo plano el envío del frame al servidor, como se observa a continuación.

```
1  if (ready) new ServerUpdate().execute();
```

La variable ready, es seteada en falso al iniciar la tarea en segundo plano de enviar frame, y vuelve a verdadero cuando termina. De este modo, permite enviar imágenes cada vez que se libera el uso de la conexión a Internet. A continuación se puede observar la función y la descripción del funcionamiento.

```
    class ServerUpdate extends AsyncTask<String, String, String> {
        @Override
        protected String doInBackground(String... params) {
      //Controlo si se pudo insertar la imagen en el servidor
      //En caso de no haber podido publico mensaje de error
            if (!onInsert())
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(MainActivity.this, "Fallo", Toast.
    LENGTH_SHORT).show();
                    }
                });
            return null;
        }
}
```

En las funciones *onPreExecute()* y *onPostExecute()* se setean la variable ready en falso y verdadero respectivamente.

El envío propiamente dicho al servidor se encuentra dentro de la función *onInsert()*. En esta función se setean las variables que corresponden al uso de los recursos HTTP.

```
    private boolean onInsert() {
        HttpClient httpClient;
        List<NameValuePair> nameValuePair;
        HttpPost httpPost;
        httpClient = new DefaultHttpClient();
        httpPost = new HttpPost(urlInsert);
        nameValuePair = new ArrayList<NameValuePair>(3);
        ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream();
        if (bm != null) {
          //Comprimo la imagen para enviar mas rapido
            bm.compress(Bitmap.CompressFormat.JPEG, 25,byteArrayOutputStream);
            //Convierto el bitmap en un byte para enviar.
            byte[] byteArray = byteArrayOutputStream.toByteArray();
```

```
14              //Agrego nombre, la imagen en compresión Base64 y el tipo, que
    requiere la base de datos
15              nameValuePair.add(new BasicNameValuePair("name", "NameImagen"));
16              nameValuePair.add(new BasicNameValuePair("imagen", Base64.
    encodeToString(byteArray, Base64.DEFAULT)));
17              nameValuePair.add(new BasicNameValuePair("tipo", "image/jpeg"));
18              try {
19                //Intento ejecutar la consulta y enviar datos al servidor
20                  httpPost.setEntity(new UrlEncodedFormEntity(nameValuePair));
21                  httpClient.execute(httpPost);
22                  //Si pude ejecutar la consulta devuelvotrue
23                  returntrue;
24              } catch (UnsupportedEncodingException e) {
25                  e.printStackTrace();
26              } catch (ClientProtocolException e) {
27                  e.printStackTrace();
28              } catch (IOException e) {
29                  e.printStackTrace();
30              }
31          }
32          //Si se produjo error, devuelvofalse
33          // e imprimo mensaje de error en consola
34          returnfalse;
35      }
```

Recibir datos en el servidor desde Android

La función mostrada anteriormente, hace el llamado a una url llamada *urlInsert* a la cual le entrega la información de la imagen. En dicha dirección *php* (Véase D.1), se decodifica el mensaje y almacena dentro de la base de datos como se puede observar a continuación.

```
1 <?php
2     # Inicio la conexión
3     $link = require 'connect.php';
4
5     # Obtengo la información recibida
6     $name = $_POST['name'];
7     $Base64Img = $_POST['imagen'];
```

```
8     $tipo = $_POST['tipo'];
9
10    # Decodifico la imagen
11    # Decodificamos $Base64Img codificada en base64.
12    $Base64Img = base64_decode($Base64Img);
13    # escribimos la información obtenida en un archivo llamado
14    # trabajofinal.png para que se cree la imagen correctamente
15    file_put_contents('trabajofinal.jpg', $Base64Img);
16
17    # Defino la consulta
18    $sql = "INSERT INTO `android`(`name`, `imagen`, `tipo`) VALUES ('".$name."
      ','".$Base64Img."','".$tipo."')";
19
20    # Ejecuto la consulta
21    mysqli_query($link,$sql);
22
23    # Cierro la conexión
24    mysqli_close($link);
25 ?>
```

La función anterior hace el llamado a otra función *php* denominada *connect.php*, que se mostrará más adelante.

Con dichas funciones cada vez que haya un frame disponible, y el dispositivo se encuentre en condiciones de enviarlo, se enviará al servidor.

Recibir información del servidor al dispositivo móvil

Para controlar si algún comando tiene un comando para ejecutar, periódicamente se ejecuta la función denominada *controlbd()*. Esta función ejecuta una tarea en segundo plano, y luego de hacerlo comprueba las variables que devuelve. Si bién esta función es simple, permite hacer control de más de una variables como se muestra a continuación.

```
1     private void controlbd(){
2         new Mostrar().execute();
3         if (Objects.equals(status.getPlay(), "1") && !startSearch){
4             onPlayPulse();
5         } else if (Objects.equals(status.getStop(), "1") && startSearch){
```

```
            onPausePulse();
        }
        if (Objects.equals(status.getConnect(), "1") && !connectBt){
            new ConnectBT().execute();
        }else if (Objects.equals(status.getConnect(), "0") && connectBt){
            new DisConnectBT().execute();
        }
    }
```

La función asíncrona *Mostrar()* ejecuta una consutla al igual que se mostró anteriormente. Mediante una función que permite separar la información recibida, se entrega los comandos requeridos de la base de datos. Se muestra el código que permite realizar dichas funciones. El mismo se encuentra separado en 3 funciones distintas (una tarea asíncrona, una funcion para filtrar los datos y una función que ejecuta la consulta).

```
    class Mostrar extends AsyncTask<String,String,String>{
        @Override
        protected String doInBackground(String... params) {
          //Si se pudo filtrar los datos, se muestra el estado
            if(filrtarDatos())mostrarEstado();
            return null;
        }
    }
        private boolean filrtarDatos(){
        if (!mostrar().equalsIgnoreCase("")){
            String[] cargarDatos = mostrar().split("/");
            for (int i = 0; i < cargarDatos.length; i++){
                String datos[] = cargarDatos[i].split("<br>");
                status.setPlay(datos[0]);
                status.setStop(datos[1]);
                status.setConnect(datos[2]);
                break;
            }
            returntrue;
        }
        returnfalse;
    }
    private String mostrar()    {
```

```
24          String request = "";
25          HttpClient httpClient;
26          HttpPost httpPost;
27          httpClient = new DefaultHttpClient();
28          httpPost = new HttpPost(urlRead);
29          ResponseHandler<String> responseHandler = new BasicResponseHandler();
30          request = httpClient.execute(httpPost,responseHandler);
31          return request;
32      }
```

Enviar información del servidor al dispositivo móvil

Lo que si es importante destacar, es que como se debe hacer una consulta a una base de datos, es necesario un *php* que permita realizarlo. El php nuevamente ejecuta una consulta y envía los datos como se puede observar a continuación.

```
1  <?php
2      # Inicio la conexión
3      $link = require 'conect.php';
4
5      # Genero la consulta
6      $sql = "SELECT * FROM `estado` order by id";
7
8      # Ejecuto la consulta
9      $quert_exec = mysqli_query($link,$sql) or die(mysqli_error($link));
10
11     # Complilo los datos recibidos
12     while ($row = mysqli_fetch_array($quert_exec)){
13         echo $row[1]."<br>".$row[2]."<br>".$row[3]."/";
14     }
15
16     # Cierro la base de datos
17     mysqli_close($link);
18
19 ?>
```

4.5.2. Comunicación Servidor-Aplicación web

Para realizar la comunicación entre el servidor y nuestra página web, se necesitarán tres funciones. Primero, una pagina html que permita juntar la información de las otras dos funciones, javaScript y php.

La función principal cuenta con una página que posee un frame, el cual ejecutará la página php con las configuraciones del javascript.

```
<html>
    <title>Refresca un div tag sin necesidad de refrescar toda la pagina</
    title>
    <head>
        <script src="ajax.js"></script>
        <meta http-equiv="Cache-control" content="no-cache">
        <meta name="expires" content="0">
        <link href="estilo2.css" rel="stylesheet" type="text/css"/>
    </head>
    <body id="imagenBody">
        <div id="contenido">
            <div name="timediv" id="timediv">
            </div>
        </div>
    </body>
</html>
```

Se puede observar que en esta función simplemente se ejecuta el javascript dentro de la división denominada *contenido*.

El agregar la linea $< scriptsrc = "ajax.js" >< /script >$, hace que al iniciar la página se ejecute la función dentro de *ajax.js*. Dicha función, se encarga de llamar en forma periódica a una función php dentro de la división *contenido* (Las funciónes se pueden observar en E.5.1 y E.5.2).

4.5.3. Función connect.php

Esta función es la encargada de hacer el llamado al servidor y a la base de datos.

```
<?php

```

```
3     # Servidor de base de datos
4     define("DBHOST", "localhost");
5     # nombre de la base de datos
6     define("DBNAME", "androidtest");
7     # Usuario de base de datos
8     define("DBUSER", "root");
9     # Password de base de datos
10    define("DBPASSWORD", "vertrigo");
11
12    # Inicio la conexión o muestro mensaje de error
13    $link = mysqli_connect(DBHOST, DBUSER, DBPASSWORD) or die(mysqli_error(
      $link));;
14
15    # Elijo la base de datos o muestro mensaje de error
16    mysqli_select_db($link, DBNAME) or die(mysqli_error($link));
17
18    # Regreso el link de conexión
19    return $link;
20
21 ?>
```

4.5.4. Página finalizada

Por último, en esta subsección, se explica el funcionamiento del código entero, la unión entre lo mencionado anteriormente y las funciones con los botones.

La página principal, se divide en tres espacios, en los que se unen 3 html distintos. En el primero, se encuentra el titulo con los logos de la Universidad. En el segundo se encuentran los frames obtenidos. Luego, en el tercero, se encuentran las tres funciones que permite realizar para comunicarse.

El código principal es relativamente simple, dado que solamente cuenta con un titulo de ventana y un script que ejecuta en una porción de la página los html mencionados. Los llamados se ejecutan en un script, y no directamente, dado que el script permite detectar el tamaño de la pantalla y utilizar porcentajes de la misma para mostrar los html. A continuación se puede observar el código.

```
<!DOCTYPE HTML>
<html>
<head>
    <link href="principal.css" rel="stylesheet" type="text/css"/>
  <title>Trabajo Final</title>
</head>
<body id="principal">
    <script>
        if (window.innerHeight){
           //navegadores basados en mozilla
           espacio_iframe = window.innerHeight - 10
        }else{
           if (document.body.clientHeight){
                //Navegadores basados en IExplorer, es que no tengo
    innerheight
                espacio_iframe = document.body.clientHeight - 10
           }else{
                //otros navegadores
                espacio_iframe = 478
           }
        }
        document.write ('<iframe src="logos.htm" width="75%" height="' + 0.1 *
    espacio_iframe + '"  frameborder="1px">')
        document.write ('</iframe>')
        document.write ('<iframe src="mostrar.htm" width="75%" height="' + 0.7
     * espacio_iframe + '"  frameborder="1px">')
        document.write ('</iframe>')
        document.write ('<iframe src="botones.htm" width="75%" height="' +
    0.18 * espacio_iframe + '"  frameborder="1px">')
        document.write ('</iframe>')
    </script>
</body>
</html>
```

Todos los códiogs html realizados, cuentan con la instrucción $< linkhref = "principal.css" rel = "stylesheet" type = "text/css" / >$, en este caso, el nombre del archivo css es principal. Los archivos css (Véase D.4) cumplen las funciones de darle formato y colores a la información

que se presenta.

Como se observa en el código mostrado arriba, se hace el llamado a tres html distintos. Luego se explicarán los funcionamientos de dichos códigos en las subsecciones siguientes.

Con este código, y las funcionalidades que se mostrarán a continuación, se puede observar que la interfaz de control resulta como se muestra en la Figura 4.5. Donde se encuentran los tres recuadros mencionados.

Figura 4.5: Página web resultante

Primer recuadro - logos.htm

En este primer recuadro, como se mencionó anteriormente se coloca un titulo inicial y se colocan las dos imágenes con los logos de la universidad. Por lo que no cuenta de una gran complejidad. Nuevamente, se utiliza el script del que se habló antes a modo de detección del tamaño de la pantalla, y con dicha información centrar correctamente las imágenes y el texto.

Segundo recuadro - mostrar.htm

En este recuadro se destacan 2 cosas. Por un lado, cuenta con fecha y hora y zona horaria, para saber si el programa inició correctamente. Por otro lado, se tiene un recuadro dentro del que se verán los frames a medida que se conecte el dispositivo móvil. La principal funcionalidad de esta página se encuentra en poder observar que recorrido se encuentra haciendo

el robot, mientras busca a su objetivo.

Tercer recuadro - botones.htm

Este tercer y último recuadro, posee las opciones para controlar al robot. Solo cuenta con tres funciones principales que son, iniciar o detener búsqueda y conectar dispositivo móvil con robot. Se podrían agregar otras, pero como es autónomo, no cumpliría con su objetivo. En la sección E.5.3, se agrega el código que ejecuta las acciones a medida que se presionan los botones

4.6. Programación del robot

Se puede observar a continuación un diagrama de flujos del código del robot.

