UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS

ELECTRÓNICA E INDUSTRIAL

Carrera de Ingeniería en Electrónica y Comunicaciones

TEMA:

"SISTEMA ELECTRÓNICO DE AYUDA PARA EL ESTACIONAMIENTO DE VEHÍCULOS LIVIANOS"

Proyecto de Trabajo de Graduación Modalidad: TEMI. Trabajo Estructurado de Manera Independiente, presentado previo a la obtención del título de Ingeniería en Electrónica y Comunicaciones.

SUBLÍNEA DE INVESTIGACIÓN: Sistemas Embebidos

AUTOR: Renato Andrés Lasluisa Vargas

PROFESOR REVISOR: Ing. Mg. Santiago Manzano

Ambato – Ecuador

Abril 2015

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de investigación sobre el tema: “SISTEMA ELECTRÓNICO DE AYUDA PARA EL ESTACIONAMIENTO DE VEHÍCULOS LIVIANOS” del señor Lasluisa Vargas Renato Andrés, estudiante de la Carrera de Ingeniería Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe de investigación reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el Art. 16 del Capítulo II, del Reglamento de Graduación para Obtener el Título Terminal de Tercer Nivel de la Universidad Técnica de Ambato.

Ambato, Abril 2015

EL TUTOR

……

Ing. Mg. Santiago Manzano

AUTORÍA

El presente trabajo de investigación titulado “SISTEMA ELECTRÓNICO DE AYUDA PARA EL ESTACIONAMIENTO DE VEHÍCULOS LIVIANOS” es absolutamente original, auténtico y personal en tal virtud, el contenido, efectos legales y académicas que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, Abril 2015

……

Renato Andrés Lasluisa Vargas

CC.: 1804360392

APROBACIÓN DE LA COMISIÓN CALIFICADORA

La Comisión Calificadora del presente trabajo conformada por los señores docentes aprobó el Informe Final del trabajo de graduación titulado “SISTEMA ELECTRÓNICO DE AYUDA PARA EL ESTACIONAMIENTO DE VEHÍCULOS LIVIANOS” presentado por el señor Lasluisa Vargas Renato Andrés de acuerdo al Art. 17 del Reglamento de Graduación para obtener el título Terminal de tercer nivel de la Universidad Técnica de Ambato.

…….

Ing. Vicente Morales Lozada, Mg.

PRESIDENTE DEL TRIBUNAL

……..	……
Ing. Marco Jurado, Mg.	Ing. Geovanni Brito, Mg.
DOCENTE CALIFICADOR	DOCENTE CALIFICADOR

DEDICATORIA

Dedico este trabajo a Dios por darme sus bendiciones y sabiduría en cada meta que me he propuesto.

A mis padres, Rodrigo Lasluisa y Carmen Vargas quienes me dieron la vida, las ganas y la fuerza para seguir adelante.

A mi hermana Gabriela Lasluisa quien siempre estuvo ahí para darme aliento y apoyo en toda mi vida universitaria.

A mi familia quienes me brindaron sus consejos para ser una persona de bien.

Y finalmente a mi pareja, Anita Amaya quien me acompañado en todo momento en el transcurso de mi carrera.

Renato Andrés Lasluisa Vargas

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Ambato es una de las ciudades con mayor número de vehículos circulando por las calles dando lugar a que los conductores se vean en la necesidad de estacionarse en espacios reducidos, logrando así, que el vehículo al momento de parquearse tienda a provocar choques, ralladuras, y accidentes con otros carros. Con el avance de la tecnología se han desarrollado múltiples sistemas de parqueo para que los conductores puedan estacionarse con mayor facilidad. Además los fabricantes de automóviles ya instalan sus sistemas propios de parqueo en sus vehículos de última generación.

Este proyecto presenta un nuevo sistema de parqueo alternativo basado en software y hardware libre. El cual está constituido de sensores ultrasónicos que miden la distancia de un obstáculo que se encuentre en la parte delantera y trasera del vehículo, una cámara que visualiza el objeto detectado por los sensores ubicado en la parte trasera del automóvil, un zumbador que suena a diferente frecuencia dependiendo de la cercanía del objeto al carro, una pantalla que presenta la distancia medida por los sensores y la visualización de la cámara y por último de una placa que es el cerebro del sistema de estacionamiento. También cuenta con una aplicación realizada en Android con el entorno de desarrollo Eclipse.

Cualquier persona con conocimientos de programación y electrónica puede modificar y aumentar su funcionamiento dependiendo de sus necesidades. Este sistema de estacionamiento funcionará incluso en vehículos grandes realizando pocas modificación en su diseño.

Palabras clave: sistemas embebidos, sensor, software y hardware libre.

ABSTRACT

Ambato is one of the cities with the highest number of vehicles on the streets leading to that drivers see the need to park in tight spaces, making thus the vehicle to park when tending to cause shock, scratches, and accidents with other vehicles. With the advancement of technology have developed multiple systems parking for drivers to park more easily. In addition automakers and install their own systems of parking in their newest cars.