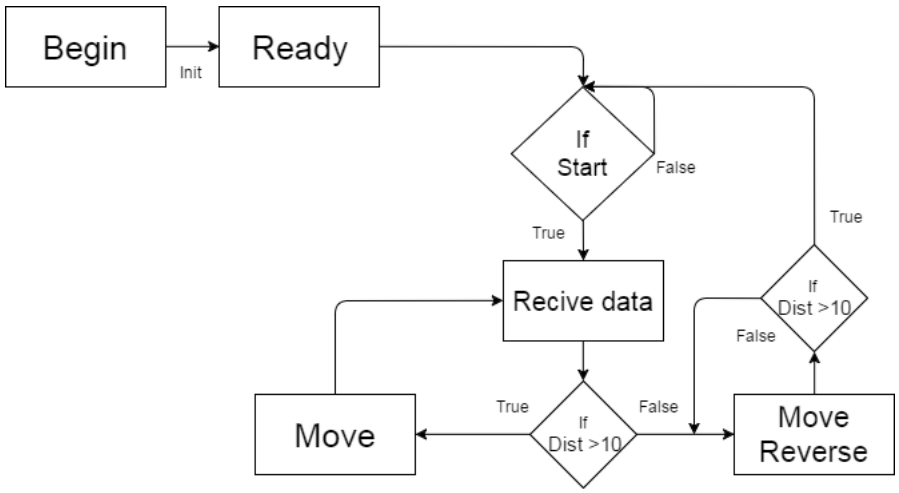

Figura 4.6: Diagrama de funcionamiento del robot

Dado que el envío de información es asíncrona, no fue agregada como un bloque, y será explicado más adelante.

Para que el funcionamiento del robot sea correcto, ciertos bloques deben tener mayor prioridad que otros. El de principal prioridad es el controlador de distancia, dado que si bien puede recibir un mensaje de movimiento, pero si tiene un objeto frente a él debe detenerse, y realizar in cambio de trayectoria.

4.6.1. Interconexión del sistema

Antes de explicar el Software implementado, se realizará una breve explicación de las conexiones utilizadas. En la Figura 4.7 se puede observar las conexiones necesarias para controlar los motores, recibir y enviar datos por Bluetooth y para obtener la distancia con el sensor de ultrasonido.

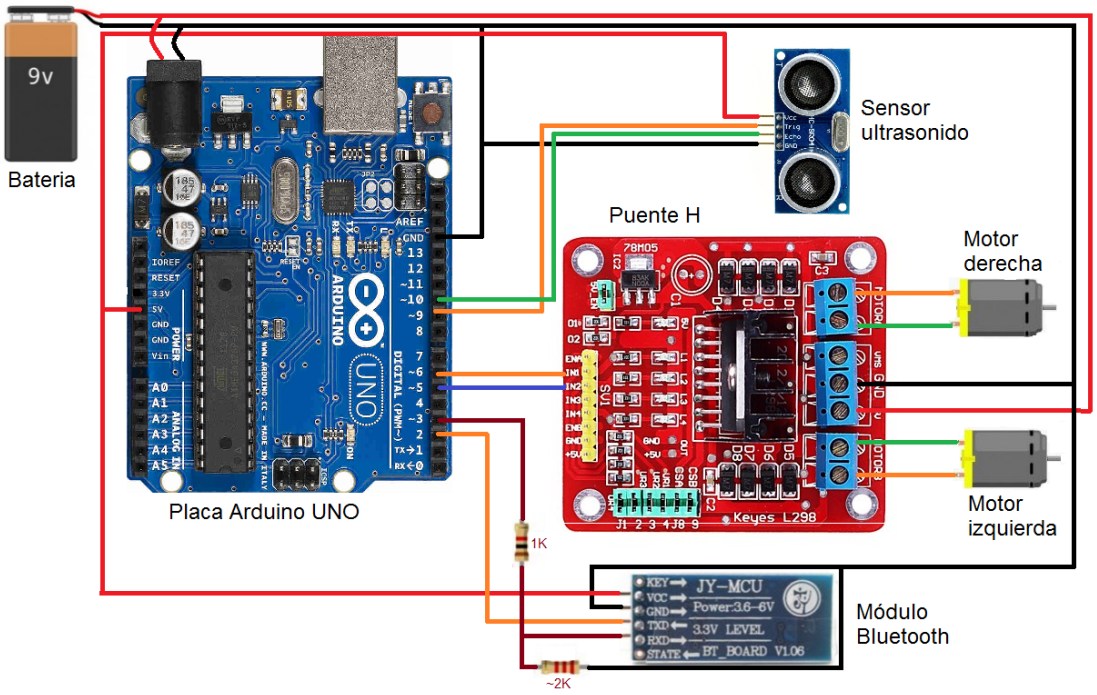

Figura 4.7: Conexiones de la placa arduino

Para el sensor de ultrasonido, se conectan los Pines 9 y 10 del Arduino al Trigger y Echo del sensor de ultrasonido respectivamente, además de la alimentación del sensor.

Los Pines 5 y 6 serán los que envíen la información del PWM al puente H que controlará los motores.

Por último para el Bluetooth, que recibirá información de control desde Android, deberá conectarse, ademas de la alimentación, el Pin 2 al TX (transmisión del módulo) y el Pin 3, mediante un divisor de tensión, al RX (recepción del módulo). El divisor resistivo es necesario, dado que el Arduino trabaja con niveles de 0-5V mientras que el módulo Bluetooth trabaja con niveles de 0-3.3V.

4.6.2. Inicialización del programa

Al iniciar el código, se ejecuta, la función *setup()*. Dicha función setea todas las variables necesarias para el funcionamiento y las inicializa, esta función se ejecuta solamente una vez. A continuación se muestra la inicialización.

```
//Seteo de variables
void setup() {
  // put your setup code here, to run once:
  TCCR0B = TCCR0B & 0b11111000 | 1; //setea frecuencia del PWM
   Serial.begin(9600);            //inicio de puerto serie
  BT.begin(9600);                 //Carro con bt no tiene el bt a tx y rx
   pinMode(motor_izq, OUTPUT);    //seteo de pin de motor izquierdo
   pinMode(motor_der, OUTPUT);    //seteo de pin de motor derecho
   pinMode(led,OUTPUT);           //seteo de pin de led testigo
   pinMode(trigPin, OUTPUT);     //Pin de trigger como salida
   pinMode(echoPin, INPUT);      //Pin de echo como entrada
   write_pwm(pot_stop, pot_stop); //inicio de motores frenados
   MsTimer2::set(300,sensor);     //Seteo con periodo de 500mSeg
   MsTimer2::start();        //Inicio del loop
}
```

A continuación se explicarán algunos de los seteos principales, necesarios para comprender el funcionamiento del código.

El seteo de registro $TCCR0B$ se utiliza para cambiar la frecuencia de trabajo del PWM.

En este proyecto se cuenta con dos puertos series. El puerto nombrado *Serial* que corresponde a un puerto serie físico para comunicación con la computadora (en caso de estar conectado) y el puerto BT, que es un puerto serie virtual que se encuentra conectado al módulo bluetooth. Luego, la función $Serial.begin()$ y $BT.begin()$ funcionan para inicializar las comunicaciones de ambos puertos. El valor que se ingresa entre paréntesis corresponde al Baud Rate (Véase E.3) con el que funciona la comunicación.

Luego aparecen los seteos de pines que se usan como entrada o salida según corresponda.

En la función $write\_pwm()$ que aparece al final del seteo cumple la función de iniciar al robot detenido, dado que por defecto Arduino no coloca un valor de detención (PWM = 127).

Por último, las funciones $MsTimer2 :: set()$ y $MsTimer2 :: start()$ cumplen la función de realizar el seteo y luego el inicio de una interrupción cada un cierto tiempo, en nuestro

caso se colocó cada $300mSeg$. Es decir, cada $300mSeg$ realizará la función *sensor*() que se explicará mas adelante.

4.6.3. Comunicación Bluetooth

Para realizar la comunicación por Bluetooth, el Arduino no posee una interrupción predefinida, por lo tanto, el modo de hacerlo es chequear continuamente si ingresó un carácter. Para ello, se coloca el llamado a la función dentro de la función *loop*() dado que se repetirá continuamente.

```
void loop() {
  // put your main code here, to run repeatedly:
    btEvent();  //chequeo continuamente si ingreso un nuevo dato
}
```

Cada vez que inicie la función *loop*(), ejecutara la función *btEvent*(), que puede observarse a continuación.

```
void btEvent(){
  //Controlo que el BT este activado
    if (BT.available() && controlProx){
      // Leo el puerto para controlar si hay datos
        char inChar = (char)BT.read();
        // Concateno el dato con lo obtenido anteriormente
        inputString += inChar;
        if (inChar == '\n'){
          /* Si llego un fin de mensaje
           * decodifico el mensaje y limpio la variable
           * para recibir un nuevo dato
          */
          decodificar_comandos();
          inputString = "";
        }
    }else if(!BT.available()){
        digitalWrite(led, LOW);
    } else{digitalWrite(led, HIGH);}
}
```

La función primero chequea si se encuentra disponible el bluetooth y luego intenta realizar una lectura de carácter. En el caso que el carácter sea $/n$ (carácter new line), ejecuta la función $decodificar\_comandos()$ la cual comprueba si corresponde a un comando válido y actúa en función al comando. Por último, limpia la variable $inputStrig$ para recibir un nuevo comando.

4.6.4. Detección de comandos

Una vez completada la lectura de comandos, es necesario decodificarlo para saber que función deber realizar el Arduino. Para ello, en la función $decodificar\_comandos()$ se toma una parte del comando recibido y se la compara con los comandos predefinidos. Dichos comandos pueden ser:

1. IDEN : Identificador de la placa Arduino.
2. LEFT : Moverse hacia la izquierda.
3. RIGHT : Moverse hacia la derecha.
4. CENTER : Moverse hacia adelante.
5. WAIT : Detenerse esperando nuevo comando.
6. STOP : Detener el movimiento.
7. LOOK : Iniciar la secuencia de búsqueda por el recinto.
8. START : Iniciar el movimiento.
9. BT-DONE : Bluetooth conectado.
10. BT-DISCONNECT : Bluetooth desconectado.
11. BACK : Retrocede haciendo un giro en caso de estar cerca de un objeto
12. GIRAR : Girar sobre si.
13. MOTOR : Mover con PWM distintos a los predefinidos.

Por lo tanto, la función que se muestra a continuación, solamente debe comparar los comandos predefinidos con el comando recibido y ejecutar la función que corresponda.

```
void decodificar_comandos(void) {
  //Identificar los comandos recibidos
  //Serial.println(inputString);
    if (inputString.substring(0, 6) == "CENTER" && !controlProx){
      write_pwm(pot_stop, pot_stop);
      Serial.println("Se encontro al objetivo");
    } else if (inputString.substring(0, 4) == "IDEN") {
      Serial.println("TrabajoFinalGelosi \r\n");
    } else if (inputString.substring(0, 4) == "LEFT") {
      if (startSet && cm>5) {left();}
    } else if (inputString.substring(0, 4) == "WAIT") {
      write_pwm(pot_stop,pot_stop);
    }else if (inputString.substring(0, 5) == "RIGHT") {
      if (startSet  && cm>5) {right();}
    } else if (inputString.substring(0, 6) == "CENTER") {
      if (cm>5) {center();}
    } else if (inputString.substring(0, 4) == "STOP") {
      look = 0;
      write_pwm(pot_stop, pot_stop);
      startSet =false;
    } else if (inputString.substring(0, 4) == "LOOK") {
      if (startSet) {searchEverywhere();}
    } else if (inputString.substring(0, 5) == "START"){
      look = 0;
      startSet =true;
    } else if (inputString.substring(0, 7) == "BT-DONE"){
      btConect =true;
      digitalWrite(led, HIGH);
    } else if (inputString.substring(0,5) == "MOTOR"){
      write_pwm(inputString.substring(5,8).toInt() + factor_pwm, inputString.
   substring(8,11).toInt() + factor_pwm);
    } else if (inputString.substring(0,13) == "BT-DISCONNECT"){
      digitalWrite(led, LOW);
    } else if (inputString.substring(0, 4) == "BACK"){
      girarSobreSi();
    }
}
```

4.6.5. Movimiento

Para realizar movimiento, todas las funciones hacen llamado a la función *write_pwm*(), esta función escribe el PWM deseado a los pines de los motores.

```
void write_pwm(int pwm_izq, int pwm_der) {
  analogWrite(motor_der, pwm_der);
  analogWrite(motor_izq, pwm_izq);
}
```

Luego, según el comando ingresado realiza un movimiento acorde, por lo tanto, cada función de movimiento solamente debe hacer llamado a la función de *write_pwm*() indicando que PWM deberá entregar a cada motor.

```
void left(){
  write_pwm(pot_rev_izq + factor_pwm,pot_fow_der + factor_pwm);
}
void right(){
  write_pwm(pot_fow_izq - factor_pwm,pot_rev_der - factor_pwm);
}
void center(){
  write_pwm(pot_fow_izq - factor_pwm,pot_fow_der + factor_pwm);
}
void girarSobreSi(){
  write_pwm(128,200); //Girar sobre sí.
}
```

En versiones anteriores, si no se encontraba el objetivo, se generaba una rutina de seguimiento desde Arduino. Para aumentar la velocidad y la precisión, esta rutina se implementó en Android. En la función *lookEverywhere*(), se selecciona entre tres posibles opciones, dependiendo de, la distancia, y del tiempo desde que inició la rutina.
En el caso de que se encuentre cerca de un obstáculo, ejecuta una rutina para evadirlo. Si no hay ningún obstáculo, se ejecuta una rutina de búsqueda dependiendo del tiempo, es decir, a medida que el tiempo avanza, cambia el sentido de búsqueda.

```
    private void lookEverywhere() {
        if (distancia < 100) {
```

```
            switch (last) {
                case 1:
                    if (!use2){
                        setPot(iFL, dFL);
                    }else{
                        setPot(iFL2, dFL2);
                    }
                    break;
                case 2:
                    if (!use2){
                        setPot(iFF, dFF);
                    }else{
                        setPot(iFF2, dFF2);
                    }
                    break;
                case 3:
                    if (!use2){
                        setPot(iFR, dFR);
                    }else{
                        setPot(iFR2, dFR2);
                    }
                    break;
                default:
                    if (!use2){
                        setPot(iFF, dFF);
                    }else{
                        setPot(iFF2, dFF2);
                    }
                    break;
            }
            timeInit = System.currentTimeMillis();
        }else {
            timeFin = System.currentTimeMillis();
            long time =timeFin - timeInit;
            if (time < 2000) {
                if (!use2){
                    setPot(iFR, dFR);
                }else{
                    setPot(iFR2, dFR2);
```

```
                }
            }else if (time < 6000){
                if (!use2){
                    setPot(iFF, dFF);
                }else{
                    setPot(iFF2, dFF2);
                }
            }else if (time < 10000){
                if (!use2){
                    setPot(iFL, dFL);
                }else{
                    setPot(iFL2, dFL2);
                }
            }else{
                timeInit = System.currentTimeMillis();
            }
        }
    }
```

La variable *use*2 es una variable auxiliar que de forma periódica actualiza los valores de PWM, dependiendo de todo lo mencionado anteriormente.

4.6.6. Interrupción de sensor de proximidad

Cada $500mSeg$ se ejecuta una función la cual registra si hay algún tipo de obstáculo. Lo que hará esta función es enviar un pulso de $10\mu Seg$ y esperar el eco con la función $pulseIn()$, esta función puede entregar un valor de tiempo que tardo el pulso en ir y volver, o 0 en caso de que no encuentre nada.

Luego, lo que se debe hacer, es controlar si el objetivo se encuentra a una distancia menor a $15cm$, con lo cual, si no es el objetivo deseado, deberá esquivarlo.

```
void sensor(){
  digitalWrite(trigPin, LOW);  //envio un pulso
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

```
7    duration = pulseIn(echoPin, HIGH);  //miro lo que tardo en regresar -
       Timeout 1000uSec (check)
8    cm = duration/29/2;  //calculo la distancia
9    //Serial.println(cm);
10   if (cm > 18 ||cm == 0){
11     count = count + 1;
12     controlProx =true;
13       //BT.print("V");
14     //Serial.println("V");
15     if (!startSet){
16       write_pwm(pot_stop,pot_stop);
17     }
18   }else{
19     write_pwm(pot_stop,pot_stop);
20     delay(10);
21     write_pwm(55,90);
22     //Serial.println("F");
23       //BT.print("F");
24     controlProx =false;
25     delay(500);
26   }
27   //BT.print(" - Dist = ");
28   if (cm >= 100){
29     BT.println(cm);
30   }else if (cm >= 10){
31     BT.print("0");
32     BT.println(cm);
33   } else {
34     BT.print("00");
35     BT.println(cm);
36   }
37 }
```

4.7. Resultados Obtenidos

4.7.1. Aplicación en Android

A continuación se explicarán y mostraran las funcionalidades de la aplicación y el significado de cada una de sus partes.

En un principio, luego de instalar la aplicación, aparecerán los dos iconos que se muestran en la Figura 4.8. La marcada con el número 1 corresponde a la aplicación para realizar el seguimiento. Mientras que la marcada con el número 2 corresponde a una aplicación necesaria para utilizar en forma correcta la librería de OpenCv.

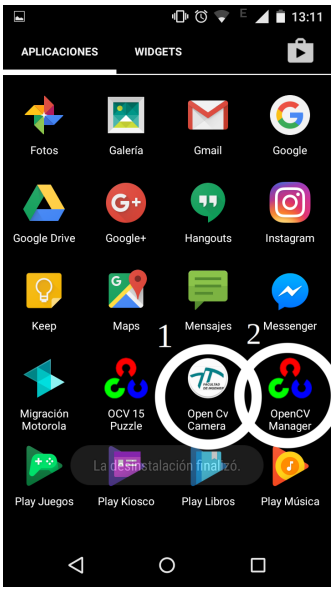

Figura 4.8: Iconos necesarios para la aplicación

Pantalla principal

Una vez seleccionada la aplicación para realizar el seguimiento, se podrá observar la siguiente pantalla.

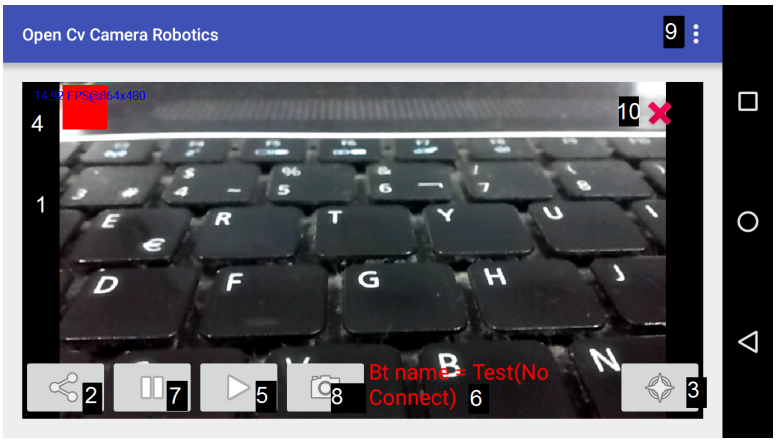

Figura 4.9: Pantalla principal al iniciar la aplicación

A continuación se realizará una explicación de las funciones de la aplicación en función de orden de uso. Siguiendo la enumeración de la Figura 4.9, con el Nro.1 se puede observar la pantalla que muestra los frames que se procesaron. A continuación (con el Nro.2) se observa el botón que permite realizar la conexión con el dispositivo Bluetooth seleccionado (la selección del dispositivo se realiza en el menú de opciones en caso de no utilizar el dispositivo por defecto).

Luego, se procede a la selección de un nuevo color, marcado con el Nro.3. Dicho color se podrá observar en la parte superior-izquierda de la pantalla, junto con la velocidad de FPS a la que se encuentra trabajando (Nro.4).

El botón Nro.5 permite iniciar la búsqueda en caso de no encontrarse conectado con el servidor. En caso de iniciar la búsqueda se podrá observar en el texto marcado con el Nro.6 el estado del Bluetooth y el tiempo desde que inició la búsqueda. En caso de desear detener la búsqueda se deberá presionar el botón Nro.7.

Luego, el botón Nro.8 es utilizado para iniciar el flash. Con el Nro.9 se observa el menú de opciones y por último con el Nro.10 un botón para cerrar la aplicación.

Menú de opciones

Si se presiona el menú aparecerá la siguiente pantalla.

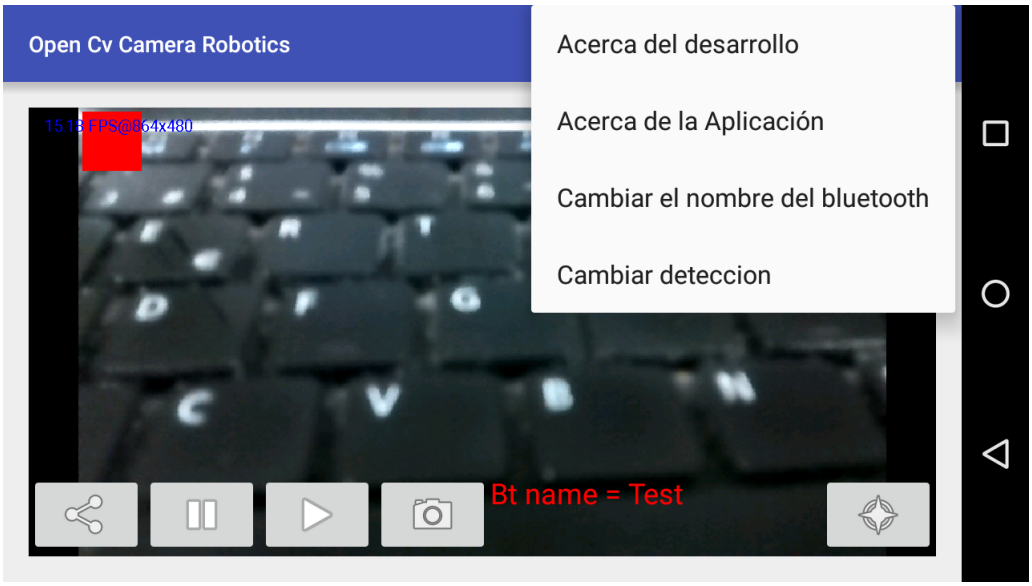

Figura 4.10: Pantalla principal al presionar el botón menú

Al seleccionar alguna de las opciones que aparecen, nos enviarán a una ventana nueva con la información deseada. En el caso de presionar *Acerca del desarrollo* nos enviará a una ventana con información sobre el desarrollo y contactos como se puede observar a continuación.

Figura 4.11: Pantalla de *Acerca del desarrollo*

Si en la ventana anterior presionamos la flecha nos regresará al modo de búsqueda. Luego, en la siguiente opción, *Acerca de la aplicación*, posee información de cual fue el objetivo de la aplicación, como puede observarse a continuación.

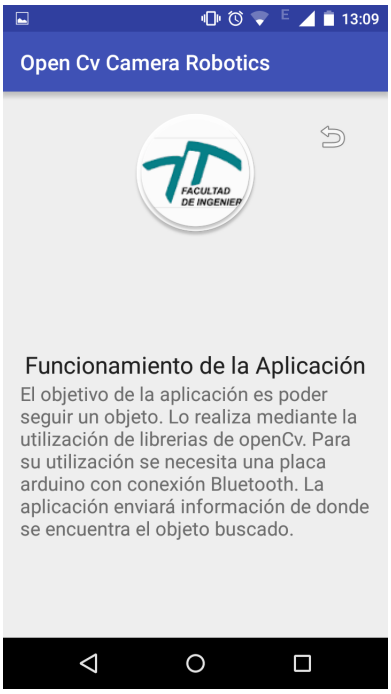

Figura 4.12: Pantalla de *Acerca de la aplicación*

Nuevamente, al presionar la flecha regresará a la pantalla inicial. En la siguiente opción que figura en la Figura 4.10, se puede observar *Cambiar el nombre del Bluetooh*, al presionar este botón, nos enviará a la siguiente pantalla.

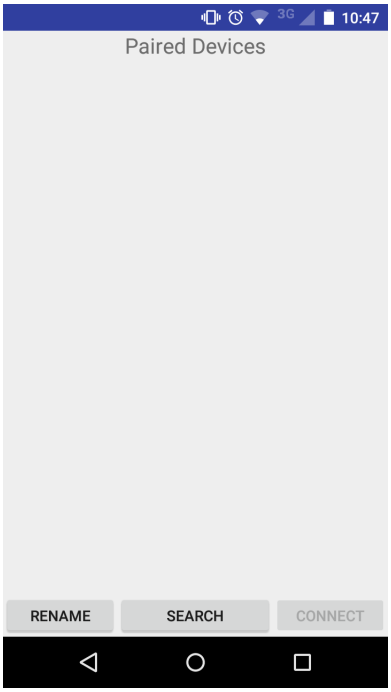

Figura 4.13: Pantalla de *Cambiar el nombre del Bluetooh*

Luego, en el caso de presionar el botón de búsqueda, aparecerán los dispositivos que se

encuentran en modo paring con el dispositivo de prueba como se observa en la Figura 4.14. Esta pantalla nos permite realizar la selección de un dispositivo y realizar la conexión. Además nos permite volver a la pantalla inicial.

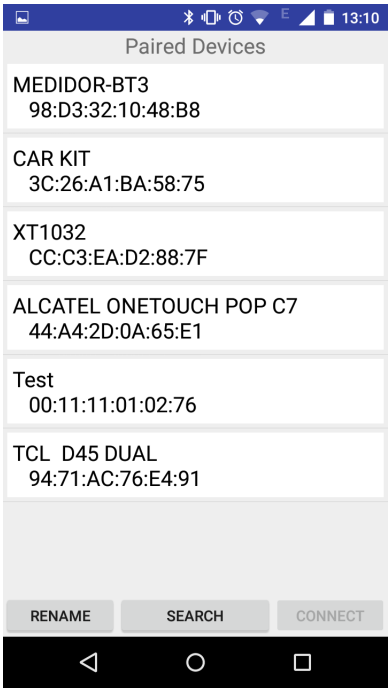

Figura 4.14: Pantalla de *Cambiar el nombre del Bluetooh - Con búsqueda activada*

Por último,al presionar el botón "rename"permite cambiarle el nombre en modo manual con un dispositivo no reconocido aún.

4.7.2. Aplicación WEB

Uno de los principales resultados a destacar en esta sección, está dado por la velocidad de transmisión y recepción de frames. Luego de pruebas y correcciones, se logró obtener (con la aplicación en pleno funcionamiento) una velocidad de transmisión de datos de entre 150 y 250 ms entre imagen e imagen. Es un resultado importante, dado que a su vez, el dispositivo móvil se encuentra, enviando y recibiendo datos por bluetooth, enviando y recibiendo datos por el servidor y recibiendo y procesando frames.

Otro resultado logrado, es una velocidad de respuesta de 350ms aproximadamente entre accionar un control de la página y que lo ejecute el robot. Dado que al presionar un botón, debe enviar información al servidor, el dispositivo debe obtener dicha información, luego procesarla y enviarla por bluetooth, y por último el robot debe recibir esa información, procesarla y ejecutar dicha acción.

Figura 4.15: Página web funcionando y recibiendo información

En la Figura 4.15, se puede observar que la pantalla tiene información del color que se ha seleccionado, y del objeto que busca.

Capítulo 5

Conclusiones y trabajos futuros

5.1. Conclusiones

La eficiencia del método de detección ha sido la esperada. El único problema con este método es la variación de los resultados dependiendo de las condiciones de luz. Aun así, en el entorno de trabajo, el robot ha tenido un buen comportamiento, con una buena velocidad de respuesta.
Siendo que el sistema funciona con la detección colores, se debe tener en cuenta que el recinto dentro del cual se mueve el robot se diferencie del objetivo a encontrar.
Otro ítem a tener en cuenta, es la detección de obstáculos, dado que el robot posee solamente un sensor de proximidad para evitar las colisiones frontales, produce que en ciertas condiciones, dicho sensor, puede no observar correctamente el obstáculo (si no se encuentra dentro del ángulo de detección del sensor) y colisionar.
Dado que se observó que, no solamente detecta el objetivo, sino se puede realizar el seguimiento del mismo y realizar el monitoreo en forma remota, se puede decir que se han logrado alcanzar los objetivos planteados al comienzo del proyecto.

5.2. Mejoras y trabajos a futuro

En los trabajos orientados a la robótica se pueden encontrar siempre mejoras. En este trabajo se dividirán las mejoras en tres grupos, para la aplicación Android, para el robot, y por último para la página web.

5.2.1. Aplicación en Android

1. Durante el desarrollo de la aplicación, se observó que la velocidad del dispositivo es un determinante el la velocidad de procesamiento de frames y envío de información.

2. Además, se logró observar que si se utilizara un dispositivo dedicado para este procesamiento, el rendimiento sería mejor. Dado que hay aplicaciones que se encuentran funcionando continuamente en segundo plano.

3. En cuanto al algoritmo de detección, se descubrió que si el procesamiento de los frames se hicieran a un menor nivel de programación (como el Native de Android), es posible encontrar objetivos a mayor distancia y con mayor precisión.

5.2.2. Robot - Arduino

1. El primer problema con el que se encontró fue el uso de baterías. Las que posee actualmente no tienen la suficiente carga para poder soportar el uso constantes de los motores durante un tiempo prolongado de uso.

2. Otro ítem importante, es el agregado de sensores de proximidad. Como se mencionó anteriormente, el robot posee una buena respuesta para evadir colisiones frontales, pero si el objetivo se encuentra fuera del ángulo de detección del único sensor, podría colisionar.

5.2.3. Página web

1. El principal conflicto es al cargar la nueva imagen a la página, se genera un “parpadeo”. Dicho parpadeo, no afecta a la utilización del sistema, pero puede resultar molesto al usuario.

2. Por otro lado, se podría mejorar el método de utilización del servidor. Dado que no se cuenta con mucha experiencia en el área de programación web, el método de cargar y descargar archivos de la base de datos no es óptima.

Apéndice A

Bluetooth

A.1. Bluetooth

Bluetooth es una especificación industrial para redes inalámbricas de área personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia. Está especialmente diseñado para dispositivos de bajo consumo que requieren un rango corto de emisión (distancia máxima de 10 metros). Los objetivos principales de la tecnología son:

1. Facilitar las comunicaciones entre los equipos móviles.
2. Eliminar los cableados y conectores entre dispositivos.
3. Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Estos dispositivos,están preparados para funcionar en la banca ISM de GHz. Funcionando con una modulación de espectro esparcido por salto en frecuencia (Frequency Hopping Spread Spectrum o FHSS). A su vez, para la transmisión de datos, se encuentra basado en el modo TDD (Time-División Duplexing), la cual permite funcionar en modo full-duplex para la transmisión de información.
Por otra parte, para la comunicación entre dispositivos, funcionan con el mecanismo de Master/Slave, en el cual uno se encarga de controlar la comunicación y enviar ordenes, pero ambos extremos, pueden transmitir información. Existen tres formas de conexión para estos dispositivos:

1. **Point-to-Point:** Es la comunicación mas común, donde existe una única conexión entre un maestro y un esclavo.
2. **Multi-Point:** En estos modos de accesos un maestro, controla a mas de un esclavo.
3. **Scatternet:** Se trata de una conexión entre varias piconet. En estos casos pueden existir un dispositivo que sea esclavo de más de un maestro, y maestros que a su vez, pueden ser esclavos de otra red.

A fines de nuestro proyecto, se utilizará el modo de conexión point-to-point, dado que el dispositivo móvil controlará al Arduino. Por lo tanto, el dispositivo móvil funcionará como maestro, mientras que el robot hará de esclavo.

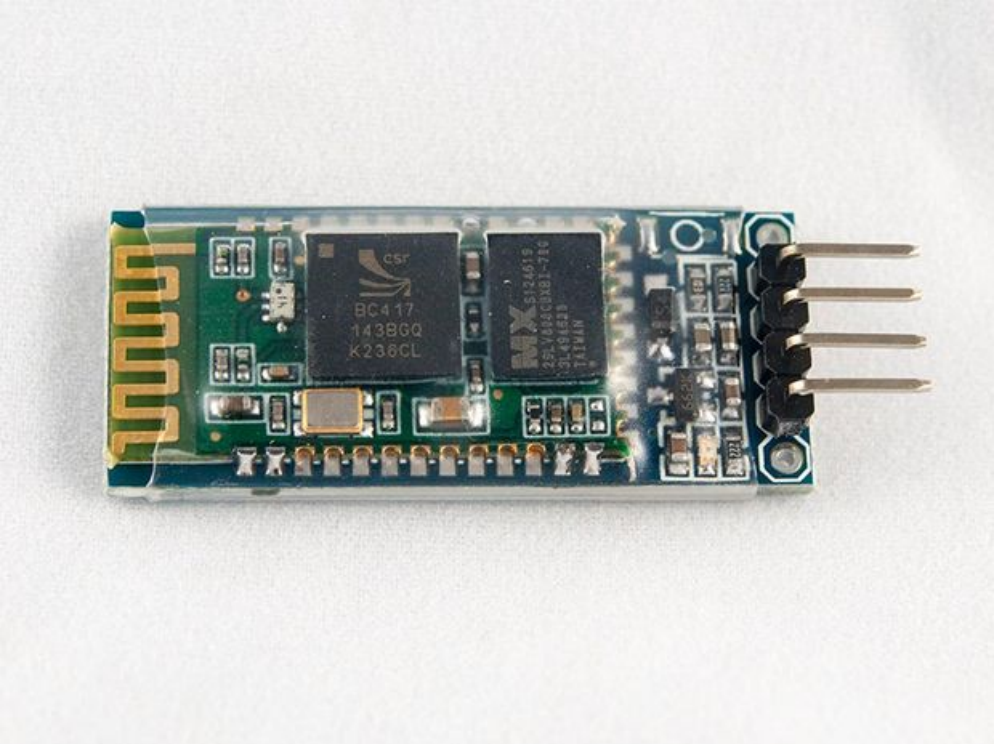

Figura A.1: Modulo Bluetooth utilizado - HC-06

A.2. Modulo Bluetooth

El dispositivo bluetooth seleccionado, posee las siguientes características.

1. Wireless transceiver
    1. Sensitivity (Bit error rate) can reach $-80dBm$

2. The change range of output?s power: $-4 - +6dBm$.

2. Function description

    1. Has an EDR module; and the change range of modulation depth: $2Mbps - 3Mbps$.
    2. Has a build-in $2{,}4GHz$ antenna; user needn't test antenna.
    3. Has the external $8Mbit$ FLASH
    4. Can work at the low voltage ($3{,}1V \approx 4{,}2V$). The current in pairing is in the range of $30 \approx 40mA$.
    5. The current in communication is $8mA$.
    6. Standard HCI Port (UART or USB)
    7. USB Protocol: Full Speed $USB1{,}1$, Compliant With 2,0
    8. This module can be used in the SMD.
    9. It's made through RoHS process.
    10. The board PIN is half hole size.
    11. Has a$2{,}4GHz$ digital wireless transceiver.
    12. Bases at CSR BC04 Bluetooth technology.
    13. Has the function of adaptive frequency hopping.
    14. Small ($27mm \times 13mm \times 2mm$)
    15. Peripherals circuit is simple.
    16. It's at the Bluetooth class 2 power level.
    17. Storage temperature range: $-40°C - 85°C$, work temperature range: $-25°C - +75°C$
    18. Any wave inter Interference: $2{,}4MHz$, the power of emitting: $3dBm$.
    19. Bit error rate: 0.

3. Low power consumption

4. Has high-performance wireless transceiver system

5. Low Cost

6. Application fields:
    1. Bluetooth Car Handsfree Device
    2. Bluetooth GPS
    3. Bluetooth PCMCIA , USB Dongle
    4. Bluetooth Data Transfer
7. Software: CSR

A continuación se puede observar un diagrama de la distribución de pines, los cuales serán explicados luego.