This project presents a new alternative parking system based on free software and hardware. Which consists of ultrasonic sensors which measure the distance of an obstacle which is in front and rear of the vehicle, a camera that displays the object detected by the sensors located on the rear of the car, a buzzer sounding different frequency depending on the proximity of the object to car, a screen displaying the distance measured by the sensors and camera display and finally a plate which is the brain of the parking system. It also has an application on Android with Eclipse development environment.

Anyone with knowledge of programming and electronics can modify and enhance its operation depending on your needs. This parking system will work even on large vehicles making few changes to its design.

Keywords: embedded systems, sensor, software and free hardware.

GLOSARIO DE TÉRMINOS Y ACRÓNIMOS

AC/DC	*Alternate Current/Direct Current* - Corriente Alterna/Corriente Directa
ADT	*Android Development Tools* - Herramientas de desarrollo de Android
API	*Application Programming Interface* - Interfaz de Programación de Aplicaciones
ARM	*Advanced RISC Machine* - Máquina avanzada RISC
CPU	*Central Processing Unit* - Unidad de Procesamiento Central
CSI	*Computer System Interface* - Interfaz Estándar de Equipos
DC	*Direct Current* - Corriente directa
DDR3	*Double Data Rate type three* - Doble velocidad de datos tipo tres
DSC	*Digital Signal Controller* - Controlador Digital de Señales
DSP	*Digital Signal Processing* - Procesamiento Digital de Señales
DVD	*Digital Versatile Disc2* - Disco Versátil Digital
E/S	Entrada/Salida
ECJ	*Eclipse Compiler for Java* - Compilador de Eclipse para Java
FAT	*File Allocation Table* - Tabla de Asignación de Archivos
FSF	*Free Software Foundation* - Fundación para el software libre
GB	*Gigabyte*
GND	*Ground* - Tierra

GPIO	*General Purpose Input/Output* - Entrada/Salida de Propósito General
GPS	*Global Positioning System* - Sistema de Posicionamiento Global
GPU	*Graphics Processing Unit* - Unidad de Procesamiento Gráfico
HD	*High Definition* - Alta definición
HDMI	*High Definition Multimedia Interface* - Interfaz Multimedia de alta definición
I/O	Entrada/Salida (digital)
I2C	*Inter- Integrated Circuits* - Circuitos Integrados Internacionales
IDE	*Integrated Development Environment* - Entorno integrado de desarrollo
JDK	*Java Development Kit* - Kit de desarrollo de Java
JDT	*Java Development Tools* - Herramientas de desarrollo de Java
JRE	*Java Runtime Environment* - Entorno de ejecución de Java
JVM	*Java Virtual Machine* - Máquina virtual de Java
LCD	*Liquid Crystal Display* - Pantalla de Cristal liquido
LED	*Light Emitting Diode* - Diodo Emisor de Luz
LTS	*Long Term Support* - Soporte a Largo Plazo
LVDS	*Low Voltage Differential Signaling* - Señal Diferencial de Bajo Voltaje
OS	*Operating System* - Sistema operativo
PC	*Personal Computer* - Computadora personal

PWM	*Pulse Width Modulation* - Modulación por Ancho de Pulso
QSXGA	*Quad Super Extended Graphics Array* - Cuatro Súper Matrices Gráficas Extendidas
RAM	*Random Access Memory* - Memoria de Acceso Aleatorio
RGB	*Red, Green, Blue* – Rojo, Verde, Azul
RISC	*Reduced Instruction Set Computer* - Computador con Conjunto de Instrucciones Reducidas
RJ45	*Registred Jack 45* - Conector 45 registrado
RS-232	*Recommended Standard 232* - Norma recomendada 232
SATA	*Serial Advanced Technology Attachment* – Accesorio Tecnológico Avanzado Serial
SD	*Secure Digital* - Seguridad Digital
SDK	*Software Development Kit* - Kit de desarrollo de software
SNR	*Signal Noise Ratio* - Relación Señal Ruido
TFT	*Thin Film Transistor* - Transistores de Película Fina
USB	*Universal Serial Bus* - Bus Serial Universal
VCC	*DC Voltage* - Voltaje de Corriente Continua
Vdc	*Voltage of Continuos Current* - Voltaje de Corriente Continua
VGA	*Video Graphics Array* - Adaptador Gráfico de Video
Wifi	*Wireless Fidelity* - Fidelidad Inalámbrica
XML	*eXtensible Markup Language* - Lenguaje de Marcas Extensible

INTRODUCCIÓN

Esta investigación se realiza debido a que los sistemas de estacionamiento actualmente disponibles en el mercado son muy costosos, no se puede cambiar su funcionalidad ni aumentar accesorios adicionales. Por lo que se ha diseñado un sistema electrónico de ayuda para el estacionamiento de vehículos livianos con el objetivo de tener un sistema de fabricación nacional económico accesible para cualquier persona. Permitiendo modificar su funcionamiento fácilmente en la parte del hardware y software.

En el Capítulo I se describe el problema que tienen los conductores al momento de estacionarse y los inconvenientes de los sistemas de estacionamiento disponibles en el mercado, la delimitación del proyecto, la justificación y los objetivos que se alcanzó al realizar el sistema de estacionamiento.

En el Capítulo II se estudia los sistemas de estacionamiento proporcionados por los fabricantes en cada uno de sus automóviles. El sistema operativo y entorno de desarrollo utilizado para realizar la aplicación. Asimismo se describe los conceptos básicos sobre los elementos que se ha utilizado en el proyecto y se presenta la propuesta de solución.

En el Capítulo III se realiza una investigación de campo, bibliográfica y documental, la recolección de información indispensable para el sistema de estacionamiento, los procesos y análisis de datos, detallando los pasos que se sigue para el desarrollo del proyecto.

En el Capítulo IV se elabora una comparación de cada uno de los dispositivos que se emplea en el sistema de estacionamiento, el desarrollo de la interface de usuario, el esquema del circuito de estacionamiento y la instalación del prototipo con la estimación del costo de fabricación.

En el Capítulo V se presentan las conclusiones y recomendaciones del sistema electrónico de ayuda para el estacionamiento de vehículos livianos.

CAPÍTULO I

EL PROBLEMA

1.1 TEMA

SISTEMA ELECTRÓNICO DE AYUDA PARA EL ESTACIONAMIENTO DE VEHÍCULOS LIVIANOS.

1.2 PLANTEAMIENTO DEL PROBLEMA

Uno de los inventos más significativos del siglo XX ha sido sin duda el automóvil. Los primeros prototipos fueron creados a finales del XIX, pero no fue hasta algunas décadas después cuando los vehículos empezaron a ser útiles para la sociedad.

El comienzo del vehículo se da con los autopropulsados por vapor en el siglo XVIII. En 1885, se crea el primer automóvil por motor de combustión interna con gasolina. Francia y Estados Unidos son quienes empezaron la producción masiva de automóviles *Panhard et Levassor* y *Peugeot* fueron las primeras compañías francesas creadas para fabricarlos. En 1908, se comenzó a realizar automóviles en una cadena de montaje creada por Henry Ford, que le permitió alcanzar una producción enorme de vehículos con un sistema totalmente innovador.

A partir de 1920 aparecen muchas mejoras de seguridad en los vehículos como: los primeros sistemas antirrobo, caja de cambios, ruedas, carrocería. Incluso los diseños han evolucionado para adaptarse a las nuevas necesidades de las personas. En la actualidad gracias a la tecnología los automóviles están comenzando a incorporar nuevas dispositivos como es el GPS, cámaras de video, sistemas de monitoreo, etc [1].

De acuerdo a las estadísticas de la Agencia Nacional de Tránsito del Ecuador los accidentes por rozamiento y obstáculos al momento de parquearse en el año 2012 fueron de 2.328, en el 2013 de 2.626 y en el 2014 de 2.735. Esto se debe a que cada vez existen menos lugares donde estacionarse y los carros se encuentran tan juntos unos a otros que al momento de salir del estacionamiento provocan choques con otros carros u objetos a su alrededor [2].

En las calles se observa que los conductores estacionan sus vehículos en espacios reducidos con mucha dificultad igualmente que al ingreso o salida de los garajes, las personas dañan o rozan sus vehículos con las paredes u obstáculos causando abolladuras y rayones en la pintura. Al momento en que un usuario desea estacionar su vehículo surge el problema de que no todos los objetos a su alrededor son visibles y no es fácil reconocer la distancia real entre el vehículo y otro ubicado a su alrededor, lo que dificulta el parqueo, además existen otros factores adicionales a considerar como: el clima (la lluvia, el polvo, la neblina, etc.), el descuido y otras circunstancias.

El mayor problema de los conductores son las dimensiones de sus vehículos debido a que mientras más grande existe menor visibilidad y no pueden observar lo que tienen atrás del automóvil, asimismo por las diferentes estaturas de las personas (ancianos, niños, adultos, etc.) no son vistas dando lugar a accidentes y/o atropellados.

El estacionarse es algo muy habitual para los conductores, pero cada vez se incrementa el número de automóviles en el país y por lo tanto los espacios para estacionarse se reducen, y el conductor se ve en la necesidad de realizar nuevas maniobras para poder evitar accidentes.

Debido a este inconveniente, los conductores optan por ubicarse en lugares no permitidos, provocando que sean multados por cometer una infracción de tránsito y en algunas ocasiones el automóvil sea retenido.

Hoy en día existen múltiples sistemas que ayudan al conductor a estacionarse, pero el mayor inconveniente es que estos sistemas generalmente se encuentran en automóviles de última generación o son muy caros.