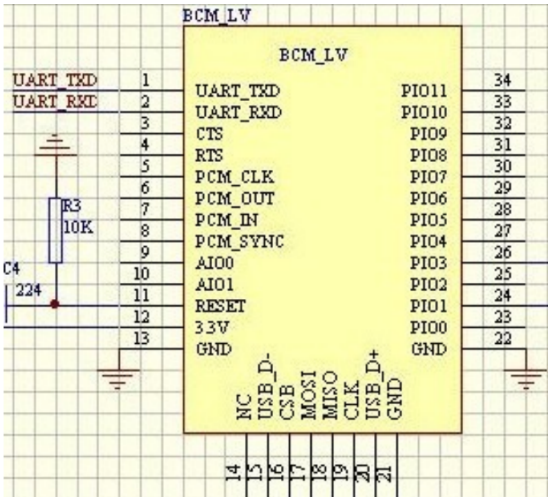

Figura A.2: Distribución de pines con nombres

PIN name	PIN #	Pad type	Description
GND	13 21 22	VSS	Ground pot
1V8	14	VDD	Integrated 1.8V (+) supply (1.7-1.9V)
VCC	12	3.3V	
AIO0	9	Bi-Directional	Programmable input/output line
AIO1	10	Bi-Directional	Programmable input/output line
PIO0	23	Bi-Directional RX EN	Programmable input/output line
PIO1	24	Bi-Directional TX EN	Programmable input/output line
PIO2	25	Bi-Directional	Programmable input/output line
PIO3	26	Bi-Directional	Programmable input/output line
PIO4	27	Bi-Directional	Programmable input/output line
PIO5	28	Bi-Directional	Programmable input/output line
PIO6	29	Bi-Directional	Programmable input/output line
PIO7	30	Bi-Directional	Programmable input/output line
PIO8	31	Bi-Directional	Programmable input/output line
PIO9	32	Bi-Directional	Programmable input/output line
PIO10	33	Bi-Directional	Programmable input/output line
PIO11	34	Bi-Directional	Programmable input/output line
RESETB	11	CMOS Input	
$UART - RTS$	4	CMOS output	UART request to send, active low
$UART - CTS$	3	CMOS input	UART clear to send, active low
$UART - RX$	2	CMOS input	UART Data input
$UART - TX$	1	CMOS output	UART Data output
$SPI - MOSI$	17	CMOS input	Serial peripheral interface data input
$SPI - CSB$	16	CMOS input	Chip select for serial peripheral interface
$SPI - CLK$	19	CMOS input	Serial peripheral interface clock
$PI - MISO$	18	CMOS input	Serial peripheral interface data Output
$USB-$	15	Bi-Directional	
$USB+$	20	Bi-Directional	
1.8V	14		1.8V external power supply input
$PCM - CLK$	5	Bi-Directional	
$PCM - OUT$	6	CMOS output	
$PCM - IN$	7	CMOS Input	
$PCM - SYNC$	8	Bi-Directional	

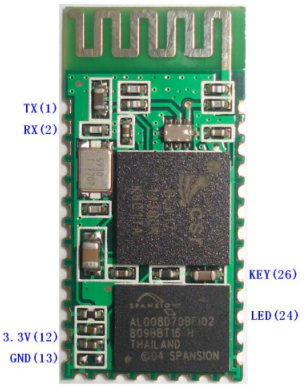

Figura A.3: Modulo utilizado

A continuación se puede observar, una lista de los comandos para configuración principales del mismo.

Comando	Descripción
AT /r/n	Comando de prueba, debe responder con OK /r/n
AT+ROLE=1 /r/n	Comando para colocar el módulo en modo Maestro (Master)
AT+ROLE=0 /r/n	Comando para colocar el módulo en modo Esclavo (Slave)
AT+VERSION? /r/n	Obtener la versión del firmware
AT+UART=115200,1,2 /r/n	Configurar el modo y velocidad de transmisión
AT+PIO=10,1 /r/n	Colocar el pin de IO de propósito general a nivel alto

Apéndice B

Arduino

B.1. Arduino

Arduino es una compañía de hardware libre, y comunidad tecnológica, que diseña y manufactura placas de desarrollo de hardware y software compuesta respectivamente por circuitos impresos que integran un microcontrolador, y un entorno de desarrollo (IDE) en donde se programa cada placa. Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios. Toda la plataforma, tanto para sus componentes de hardware como de software son liberados bajo licencia de código abierto que permite libertad de acceso a los mismos.

El hardware consiste en una placa de circuito impreso con un microcontrolador, usualmente Atmel AVR, puertos digitales y analógicos de entrada/salida, los cuales pueden conectarse a placas de expansión (shields) que amplían las características de funcionamiento de la placa Arduino. Asimismo posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación serie con el computador.

Por otro lado, el software consiste en un entorno de desarrollo (IDE) basado en el entorno de Processing y lenguaje de programación basado en Wiring, así como en el cargador de arranque (bootloader) que es ejecutado en la placa. El microcontrolador de la placa se programa a través de un computador, haciendo uso de comunicación serie mediante un convertidor de niveles RS-232 a TTL serie.

La primer placa Arduino fue introducida en el 2005, ofreciendo un bajo costo y facilidad de uso para novatos y profesionales buscando desarrollar proyectos interactivos con su entorno mediante el uso de actuadores y sensores. A partir de octubre de 2012, se incorporaron nuevos

modelos de placas de desarrollo que hacen uso de microcontroladores Cortex M3, ARM de 32 bits, que coexisten con los originales modelos que integran microcontroladores AVR de 8 bits. ARM y AVR no son plataformas compatibles en cuanto a su arquitectura y por lo tanto su set de instrucciones, pero se pueden programar y compilar bajo el IDE predeterminado de Arduino sin ningún cambio.

Las placas Arduino están disponibles de forma ensambladas o en forma de Kits "Hazlo tu mismo"(por sus siglas en inglés «DIY»). Los esquemáticos de diseño del Hardware están disponibles bajo licencia Libre, permitiendo a cualquier persona crear su propia placa Arduino sin necesidad de comprar una prefabricada. Adafruit Industries estimó a mediados del año 2011 que alrededor de 300,000 placas Arduinos habían sido producidas comercialmente, y en el año 2013 estimó que alrededor de 700.000 placas oficiales de la empresa Arduino estaban en manos de los usuarios.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software tal como Adobe Flash, Processing, Max/MSP, Pure Data, etc. Una tendencia tecnológica es utilizar Arduino como tarjeta de adquisición de datos desarrollando interfaces en software como JAVA, Visual Basic y LabVIEW. Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

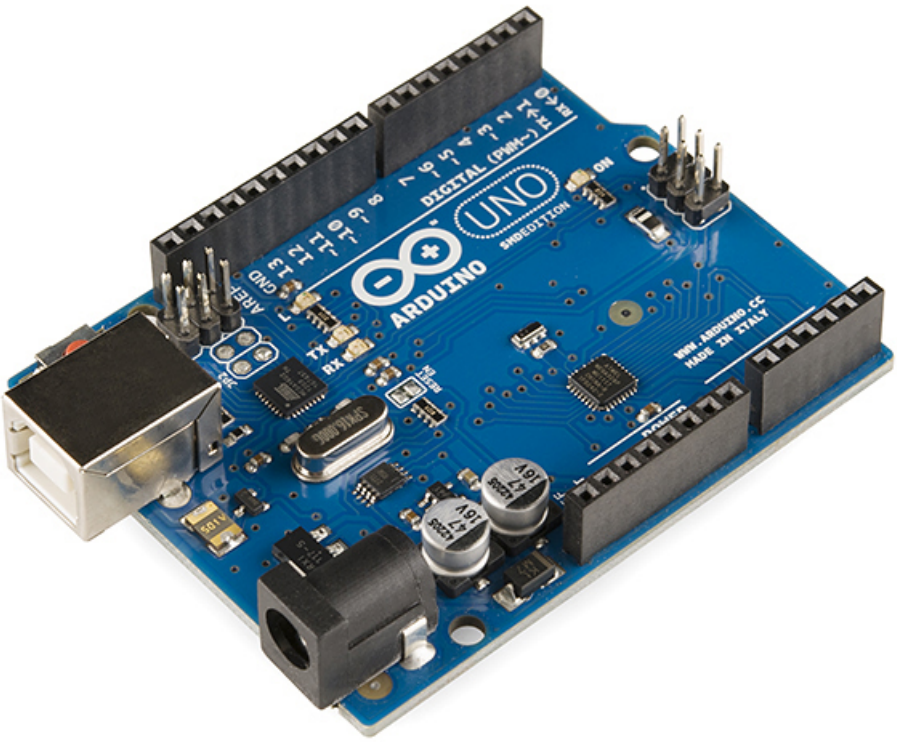

Figura B.1: Placa Arduino UNO

B.2. Entorno de programación

Para programar la placa es necesario descargarse de la página web de Arduino el entorno de desarrollo (IDE). Se dispone de versiones para Windows y para MAC, así como las fuentes para compilarlas en LINUX. En la Figura 2 se muestra el aspecto del entorno de programación. En el caso de disponer de una placa USB es necesario instalar los drivers FTDI. Estos drivers vienen incluidos en el paquete de Arduino mencionado anteriormente. Existen en la web versiones para distintos sistemas operativos.

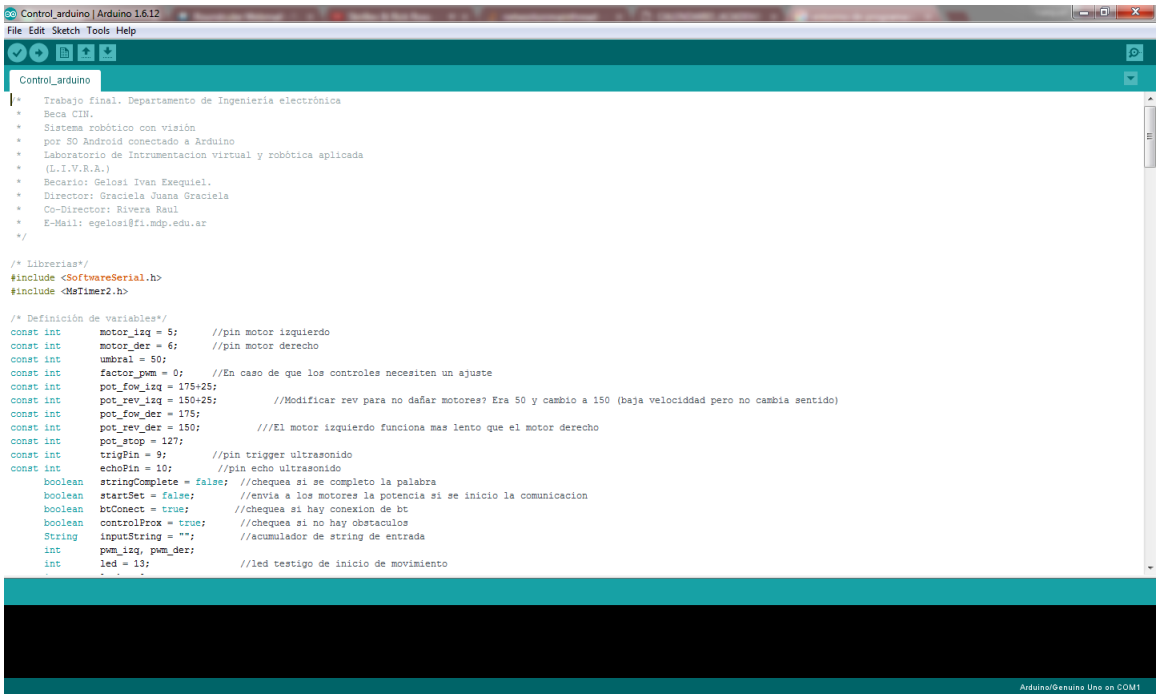

Figura B.2: IDE Arduino

Lo primero que tenemos que hacer para comenzar a trabajar con el entorno de desarrollo de Arduino es configurar las comunicaciones entre la placa Arduino y el PC. Para ello deberemos abrir en el menú "Tools"la opción "Serial Port". En esta opción deberemos seleccionar el puerto serie al que está conectada nuestra placa. En Windows, si desconocemos el puerto al que está conectado nuestra placa podemos descubrirlo a través del Administrador de dispositivos (Puertos COM & LPT/ USB Serial Port).

Apéndice C

Android

C.1. Programación en Android

El Desarrollo de Programas para Android se hace habitualmente con el lenguaje de programación similar a Java y el conjunto de herramientas de desarrollo SDK (SDK, Software-Development Kit), pero hay otras opciones disponibles. En Julio de 2013 existían más de 1.000.000 de aplicaciones contabilizadas para Android, con aproximadamente 25 mil millones de descargas. La plataforma Android ha crecido hasta ser una de las preferidas por los desarrolladores para plataformas móviles. Un estudio de junio del 2011 indica que el 67 % de los desarrolladores para móviles utilizaban la plataforma, en el momento de su publicación. En el segundo trimestre del 2012, se habían vendido alrededor de 105 millones de teléfonos Android, con un porcentaje del 68 % de las ventas de teléfonos inteligentes hasta esa fecha. El SDK (Software Development Kit) de Android, incluye un conjunto de herramientas de desarrollo.9 Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen GNU/Linux, Mac OS X 10.5.8 o posterior, y Windows XP o posterior. También puede utilizarse el propio sistema Android para desarrollos utilizando las aplicaciones AIDE - Android IDE - Java, C++(app) [AIDE - Android IDE - Java, C++] y el editor de Java. La plataforma integral de desarrollo (IDE, Integrated Development Environment) soportada oficialmente es Android Studio junto con el complemento ADT (Android Development Tools plugin). Además, los programadores pueden usar un editor de texto para escribir ficheros Java y XML y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit y Apache Ant) para crear y depurar aplicaciones, así como con-

trolar dispositivos Android que estén conectados (es decir, reiniciarlos, instalar aplicaciones en remoto, etc.).10

Las Actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad.11

Una aplicación Android está compuesta por un conjunto de ficheros empaquetados en formato .apk y guardada en el directorio /data/app del sistema operativo Android (este directorio necesita permisos de superusuario, root, por razones de seguridad). Un paquete APK incluye ficheros .dex 12 (ejecutables Dalvik, un código intermedio compilado), recursos, etc.

C.2. Entorno de programación Android Studio

Android Studio es un entorno de desarrollo integrado para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

Está basado en el software IntelliJ IDEA de JetBrains, y es publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux. [12] [13]

Las características que posse dicho entorno son:

1. Renderización en tiempo real
2. Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.
3. Soporte para construcción basada en Gradle.
4. Refactorización especifica de Android y arreglos rápidos.
5. Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, y otros problemas.
6. Plantillas para crear diseños comunes de Android y otros componentes.

- Soporte para programar aplicaciones para Android Wear.

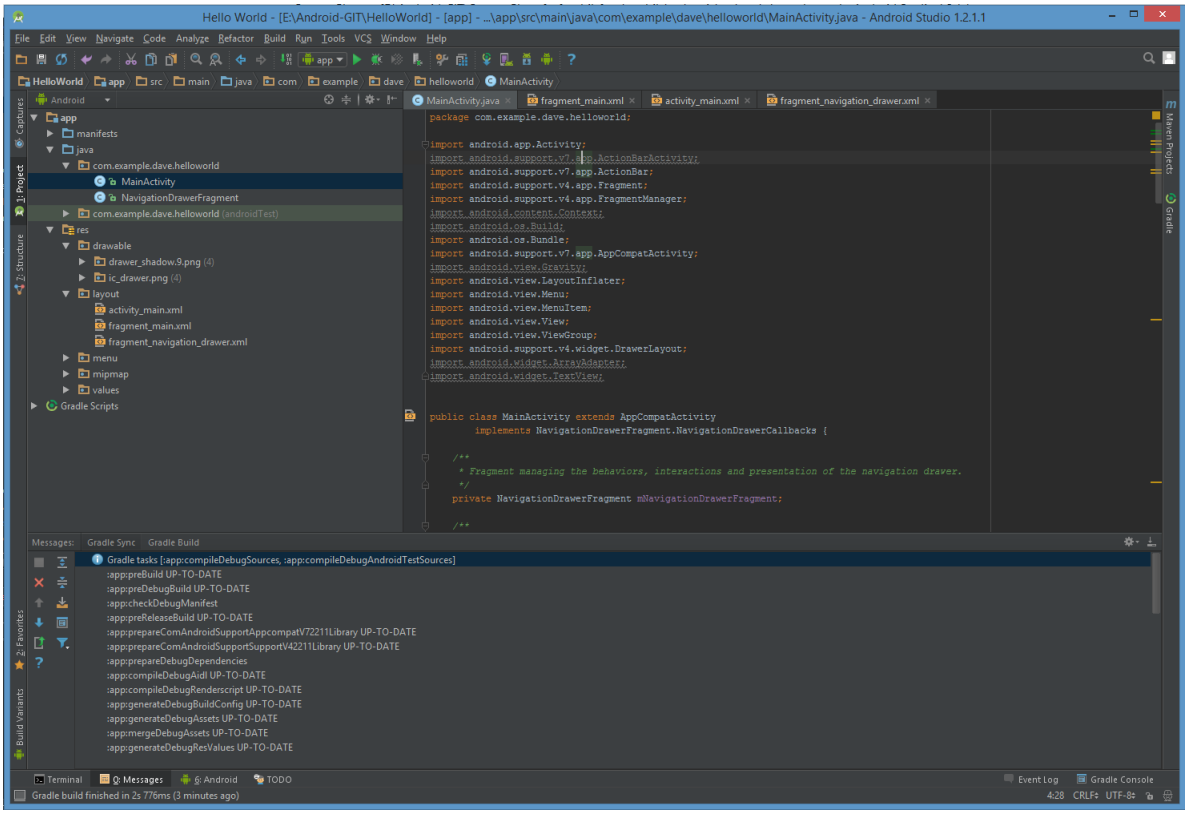

Figura C.1: IDE Android Studio

C.3. Librería android.hardware.camera

Dentro del entorno de desarrollo de Android, el acceso a todas las funcionalidades parte de librerías u objetos. En los mismos se tiene acceso a todo un conjunto de funciones predeterminadas, las cuales pueden ser editadas en caso de desearlo por preferencia del programador. En particular, la clase *android.hardware.camera* es utilizada para configurar las opciones de captura, iniciar/detener una previsualización de la imagen, tomar imágenes, obtener frames, y grabar vídeos. Esta clase permite el uso de los servicios de cámara ofrecidos por los dispositivos. Si bien en las últimas versiones de Android (a partir del API 21, Véase C.8) se lanzó la mejora en la clase *android.hardware.camera2*, la plataforma permite el uso de compatibilidad hacia atrás y por lo tanto, esta librería puede seguir siendo utilizada. La nueva librería permite un mejor manejo de los dispositivos y una mayor velocidad en la captura de frames, pero al utilizarla obligaría a que la aplicación sea utilizada por los dispositivos móviles mas recientes.

Está librería permite un uso simple de la cámara, pero es necesario aclarar, que las funciones pueden ser accedidas solamente si dentro del archivo de configuración (manifest) fue declarado el uso de la cámara, flash o vídeo. Esto se puede observar cuando se realiza la instalación de una aplicación en un dispositivo Android, aparecerá un cartel con una lista de permisos que requiere la aplicación.

C.4. Librería Native (Native development kit)

El NDK permite instalar bibliotecas escritas en C, C++ y otros lenguajes, una vez compiladas para ARM, MIPS o código nativo x86. Los programas Java corriendo en la máquina virtual Dalvik (Dalvik VM) pueden llamar a clases nativas por medio de la función System.loadLibrary, que forma parte de las clases estándares Java en Android.
Se pueden compilar e instalar aplicaciones completas utilizando las herramientas de desarrollo tradicionales. Sin embargo, según la documentación de Android, NDK no debe utilizarse para desarrollo, simplemente porque el programador prefiera programar en C/C++, ya que la utilización del NDK aumenta la complejidad sin que la mayor parte de las aplicaciones obtengan ningún beneficio por ello.
El depurador ADB proporciona un shell root en el Simulador de Android que permite cargar y ejecutar código nativo ARM, MIPS o x86. Este código puede compilarse con GCC, o el compilador C++ de Intel en un ordenador personal normal. La ejecución de código nativo es difícil porque Android utiliza una biblioteca de C propia (libc, llamada Bionic). La biblioteca gráfica que utiliza Android para controlar el acceso a este dispositivo se llama Skia Graphics Library (SGL), disponible con licencia de código abierto. Skia tiene implementaciones en win32 y Unix, permitiendo el desarrollo multiplataforma de aplicaciones, y es el motor de gráficos que soporta al navegador web Google Chrome.
NDK está basado en la línea de comandos, y al contrario que el desarrollo con Eclipse, requiere la invocación manual de comandos para construir, cargar y depurar las aplicaciones. Hay herramientas de terceros que integran el NDK con Eclipse y Visual Studio.

C.5. Librería OpenCv4Android

OpenCV [14] es una librería de visión por computador gratuita y de código abierto, desarrollada originalmente por Intel. La librería está escrita en los lenguajes C y C++ y funciona

en Windows, Linux y Mac OS X.