1.3 DELIMITACIÓN

- Área académica: Física y Electrónica
- Línea de investigación: Sistemas Electrónicos
- Sublínea de investigación: Sistemas Embebidos
- Delimitación espacial: La investigación se realizó en la Ciudad de Ambato
- Delimitación temporal: El proyecto se desarrolló del 17 de abril de 2014, al 02 de febrero de 2015 con una duración de 10 meses, a partir de su aprobación por parte del Honorable Consejo Directivo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

1.4 JUSTIFICACIÓN

Se llevó a cabo la investigación de este sistema por el gran incremento de la tecnología hoy en día, la cual poco a poco se va introduciendo en la vida cotidiana y en la actualidad en los automóviles, hasta convertirse en algo indispensable y muy importante para los conductores, también esta investigación ayudará como pauta para futuros proyectos y futuras tecnologías que se irán desarrollando e implementando en los vehículos modernos.

Los beneficiarios son todas las personas que tengan automóviles quienes con este sistema evitarán accidentes involuntarios por la falta de visibilidad de los objetos que se encuentran a su alrededor. El conductor tendrá una ayuda para reconocer a que distancia se encuentra de un vehículo u obstáculo y realizará la maniobra adecuada para poder evitar un choque. Este sistema de ayuda de estacionamiento es fácil de instalar y consta con pocos componentes los cuales

reducen el tamaño del sistema, no necesita de baterías adicionales, por lo cual este sistema se puede mejorar cada vez más y seguir agregando nuevos dispositivos para poder aumentar su funcionalidad.

En la actualidad existen varios sistemas de parqueo los cuales son muy costosos y sus arreglos consiste en cambiar casi todo el sistema, al ser un producto nacional los costos disminuirán drásticamente, si se necesita realizar mantenimiento o arreglar un fallo del sistema gracias a su fabricación permitirá tener un servicio técnico a bajo costo.

1.5 OBJETIVOS

1.5.1 OBJETIVO GENERAL

Diseñar un sistema electrónico de ayuda para el estacionamiento de vehículos livianos.

1.5.2 OBJETIVOS ESPECÍFICOS

- Analizar los sistemas de estacionamientos que se encuentran instalados actualmente en los vehículos.
- Determinar la distancia óptima para la ubicación correcta de los sensores.
- Diseñar una interfaz para la presentación visual de la distancia medida por los sensores.
- Implementar un prototipo del sistema electrónico de ayuda para el estacionamiento de vehículos livianos.

CAPÍTULO II

FUNDAMENTACIÓN TEÓRICA

2.1 ANTECEDENTES INVESTIGATIVOS

A continuación se describe algunas investigaciones realizadas al tema en las cuales se detallan las conclusiones que se obtuvo en cada uno de ellas.

El proyecto desarrollado por el Sr. Bonilla Jorge determina que es muy importante revisar todas las partes del vehículo para evitar accidentes al estacionarse, se necesita de un mantenimiento periódico del vehículo para evitar fallos en su funcionamiento. Además el sistema indica el espacio disponible en la parte trasera y delantera del vehículo. El proyecto no necesariamente puede ser implementado en automóviles modernos sino que se puede implementarlo fácilmente en otros vehículos con muy pocas modificaciones [3].

El proyecto de Suárez Carlos concluye que se debe consultar con las personas que recién están aprendiendo a manejar para conocer cuáles son sus necesidades y así poder realizar un prototipo de acuerdo a los requerimientos planteados por ellos, con esto logró que los instructores mejoren sus técnicas de aprendizaje para el estacionamiento vehicular también con esta investigación se profundizó más en el tema y se escogió que tipo de sensor era el más adecuado para su implementación que en este caso fue un sensor ultrasónico puesto que tiene mayor rango de medición que otros tipos de sensores [4].

La investigación de Mayanza Samuel definió que los sensores Z0 Max sonar son los más adecuados para este tipo de proyecto por sus características de diseño. Estos sensores son muy robustos y se adaptan a diversos tipos de climas con muy poca variación en su señal, tiene un ángulo de visión de 100º lo que hace que tenga una buena área de cobertura por lo que permite utilizar menos sensores en el sistema y así reducir el costo del sistema. Concluye que el sistema es eficiente y con muy poco consumo de energía esto convierte al sistema que sea muy comercial y que se pueda instalar en cualquier tipo de vehículo [5].

El proyecto realizado por González Francisco y Chacón Wilson en su tesis titulada “Diseño e implementación de un sistema de asistencia para estacionamiento en un vehículo Toyota Corolla” concluyen que en la parte mecánica casi no hubo ningún inconveniente, mientras que en el diseño de los circuitos electrónicos fue muy complejo puesto que no se tenía los conocimientos de ese tema, igualmente tuvieron problemas al momento de realizar las pruebas de funcionamiento ya que algunos circuitos trabajaban erróneamente y decidieron cambiar algunas elementos, al momento de instalar el sistema lo realizaron con mucho cuidado sin dañar el interior del vehículo colocándole y sujetándole en un lugar adecuado y no visible para las personas [6].