Su principal objetivo es proporcionar una interfaz de simple uso para la construcción de forma rápida aplicaciones complejas de visión por computador; está especialmente diseñada para ser computacionalmente eficiente, con especial énfasis en aplicaciones a tiempo real, aprovechando las ventajas de la programación paralela.
La librería contiene cerca de 500 funciones que abarcan diversas áreas de la visión por computador. OpenCV también integra una completa librería de machine learning (MLL) centrada en análisis estadístico para pattern recognition y clustering, que resulta muy útil para algunas tareas de computer vision. Desde que fue lanzado en 1999, OpenCV ha sido utilizado en multitud de aplicaciones, productos e investigaciones relacionados con la visión artificial. Su publicación bajo licencia BSD permite que se puedan construir productos comerciales usando la librería sin obligación de que el producto sea open source o de aportar mejoras a la comunidad. En Julio de 2011, OpenCV comenzó a ofrecer soporte para Android con la liberación de la versión 2.3. Para este proyecto usaremos la versión 2.5 de OpenCV4Android. Utilizar esta librería nos permitirá trabajar utilizando estructuras de datos, funciones y algoritmos implementados específicamente para el procesado de imágenes, lo que facilitará el manejo de las imágenes y su procesamiento en la aplicación. La contraparte es que se debe sacrificar una parte del tiempo en aprender a utilizar la librería antes de empezar a implementar una aplicación compleja. Aún así, está más que justificada la utilización de OpenCV para llevar a cabo la parte de visión por computador de este proyecto.

C.5.1. Agregar librería de OpenCv a un proyecto Android

Dado que la librería de OpenCv no se encuentra dentro del paquete Android Studio, debe ser incluida en forma manual, y esto suele ser problemático en caso que no se sigan correctamente los pasos. Por dicho motivo, a continuación se encuentran enumerados los pasos a seguir.

1. Descargar la ultima versión de OpenCv para Android desde el sitio web `www.OpenCv.org` y descomprimir el archivo.
2. Importar la librería a Android Studio. Para ello ir a: File → New → Import Module, y seleccionar la carpeta sdk/java dentro del archivo que se descomprimió anteriormente.
3. Actualizar el build.gradle. Luego de la primer actualización automática, es probable que pida instalar algunos complementos adicionales.

4. Agregar la dependencia al modulo importado. Para ello, ir a: Application → Module Settings, y seleccionar la ventana *Dependencias*. Allí, hacer seleccionar el botón + del lado derecho y elegir la opción *Module Dependency*.
5. Copiar la carpeta *libs* que se encuentra dentro de sdk/native dentro de app/src/main.
6. Por último, renombrar la carpeta *libs* como *jniLibs*.

C.6. Librería procesamiento de circulos

C.6.1. Hough Circle Transform

Encuentra círculos en una imagen en escala de grises usando la transformada de Hough. La función puede recibir los siguientes parámetros de Android:

1. **image**: Imagen en escala de grises.
2. **method**: Método de detección usado generalmente se utiliza el método $CV\_HOUGH\_GRADIENT$ ya que es el único método disponible actualmente en OpenCV.
3. **dp**: Resolución de la matriz de acumulación, si dp=1 la matriz tiene la misma resolución que la imagen de entrada (image).
4. **minDist**: Mínima distancia entre centros de los círculos.
5. **circles**: Vector de salida que contiene las coordenadas de centro y el radio que define una circunferencia, cada circunferencia está definida por un vector de tres elementos (x, y , radio).
6. **param1**: Umbral mayor de los dos umbrales fijados en Canny.
7. **param2**: Número de pixeles seguidos por los que debe pasar un circulo para ser considerado como tal.
8. **minRadius**: Radio mínimo que tendrán las circunferencias a encontrarse.
9. **maxRadius**: Radio máximo que tendrán las circunferencias a encontrarse

C.6.2. Circle

La utilidad de esta función es realizar el dibujo de un circulo sobre una imagen. Para ello, es necesario entregarle como mínimo tres parámetros, el centro (x,y) y el radio del circulo. Las opciones que permite ingresarle son las siguientes.

1. **img**: Imagen en la cual será dibujado el circulo.
2. **center**: Centro del circulo.
3. **radius**: Radio del circulo.
4. **color**: Color del circulo.
5. **thickness**: Espesor del circulo a dibujar.
6. **lineType**: Tipo de linea con el que se dibujará el circulo.

C.7. Procesos y subprocesos

Cuando un componente de la aplicación se inicia y la aplicación no tiene ningún otro componente ejecutándose, el sistema de Android inicia un nuevo proceso de Linux para la aplicación con un único subproceso de ejecución. De manera predeterminada, todos los componentes de la misma aplicación se ejecutan en el mismo proceso y subproceso (que se denomina "subproceso principal"). Si el componente de una aplicación se inicia y ya existe un proceso para la aplicación (porque otro componente de la aplicación existe), el componente se inicia dentro de ese proceso y usa el mismo subproceso de ejecución. Sin embargo, puedes configurar que diferentes componentes de la aplicación se ejecuten en procesos separados y puedes crear subprocesos adicionales para cualquier proceso.

C.8. API en Android

Los API hacen referencia a la versión y las utilidades de cada sistema operativo en Android, se pueden observar en la siguiente tabla una relación entre las API y las versiones con sus nombres comerciales.

Nombre código	Número de versión	Fecha de lanzamiento	API
	1.0	23 de Septiembre, 2008	1
	1.1	9 de Febrero, 2009	2
Cupcake	1.5	27 de Abril, 2009	3
Donut	1.6	15 de Septiembre, 2009	4
Eclair	2.0/2.1	26 de Octubre, 2009	5/7
Froyo	2.2/2.2.3	20 de Mayo, 2010	8
Gingerbread	2.3/2.3.7	6 de Diciembre, 2010	9/10
Honeycomb1	3.0/3.2.6	22 de Febrero, 2011	11/13
Ice Cream Sandwich	4.0/4.0.4	18 de Octubre, 2011	14/15
Jelly Bean	4.1/4.3.1	9 de Julio, 2012	16/18
KitKat	4.4/4.4.4, 4.4W/4.4W.2	31 de Octubre, 2013	19/20
Lollipop	5.0/5.1.1	12 de Noviembre, 2014	21/22
Marshmallow	6.0/6.0.1	5 de Octubre, 2015	23
Nougat	6.x Previa de desarrollador 4	15 de junio, 2016	24

C.9. Dispositivo de prueba

El dispositivo utilizado como prueba es un Motorola Moto G de primera generación, cuyas características relevantes para el funcionamiento de la aplicación pueden observarse a continuación.

General	
Dimensiones	129.9 x 65.9 x 11.6 mm
Peso	143 gr
Sistema operativo	Android 5.0.2 Lollipop
Chipset	Qualcomm MSM8226 Snapdragon 400
Procesador	1.2 GHz Quad Core
Memoria	8 GB
Bluetooth	4.0 con A2DP/LE
Pantalla	
Pulgadas	4.5
Resolución	720 x 1280 pixel
Tipo	IPS LCD
Touchscreen	Capacitiva
Colores	16 millones
Cámara	
Megapixel	5Mp
Resolución	2592 x 1944 pixel
Zoom	Digital 4 x
Máx FPS	30FPS

Figura C.2: Dispositivo de prueba utilizado

Apéndice D

Diseño WEB

D.1. PHP

Figura D.1: Logo PHP

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Fue creado originalmente por Rasmus Lerdorf en el año 1995. Actualmente el lenguaje sigue siendo desarrollado con nuevas funciones por el grupo PHP. Este lenguaje forma parte del software libre publicado bajo la licencia PHP, que es incompatible con la Licencia Pública General de GNU debido a las restricciones del uso del término PHP.

El intérprete de PHP solo ejecuta el código que se encuentra entre sus delimitadores. Los delimitadores más comunes son ¡?php para abrir una sección PHP y ?¿para cerrarla. El propósito de estos delimitadores es separar el código PHP del resto de código, como por ejemplo el HTML.28

Las variables se prefijan con el símbolo del dólar ($) y no es necesario indicar su tipo. Las variables, a diferencia de las funciones, distinguen entre mayúsculas y minúsculas. Las cadenas

de caracteres pueden ser encapsuladas tanto en dobles comillas como en comillas simples, aunque en el caso de las primeras, se pueden insertar variables en la cadena directamente, sin necesidad de concatenación.
Los comentarios se pueden escribir bien con dos barras al principio de la línea, o con una almohadilla. También permite comentarios multi-línea encapsulados en /* */.
En cuanto a las palabras clave, PHP comparte con la mayoría de otros lenguajes con sintaxis C las condiciones con if, los bucles con for y while y los retornos de funciones. Como es habitual en este tipo de lenguajes, las sentencias deben acabar con punto y coma (;).

D.2. HTML

Figura D.2: Logo HTML

HTML, sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del World Wide Web Consortium (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la World Wide Web (WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.
El lenguaje HTML basa su filosofía de desarrollo en la diferenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene solamente texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

Sin embargo, a lo largo de sus diferentes versiones, se han incorporado y suprimido diversas características, con el fin de hacerlo más eficiente y facilitar el desarrollo de páginas web compatibles con distintos navegadores y plataformas (PC de escritorio, portátiles, teléfonos inteligentes, tabletas, vipers etc.) No obstante, para interpretar correctamente una nueva versión de HTML, los desarrolladores de navegadores web deben incorporar estos cambios y el usuario debe ser capaz de usar la nueva versión del navegador con los cambios incorporados. Normalmente los cambios son aplicados mediante parches de actualización automática (Firefox, Chrome) u ofreciendo una nueva versión del navegador con todos los cambios incorporados, en un sitio web de descarga oficial (Internet Explorer). Por lo que un navegador desactualizado no será capaz de interpretar correctamente una página web escrita en una versión de HTML superior a la que pueda interpretar, lo que obliga muchas veces a los desarrolladores a aplicar técnicas y cambios que permitan corregir problemas de visualización e incluso de interpretación de código HTML. Así mismo, las páginas escritas en una versión anterior de HTML deberían ser actualizadas o reescritas, lo que no siempre se cumple. Es por ello que ciertos navegadores todavía mantienen la capacidad de interpretar páginas web de versiones HTML anteriores. Por estas razones, todavía existen diferencias entre distintos navegadores y versiones al interpretar una misma página web.

D.3. JavaScript

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Figura D.3: Logo JavaScript

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque existe una forma de JavaScript del lado del servidor(Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

Desde el 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1, una versión de javascript. Los navegadores más antiguos soportan por lo menos ECMAScript

3. La sexta edición se liberó en julio del 2015.

JavaScript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo, Java y JavaScript tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

D.4. CSS

Hojas de estilo en cascada (o CSS, siglas en inglés de Cascading Stylesheets) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado . Es muy usado para establecer el diseño visual de las páginas web, e interfaces de usuario escritas en HTML o XHTML; el lenguaje puede ser aplicado a cualquier documento XML, incluyendo XHTML, SVG, XUL, RSS, etcétera. También permite aplicar estilos no visuales, como las hojas de estilo auditivas.

Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web, y GUIs para muchas aplicaciones móviles (como Firefox OS).

CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o layouts, los colores y las fuentes. Esta separación busca mejorar la accesibilidad del documento, proveer más flexibilidad y control en la especificación de características presentacionales, permitir que varios documentos HTML compartan un mismo estilo usando una sola hoja de estilos separada en un archivo .css, y reducir la complejidad y la repetición de código en la estructura del documento.

La separación del formato y el contenido hace posible presentar el mismo documento marcado en diferentes estilos para diferentes métodos de renderizado, como en pantalla, en impresión,

en voz (mediante un navegador de voz o un lector de pantalla, y dispositivos táctiles basados en el sistema Braille. También se puede mostrar una página web de manera diferente dependiendo del tamaño de la pantalla o tipo de dispositivo. Los lectores pueden especificar una hoja de estilos diferente, como una hoja de estilos CSS guardado en su computadora, para sobreescribir la hoja de estilos del diseñador.

La especificación CSS describe un esquema prioritario para determinar que reglas de estilo se aplican si más de una regla coincide para un elemento en particular. Estas reglas, aplicadas con un sistema llamado en cascadas, las prioridades son calculadas y asignadas a las reglas, así que los resultados son predecibles.