La tesis realizada por Oscar Matza titulada “Sistema electrónico de control de velocidad de autobuses, para la Cooperativa de Transportes Santa” determina que el sistema propuesto es adaptable a cualquier tipo de autobús. Además que existe un mínimo retardo en el envío de información del GPS pero que no repercute de manera importante en el desempeño sistema, por último los microcontroladores utilizados cumplieron con todos los requerimientos del proyecto dando así un correcto funcionamiento del sistema [7].

2.2 FUNDAMENTACIÓN TEÓRICA

2.2.1 Tipos de sistemas de estacionamiento vehicular

Existen dos tipos de sistemas de estacionamientos que son:

- **Sistemas de estacionamiento ofertados en el mercado**

En la actualidad se oferta múltiples kits de parqueo que utilizan casi los mismos dispositivos para su funcionamiento. Empleando sensores ultrasónicos para medir la distancia, un circuito central que realiza el procesamiento de los datos y un zumbador o display para visualizar la medida obtenida por los sensores. Los primeros kits de parqueo que salieron al mercado utilizaban dos sensores ultrasónicos y un zumbador el cual sonaba más rápido dependiendo de la cercanía de un objeto al vehículo. Con el paso del tiempo se fueron incrementando los sensores llegando a tener hasta 8 sensores, 4 en la parte delantera y 4 en la parte trasera del automóvil. Los últimos sistemas de parqueo constan con una cámara de visión trasera para que el conductor observe el objeto detectado por los sensores.

El sistema de parqueo comúnmente instalado en los automóviles es el que consta de 4 sensores ultrasónicos y un display con leds en donde le indica al conductor la distancia medida por los sensores como se observa en la figura 2.1.

Fig. 2.1 Radar Parking kit auto-instalable
Fuente: http://www.tienda.siliceo.es/radar-de-aparcamiento-sensor-parking-marcha-atras-lcd-led-alarma-p-36.html

➢ Sistemas de estacionamiento instalados de fábrica

Los fabricantes de los automóviles han diseñado cada uno su propio sistema de estacionamiento, entre los más importantes se tiene:

- **Sistema Mercedes Benz**

Mercedes Benz ha creado un nuevo sistema de asistencia para el estacionamiento que se encuentra en el Mercedes Benz Clase CL como se muestra en la figura 2.2. Utiliza una tecnología de radar que ayuda al conductor a ubicar el vehículo en un espacio disponible.

Los sensores se encuentran ubicados en los parachoques delanteros y traseros, con el fin de indicar al conductor si el espacio para estacionar su vehículo está disponible, si es lo suficientemente amplio como para aparcarlo y la mejor manera de cómo debe realizar las maniobras para evitar accidentes [8].

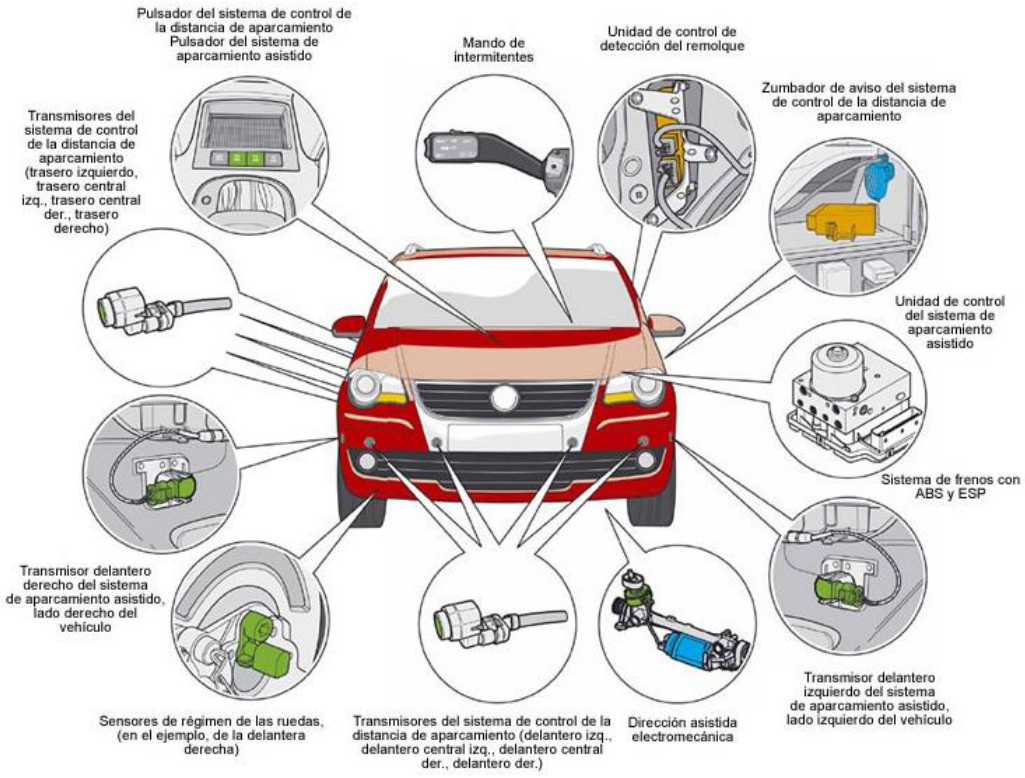


Fig. 2.2 Sistema Mercedes Benz clase "CL"
Fuente: http://www.aficionadosalamecanica.com/sistema-aparcamiento-asistido.htm

- ***Active Park Assist* (Ford)**

La compañía Ford también ha desarrollado un sistema de aparcamiento llamado *Active Park Assist* como se muestra en la figura 2.3.

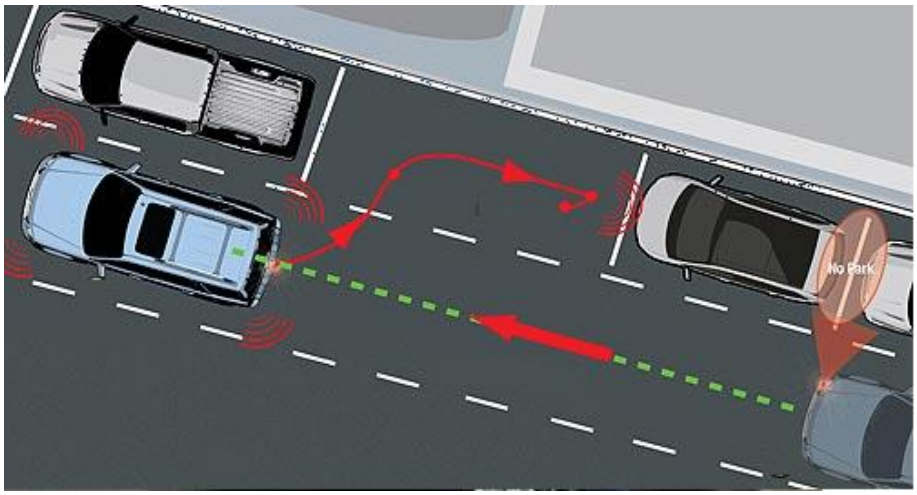


Fig. 2.3 *Active Park Assist de Ford*.
Fuente:
http://www.goauto.com.au/mellor/mellor.nsf/story2/78150FE768F3B335CA25784F0010C61E

Este sistema incorpora sensores de ultrasonido que determinan la posición del vehículo, luego automáticamente gira las ruedas (el conductor mantiene el control de la caja, el acelerador y el freno) y a medida que se acelera el sistema indica visualmente y a través de alertas de audio la proximidad con otros vehículos, objetos y personas [9].

- **Sistema VW (*Park Assist Vision*)**

Con este sistema el vehículo puede deslizarse automáticamente en un espacio disponible perpendicular. Cuenta con unas cámaras y sensores ultrasónicos que realizan el trabajo de aparcarse. Cuando el conductor encuentra un lugar, lo selecciona en la pantalla táctil y lo puede controlar desde adentro o fuera del vehículo. Una vez estacionado, el automóvil apaga el motor, bloquea puertas y sube las ventanas.

Casi todos los modelos VW como se muestra en la figura 2.4 tendrán este sistema incorporado [10].

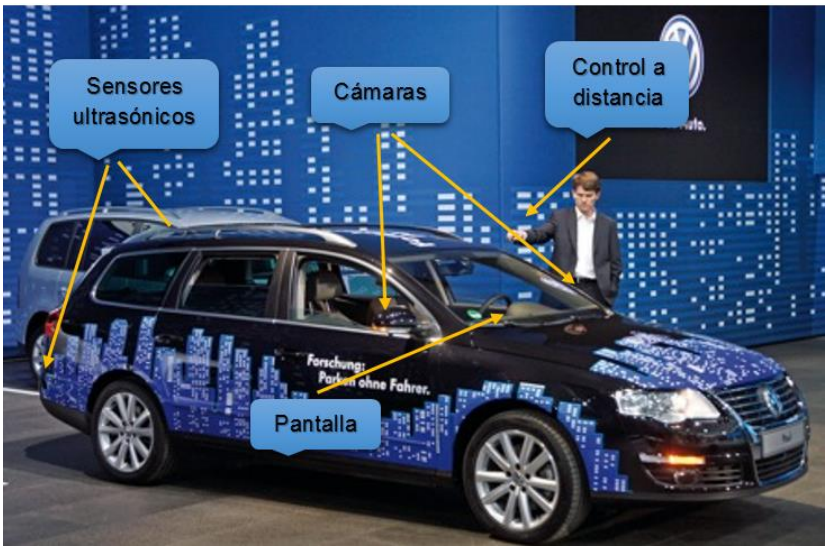


Fig. 2.4 VW *Passat* sistema de asistencia de estacionamiento totalmente automática sin necesidad de conductor.
Fuente: http://es.autoblog.com/2008/04/24/vw-presentaun-nuevo-sistema-de-estacionamiento-que-no-necesita/