La especificación CSS es mantenida por el World Wide Web Consortium (W3C). El MIME type text/css está registrado para su uso por CSS descrito en el RFC 2318 (March 1998). El W3C proporciona una herramienta de validación de CSS gratuita para los documentos CSS.

D.5. Entorno de programación

PHP Designer es un completo entorno de desarrollo y programación especialmente diseñado para los gurús de PHP, aunque también permite trabajar con comodidad en otros lenguajes de programación como HTML, XHTML, CSS y SQL.

Ofrece toda una serie de asistentes y diálogos integrados que facilitan en todo momento tu tarea, además de acceso directo a librerías de código o scripts de uso habitual, utilidades diversas y toda suerte de herramientas, todo ello en una interfaz de diseño sencillo y elegante que puedes personalizar con nada menos que dieciocho temas distintos.

Cuenta con cliente de FTP y navegador de ficheros integrado, utilidades de corrección y autocompletado, búsqueda integrada en Google y soporte para proyectos, además de usar un práctico esquema de color para la sintaxis del código fuente que facilita enormemente la programación. PHP Designer Soporta: PHP, HTML, XHTML, CSS, Java, Perl, JavaScript, VB, C# y SQL

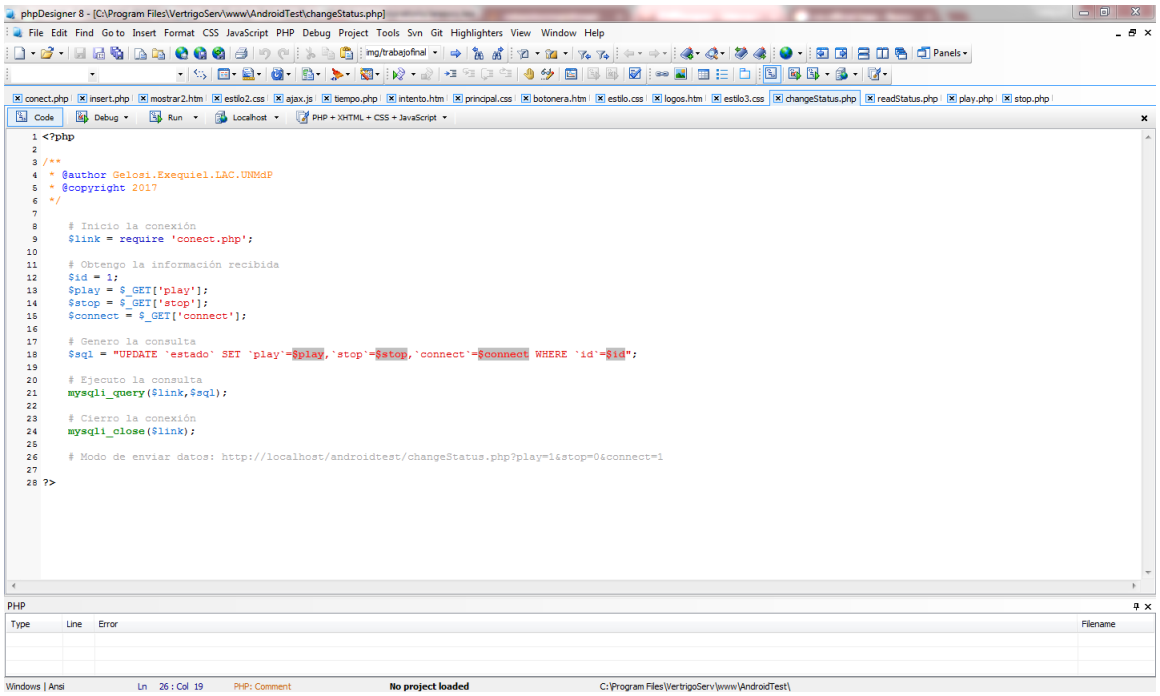

Figura D.4: Entorno PHPDesigner 8

D.6. Servidor WEB

Un servidor es una aplicación en ejecución (software) capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia. Los servidores se pueden ejecutar en cualquier tipo de computadora, incluso en computadoras dedicadas a las cuales se les conoce individualmente como «el servidor». En la mayoría de los casos una misma computadora puede proveer múltiples servicios y tener varios servidores en funcionamiento. La ventaja de montar un servidor en computadoras dedicadas es la seguridad. Por esta razón la mayoría de los servidores son procesos diseñados de forma que puedan funcionar en computadoras de propósito específico.

Los servidores operan a través de una arquitectura cliente-servidor. Los servidores son programas de computadora en ejecución que atienden las peticiones de otros programas, los clientes. Por tanto, el servidor realiza otras tareas para beneficio de los clientes. Ofrece a los clientes la posibilidad de compartir datos, información y recursos de hardware y software. Los clientes usualmente se conectan al servidor a través de la red pero también pueden acceder a él a través de la computadora donde está funcionando. En el contexto de redes Internet Protocol (IP), un servidor es un programa que opera como oyente de un socket.

Comúnmente los servidores proveen servicios esenciales dentro de una red, ya sea para usua-

rios privados dentro de una organización o compañía, o para usuarios públicos a través de Internet. Los tipos de servidores más comunes son servidor de base de datos, servidor de archivos, servidor de correo, servidor de impresión, servidor web, servidor de juego, y servidor de aplicaciones.

Un gran número de sistemas usa el modelo de red cliente-servidor, entre ellos los sitios web y los servicios de correo. Un modelo alternativo, el modelo red peer-to-peer permite a todas las computadoras conectadas actuar como clientes o servidores acorde a las necesidades.

En particular, el servidor Vertrigo (utilizado en la realización de este trabajo) es un paquete de software como WAMP incluye un servidor Apache , la base de datos MySQL , un intérprete de PHP y herramientas de administración de phpMyAdmin , SQLite , Zend Optimizer y SQLiteManager.

D.7. Base de datos MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

Figura D.5: Logo de MySQL

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL A.B. fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de doble licenciamiento anteriormente mencionado. La base de datos se distribuye en varias versiones, una Community, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones Enterprise, para aquellas empresas que quieran incorporarlo en productos privativos. Las versiones Enterprise incluyen productos o servicios adicionales tales como herramientas de monitorización y soporte oficial.

En 2009 se creó un fork denominado MariaDB por algunos desarrolladores (incluido algunos desarrolladores originales de MySQL) descontentos con el modelo de desarrollo y el hecho de que una misma empresa controle a la vez los productos MySQL y Oracle Database.
Está desarrollado en su mayor parte en ANSI C y C++. Tradicionalmente se considera uno de los cuatro componentes de la pila de desarrollo LAMP y WAMP.
MySQL es usado por muchos sitios web grandes y populares, como Wikipedia, Google (aunque no para búsquedas), Facebook, Twitter, Flickr, y YouTube.

Apéndice E

Aspectos técnicos

E.1. Sensor de proximidad

Un sensor de proximidad es un transductor que detecta objetos o señales que se encuentran cerca del elemento sensor.

Los sensores de ultrasonidos son detectores de proximidad que trabajan libres de roces mecánicos y que detectan objetos a distancias de hasta 10m. El sensor emite impulsos ultrasónicos. Estos reflejan en un objeto, el sensor recibe el eco producido y lo convierte en señales eléctricas, las cuales son elaboradas en el aparato de valoración. Estos sensores trabajan solamente en el aire, y pueden detectar objetos con diferentes formas, superficies y de diferentes materiales. Los materiales pueden ser sólidos, líquidos o polvorientos, sin embargo han de ser deflectores de sonido. Los sensores trabajan según el tiempo de transcurso del eco, es decir, se valora la distancia temporal entre el impulso de emisión y el impulso del eco.

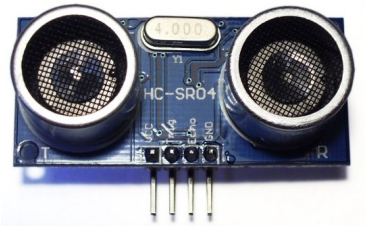

Figura E.1: Sensor de proximidad

Este sensor al no necesitar el contacto físico con el objeto ofrece la posibilidad de detectar objetos frágiles, como pintura fresca, además detecta cualquier material, independientemente del color, al mismo alcance, sin ajuste ni factor de corrección. Los sensores ultrasónicos tienen una función de aprendizaje para definir el campo de detección, con un alcance mínimo y máximo de precisión de 6 mm. El problema que presentan estos dispositivos son las zonas ciegas y el problema de las falsas alarmas. La zona ciega es la zona comprendida entre el lado sensible del detector y el alcance mínimo en el que ningún objeto puede detectarse de

forma fiable.

E.2. PWM

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

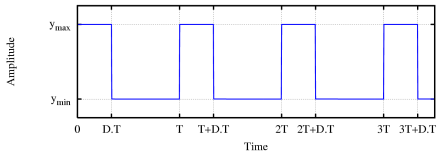

Figura E.2: Señal de onda cuadrada mostrando el ciclo de trabajo D

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$$D = \frac{\tau}{T} \tag{E.1}$$

Donde D es el ciclo de trabajo, τ es el tiempo en que la función es positiva (ancho del pulso) y T es el período de la función.

La construcción típica de un circuito PWM se lleva a cabo mediante un comparador con dos entradas y una salida. Una de las entradas se conecta a un oscilador de onda dientes de sierra, mientras que la otra queda disponible para la señal moduladora. En la salida la frecuencia es generalmente igual a la de la señal dientes de sierra y el ciclo de trabajo está en función de la portadora.

La principal desventaja que presentan los circuitos PWM es la posibilidad de que haya interferencias generadas por radiofrecuencia. Estas pueden minimizarse ubicando el controlador cerca de la carga y realizando un filtrado de la fuente de alimentación.

En la actualidad existen muchos circuitos integrados en los que se implementa la modulación PWM, además de otros muy particulares para lograr circuitos funcionales que puedan controlar fuentes conmutadas, controles de motores, controles de elementos termoeléctricos, choppers para sensores en ambientes ruidosos y algunas otras aplicaciones. Se distinguen por fabricar este tipo de integrados compañías como Texas Instruments, National Semiconductor,

Maxim, y algunas otras más.

E.2.1. Aplicaciones en motores

La modulación por ancho de pulsos es una técnica utilizada para regular la velocidad de giro de los motores eléctricos de inducción o asíncronos. Mantiene el par motor constante y no supone un desaprovechamiento de la energía eléctrica. Se utiliza tanto en corriente continua como en alterna, como su nombre lo indica, al controlar: un momento alto (encendido o alimentado) y un momento bajo (apagado o desconectado), controlado normalmente por relès (baja frecuencia) o MOSFET o tiristores (alta frecuencia).
Otros sistemas para regular la velocidad modifican la tensión eléctrica, con lo que disminuye el par motor; o interponen una resistencia eléctrica, con lo que se pierde energía en forma de calor en esta resistencia.
Otra forma de regular el giro del motor es variando el tiempo entre pulsos de duración constante, lo que se llama modulación por frecuencia de pulsos.
En los motores de corriente alterna también se puede utilizar la variación de frecuencia.
La modulación por ancho de pulsos también se usa para controlar servomotores, los cuales modifican su posición de acuerdo al ancho del pulso enviado cada un cierto período que depende de cada servo motor. Esta información puede ser enviada utilizando un microprocesador como el Z80, o un microcontrolador (por ejemplo, un PIC 16F877A, 16F1827, 18F4550, etc. de la empresa Microchip), o un microcontrolador de hardware libre como es Arduino.

E.3. BaudRate

Aunque a veces se confunden los baudios con los bits por segundo, son conceptos distintos. En transmisiones digitales ocurre lo siguiente: la información digital, codificada en bits, normalmente no se puede enviar directamente por el medio de transmisión (por ejemplo asociando un nivel eléctrico al 1 y al 0, típicamente 5 V y 0 V, respectivamente) debido a que los medios de transmisión suelen estar limitados en banda ?esto es, que solo dejan pasar las componentes frecuenciales de una señal que se encuentren en un rango determinado de frecuencias (por ejemplo, entre 1 kHz y 4 kHz).
Ocurre que al codificar los bits como un nivel eléctrico, la señal sufre transiciones muy rápidas, lo que genera frecuencias muy altas. Por ejemplo, si se quiere transmitir un 1 y después un 0, hay que pasar de 5 V a 0 V inmediatamente.

Una manera de solucionar esto es codificando los bits de otra manera; por ejemplo, asociando cada bit a una señal que el medio sí admita, como por ejemplo, senos y cosenos; si el medio limita a señales que se encuentren en el rango de 1 kHz y 4 kHz, podemos transmitir una señal sinusoidal de 2 kHz para expresar un 1 y otra de 3 kHz para expresar un 0, lo que sería una manera primitiva de modulación FSK (frequency shift keying o "codificación por desplazamiento en frecuencia"). Estas señales tienen un tiempo de duración comúnmente llamado tiempo de símbolo T, de modo que cada T segundos se transmite una de las dos señales. Como cada señal codifica 1 bit, cada T segundos se transmite 1 bit, luego la tasa de bits es 1/T bps (bits por segundo), que en este caso coincide con los baudios.
Puede ser interesante codificar de una manera más complicada, usando por ejemplo cuatro senos; así un seno a 1 kHz significa 00, uno a 2 kHz corresponde a 01, los 3 kHz a 10 y los 4 kHz a 11. De este modo, la tasa de baudios sigue siendo 1/T baudios ya que se transmite una señal cada T segundos. No obstante, la tasa de bits es distinta porque en cada señal van 2 bits, esto es 2/T bps (o 2 bits cada T segundos). De manera general, una señal puede codificar 2^n bits (1, 2, 4, 8, 16, 32, 64, 128...) y se define la tasa de símbolo Rs como el número de símbolos (señales) que se transmiten en un tiempo de símbolo T, normalmente:

$$Rs = \frac{1}{T}[baudios] \tag{E.2}$$

Asimismo, la tasa de bits (bitrate) es el número de bits que se transmiten en un tiempo T y se calcula como:

$$Rb = \frac{2^n}{T}[bps] \tag{E.3}$$

E.4. Compresión en Base64

Base 64 es un sistema de numeración posicional que usa 64 como base. Es la mayor potencia de dos que puede ser representada usando únicamente los caracteres imprimibles de ASCII. Esto ha propiciado su uso para codificación de correos electrónicos, PGP y otras aplicaciones. Todas las variantes famosas que se conocen con el nombre de Base64 usan el rango de caracteres A-Z, a-z y 0-9 en este orden para los primeros 62 dígitos, pero los símbolos escogidos para los últimos dos dígitos varían considerablemente de unas a otras. Otros métodos de codificación como UUEncode y las últimas versiones de binhex usan un conjunto diferente de 64 caracteres para representar 6 dígitos binarios, pero estos nunca son llamados Base64.

E.5. Funciones

En esta sección, se mostrarán las funciones que se utilizaron para el diseño que no se muestran en la página principal. Se realizó de este modo, ya que el código es extenso, y dificulta la lectura y comprensión del trabajo.

E.5.1. Código ajax.js

Este script se encarga de repetir periódicamente la ejecución y carga de imágenes desde el servidor.