- **Sistema Automático BMW**

El departamento de investigación del BMW Group ha desarrollado un sistema completamente automático, capaz de estacionar el coche y guiarlo al salir del garaje como se observa en la figura 2.5.

Fig. 2.5 Sistema BMW serie 7.
Fuente: http://www.dieselstation.com/BMW/7-Series-2013/BMW-7-Series-2013-widescreen-wallpaper-ds17-i4823.html

Este sistema se encuentra en la serie 7 del BMW, consta de una cámara vídeo, colocada a la altura del espejo retrovisor y que capta la zona

delantera del vehículo, tiene unos sensores ultrasónicos que se ocupan de determinar la distancia desde el vehículo hasta los posibles obstáculos. La computadora del sistema se encarga de controlar la puesta en marcha y desconexión del motor, de elegir la marcha, de maniobrar con el volante y de utilizar el sistema. También actúa sobre los frenos, activa los faros y las luces intermitentes de emergencia. Automáticamente al estacionar los espejos retrovisores exteriores se contraen y se vuelven a abrir al salir del estacionamiento [11].

- ***Smart Parking Assist System*** **(Hyundai)**

Hyundai desarrolló un sistema de asistencia de parqueo llamado *Smart Parking Assist System* consta de 8 sensores ultrasónicos, 4 en la parte trasera con una cámara de reversa y 4 en la parte del frente, además tiene dos pantallas una de 7 pulgadas y la otra de 3.5 pulgadas como se observa en la figura 2.6. Al momento de estacionarse se presiona el botón de parqueo el cual buscará un espacio disponible, al encontrarlo se observará en la pantalla de 3.5 pulgadas como se estacionará el vehículo, se coloca la marcha atrás y el sistema estaciona el automóvil, el conductor controla el freno, el acelerador y puede observar la parte trasera del automóvil por medio de la pantalla de 7" [12].

Fig. 2.6 Sistema Hyundai Avante.
Fuente: http://www.autoblog.com.uy/2013/08/pero-si-estas-igual-hyundai-actualiza.html

- ***Automatic Parking Assist*** **(Chevrolet)**

Automatic Parking Assist es así como Chevrolet le llama a su sistema automático de estacionamiento, éste detecta de forma precisa un espacio disponible para estacionarse y ayuda al conductor para que pueda maniobrar y estacionar el automóvil. El sistema posee cuatro sensores ultrasónicos en la parte delantera y cuatro en la parte trasera los cuales determinan la distancia para poder estacionarse igualmente posee una cámara trasera y una pantalla táctil como se muestra en la figura 2.7 [13].

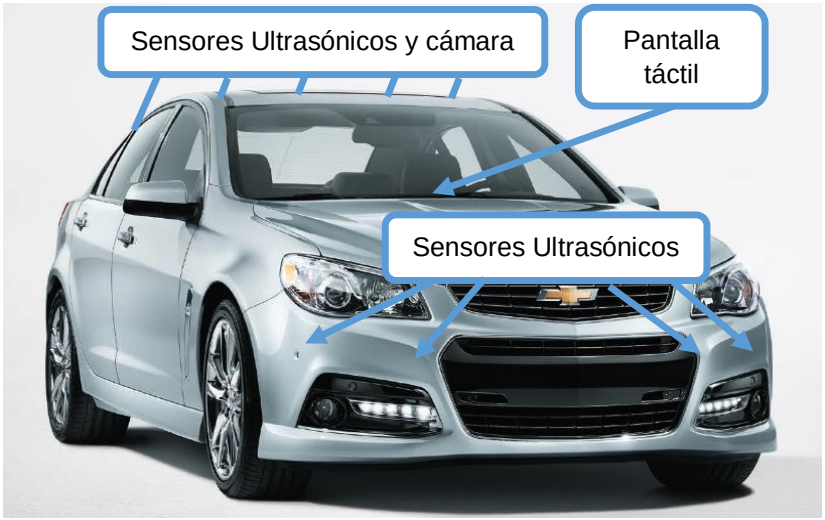


Fig. 2.7 Sedan Deportivo Chevrolet SS 2014
Fuente:
http://es.chevrolet.com/content/dam/Chevrolet/northamerica/usa/nscwebsite/en/Home/Vehicles/Cars/2014_SS/Model Overview/02_Pdf/MY14 SS eBrochure_111313.pdf