```
1    var milis = 100; // el tiempo en que se refresca
2  var divid = "contenido"; // el div que quieres actualizar!
3  var url = "tiempo.php"; // el archivo que ira en el div
4    var miVar =false;
5    var cant = 0;
6
7  function refreshdiv(){
8    var xmlHttp;
9    try{
10     xmlHttp=new XMLHttpRequest(); // Firefox, Opera 8.0+, Safari
11   }
12   catch (e){
13     try{
14       xmlHttp=new ActiveXObject("Msxml2.XMLHTTP"); // Internet Explorer
15     }
16     catch (e){
17       try{
18         xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
19       }
20       catch (e){
21         alert("Tu explorador no soporta AJAX.");
22         returnfalse;
23       }
24     }
25   }
26   var timestamp = parseInt(new Date().getTime().toString().substring(0, 10))
    ;
27   var nocacheurl = url+"?mivar="+cant+"&dir="+"trabajofinal.jpg";
```

```
28
29      xmlHttp.onreadystatechange=function(){
30        if(xmlHttp.readyState== 4 && xmlHttp.readyState != null){
31          document.getElementById(divid).innerHTML=xmlHttp.responseText;
32          setTimeout('refreshdiv()',milis);
33        }
34      }
35      xmlHttp.open("GET",nocacheurl,true);
36      xmlHttp.send(null);
37    }
38
39    window.onload = function(){
40      refreshdiv();
41    }
```

Se encuentra dividido en dos funciones, la primera se ejecuta cuando se inicia la página (*window.onload*), y hace el llamado a la otra función. Esta otra función, *refreshdiv*, se encarga de repetir periódicamente el *php* llamado *tiempo.php*. Antes de ejecutarlo, le agrega una serie de parámetros al link, entre dichos parámetros se encuentra la dirección donde se descargó la imagen.

E.5.2. Código tiempo.php

Este php, imprime la fecha y hora en pantalla y luego imprime la imagen. A esta imagen se le da un ancho proporcional al tamaño de la pantalla.

```
1  <?php
2      $miVarNew = $_GET['mivar'];
3      // Formateamos la salida de la variable.
4      $str = "It is %a on %b %d, %Y, %X - Time zone: %Z";
5      // Imprimimos el resultado
6      $local = $_GET['dir'];
7      $imagen ='<img  src="'.$local.'" width="68%" />';
8      echo (gmstrftime($str,time()));
9      echo '<br>';
10     echo $imagen;
11     echo '<br>';
```

```
?>
```

E.5.3. Código de funcionamiento de los botones

```
<!DOCTYPE HTML>
<html>
<head>
  <meta http-equiv="content-type" content="text/html" />
  <meta name="author" content="www.intercambiosvirtuales.org" />
    <meta http-equiv="cache-control" content="no-cache" />
    <meta http-equiv="Pragma" content="no-cache"/>
    <link href="estilo.css" rel="stylesheet" type="text/css"/>

</head>

<body id="botonesBody">



    <div id ="botones" >
                <script>
                    if (window.innerHeight){
                        //navegadores basados en mozilla
                        espacio_iframe = window.innerHeight - 10
                    }else{
                        if (document.body.clientHeight){
                            //Navegadores basados en IExplorer, es que no
    tengo innerheight
                            espacio_iframe = document.body.clientHeight - 10
                        }else{
                            //otros navegadores
                            espacio_iframe = 478
                        }
                    }
                    document.write ('<h2>Comandos de movimiento</h2>')
                </script>
                <script>
```

```
33                    if (window.innerHeight){
34                        //navegadores basados en mozilla
35                        espacio_iframe = window.innerHeight - 10
36                    }else{
37                        if (document.body.clientHeight){
38                            //Navegadores basados en IExplorer, es que no
    tengo innerheight
39                            espacio_iframe = document.body.clientHeight - 10
40                        }else{
41                            //otros navegadores
42                            espacio_iframe = 478
43                        }
44                    }
45                    document.write ('<form id="func1" name="frmimage" id="
    frmimage" method="post" style="float: left"')
46                    document.write ('enctype="multipart/form-data" action="
    fun1.php" title="Iniciar el movimiento del robot">')
47                    document.write ('<a href="#" onClick="myFunction();"><img
    src="play.png" height="'+ 0.45*espacio_iframe +'" width="'+ 0.45*
    espacio_iframe +'" />')
48                    document.write (' </form>')
49                </script>
50                <script>
51                    if (window.innerHeight){
52                        //navegadores basados en mozilla
53                        espacio_iframe = window.innerHeight - 10
54                    }else{
55                        if (document.body.clientHeight){
56                            //Navegadores basados en IExplorer, es que no
    tengo innerheight
57                            espacio_iframe = document.body.clientHeight - 10
58                        }else{
59                            //otros navegadores
60                            espacio_iframe = 478
61                        }
62                    }
63                    document.write (' <form id="func2" name="frmimage" id="
    frmimage" method="post" style="float: left"')
```

```
64                    document.write ('enctype="multipart/form-data" action="
    fun2.php" title="Detener el movimiento del robot">')
65                    document.write ('<a href="#" onClick="myFunction1();"><img
     src="pause.png" height="'+ 0.45*espacio_iframe +'" width="'+ 0.45*
    espacio_iframe +'"/>')
66                    document.write (' </form>')
67                </script>
68                 <script>
69                    if (window.innerHeight){
70                       //navegadores basados en mozilla
71                       espacio_iframe = window.innerHeight - 10
72                    }else{
73                       if (document.body.clientHeight){
74                            //Navegadores basados en IExplorer, es que no
    tengo innerheight
75                            espacio_iframe = document.body.clientHeight - 10
76                       }else{
77                            //otros navegadores
78                            espacio_iframe = 478
79                       }
80                    }
81                    document.write ('<form id="func3" name="frmimage" id="
    frmimage" method="post" style="float: left"')
82                    document.write ('enctype="multipart/form-data" action="
    fun3.php" title="Iniciar la conexión entre dispositivos">')
83                    document.write ('<a href="#" onClick="myFunction2();"><img
     src="connect.png" height="'+ 0.45*espacio_iframe +'" width="'+ 0.45*
    espacio_iframe +'"/>')
84                    document.write (' </form>')
85                </script>
86            <div style="clear: both"></div>
87     </div>
88
89
90 <script>
91     function myFunction() {
92             // The XMLHttpRequest object
93
94     var xmlHttp;
```

```
95      try{
96        xmlHttp=new XMLHttpRequest(); // Firefox, Opera 8.0+, Safari
97      }
98      catch (e){
99        try{
100         xmlHttp=new ActiveXObject("Msxml2.XMLHTTP"); // Internet Explorer
101       }
102       catch (e){
103         try{
104           xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
105         }
106         catch (e){
107           alert("Tu explorador no soporta AJAX.");
108           returnfalse;
109         }
110       }
111     }
112       xmlHttp.open("GET","play.php",true);
113     xmlHttp.send(null);
114     }
115
116     function myFunction1() {
117             // The XMLHttpRequest object
118
119     var xmlHttp;
120     try{
121       xmlHttp=new XMLHttpRequest(); // Firefox, Opera 8.0+, Safari
122     }
123     catch (e){
124       try{
125         xmlHttp=new ActiveXObject("Msxml2.XMLHTTP"); // Internet Explorer
126       }
127       catch (e){
128         try{
129           xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
130         }
131         catch (e){
132           alert("Tu explorador no soporta AJAX.");
133           returnfalse;
```

```
134         }
135       }
136     }
137       xmlHttp.open("GET","stop.php",true);
138     xmlHttp.send(null);
139     }
140
141         function myFunction2() {
142             // The XMLHttpRequest object
143
144     var xmlHttp;
145     try{
146       xmlHttp=new XMLHttpRequest(); // Firefox, Opera 8.0+, Safari
147     }
148     catch (e){
149       try{
150         xmlHttp=new ActiveXObject("Msxml2.XMLHTTP"); // Internet Explorer
151       }
152       catch (e){
153         try{
154           xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
155         }
156         catch (e){
157           alert("Tu explorador no soporta AJAX.");
158           returnfalse;
159         }
160       }
161     }
162       xmlHttp.open("GET","connect.php",true);
163     xmlHttp.send(null);
164     }
165 </script>
166 </body>
167 </html>
```

Bibliografía

[1] Autor: GARCIA E. M.

Titulo: *Visión artificial*,

Ed. UOC. 2012.

[2] Autores: FU K.S., GONZALEZ R.C., LEE C.S.

Titulo: *Robótica: Control, Visión, Detección e Inteligencia*,

Graw-Hill- Madrid. 1988.

[3] Autor: CRAIG, John J.

Titulo: *Robótica*,

Pearson Education 3era Ed. 2006.

[4] Autores: ASTUA C., BARBER R., CRESPO J. y JARDON J.

Titulo: *"Object Detection Techniques Applied on Mobile Robot Semantic Navigation"*.,

SENSORS. www.mdpi.com/journal/sensors.2014.

[5] Autores: SOLARI F., CHESSA M., SABATINI S.

Titulo: *Machine Vision - Applications and Systems.*,

Ed. INTECH 2012..

[6] Autor: BUCHLI J.

Titulo: *Mobile robotics, moving intelligence.*,

Ed. VERLAG 2006.

[7] Autor: KYPRAIOS I.

Titulo: *Advances in Object Recognition Systems.*,

Ed. INTECH 2012.

[8] Autor: OBINATA G., DUTTA A.

Titulo: *Vision systems applications.*,

Ed. I-TECH 2007.

[9] Autor: RIVERA R. R.

Titulo: *Instrumentación virtual aplicada al estudio de sistemas complejos.*,

www3.fi.mdp.edu.ar/electronica/tesis/rrivera.pdf. Tesis Doctoral, 2011.

[10] Autores: GEMIN W. A., RIVERA R. R., FERNÁNDEZ J. G., HIDALGO R. M., REVUELTA M. A., URIZ A. J.

Titulo: *"Control de Sillas de Ruedas Eléctricas Mediante Joystick y Celular".*,

INNOVAR 2013.

[11] Autores: GEMIN W. A., RIVERA R. R., FERNÁNDEZ J. G., HIDALGO R. M., REVUELTA M. A., URIZ A.J.

Titulo: *"Desarrollo de un sistema de control y potencia de bajo costo para sillas de ruedas motorizadas".*,

JAIIO 42 - AST 2013.

[12] Autores: BURNETTE Ed

Titulo: *"Introducing Google's Mobile Development Plataform ".*,

Third Edition.

[13] Autores: MEIER R.

Titulo: *"Android Application Development".*,

Worx.

[14] Fuente: OpenCV Documentation.

Link: *http://docs.opencv.org/2.4/modules/imgproc/doc.html.*

[15] Fuente: comScore.

Link: *http://www.comscore.com/Insights/Presentations-and-Whitepapers/2016/2016-Global-Digital-Future-in-Focus.*

[16] Autores: Leonardo Enrique Solaque Guzmán, Manuel Alejandro Molina Villa, Edgar Leonardo Rodríguez Vásquez.

Titulo: *"Seguimiento de trayectorias con un robot móvil de configuración diferencial".*

Universidad Militar Nueva Granada.

[17] Autores: Pérez Arreguín Jorge Israel, Tovar Arriaga Saúl, Ubaldo Giovanni Villaseñor Carrillo, Gorrostieta Hurtado Efrén, Pedraza Ortega Jesús Carlos, Vargas Soto José Emilio, Ramos Arreguín Juan Manuel y Sotomayor Olmedo Artemio.

Titulo: *"Robot Móvil de Tracción Diferencial con Plataforma de Control Modular para Investigación y Desarrollo Ágil de Proyectos.".*

Universidad Autónoma de Querétaro, Facultad de Informática.

[18] Fuente: Php.

Link: *http://php.net.*

[19] Fuente: w3schools.

Link: *http://www.w3schools.com/html/.*

[20] Fuente: Mozilla Developer Network.

Link: *https://developer.mozilla.org/es/docs/Web/HTML.*

[21] Autor: Robert Schifreen.

Titulo: *"How to create Web sites and applications with HTML, CSS, Javascript, PHP and MySQL..".*

Link: *http://www.itp.uzh.ch/ suzanne/ebooks/The %20Web %20Book-A4-HM.pdf*