- **Sistema de parqueo Nissan**

El nuevo Asistente de Aparcamiento Automático de Nissan es capaz de aparcarse vertical y horizontalmente maniobrando automáticamente el volante y dejando al conductor la responsabilidad de seleccionar las marchas accionar el freno o el acelerador. También indica en la pantalla imágenes exteriores, la distancia entre el vehículo, diferentes obstáculos y la posición exacta del vehículo en un recuadro en el que se ubicará. Este sistema posee cuatro cámaras de alta resolución que se ubican en el frente, a los costados y en la parte trasera del coche que proveen una

visión de 360º. Asimismo posee unos sensores ultrasónicos en las esquinas del automóvil, una alarma avisa al conductor cuando se acerca a un obstáculo fijo como se observa en la figura 2.8 [14].

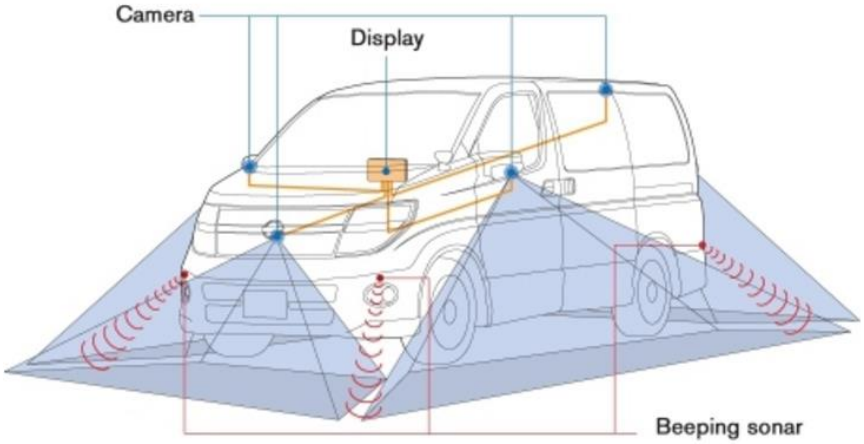


Fig. 2.8 Sistema Nissan
Fuente: http://www.pocket-lint.com/news/83561-nissan-camera-infiniti-elgrand/gallery?photo=2

- **Sistema de parqueo Kia**

El sistema de Kia reduce el esfuerzo de aparcarse horizontalmente, reconoce el espacio adecuado y le guía automáticamente hacia él calculando la distancia con los vehículos de adelante y de atrás, gracias a sus sensores ultrasónicos instalados en la parte trasera y delantera del automóvil como se muestra en la figura 2.9.

Fig. 2.9 Kia cee'd Sportswagon
Fuente: http://www.kia.com/es/campaigns-and-redirects/gama-ceed/

El conductor regula la velocidad, cambia las marchas y lo frena. La cámara trasera le permite ver en la pantalla LCD lo que se encuentra detrás del vehículo [15].

2.2.2 Sistema electrónico

Un sistema electrónico es un conjunto de dispositivos que interactúan entre sí, cuyo funcionamiento depende del flujo de electrones los cuales generan, reciben, transmiten y almacenan información [16].

2.2.3 Software Libre

Software libre es aquel software en donde el usuario puede acceder al código fuente para ejecutarlo, copiarlo, distribuirlo, estudiarlo y mejorarlo. También, el usuario puede modificarlo de acuerdo a sus necesidades y redistribuirlo para que otras personas lo puedan utilizar, modificar y nuevamente redistribuirlo con el código fuente y con las modificaciones respectivas del código.

La Fundación para el Software Libre (FSF, *Free Software Foundation*) ha definido cuatro parámetros que todo software debe brindar al usuario para que sea admitido como libre:

- Libertad de ejecutar el software.
- Libertad de acceder al código fuente, estudiarlo y adaptarlo de acuerdo a las necesidades.
- Libertad de distribuir copias del software.
- Libertad de mejorar el software y distribuir las modificaciones a otros usuarios para su utilización [17].

2.2.4 Android

Android sigue siendo el sistema operativo más utilizado en *Smartphone* seguido por iOs y *Windows Phone*. En Estados Unidos, Android controla el 63.6 % mientras que Apple controla solo el 27.1% y *Windows Phone* el

3.2%. En cuanto a Latinoamérica, Android controla el 86.4%, iOS el 2.8% y *Widows Phone* el 5.2% [18].

Android es un sistema operativo el cual está basado en Linux para *Smartphone, tablets, netbooks*, PC's, etc. Android se programa en un entorno de trabajo llamado *framework* de Java, las aplicaciones se realiza sobre una máquina virtual *Dalvik* con compilación en tiempo de ejecución. Se diferencia con otros sistemas operativos debido a que cualquier persona que programe en Java puede realizar sus propias aplicaciones, widgets y modificar el propio sistema operativo de acuerdo a sus necesidades, además es de código libre y muy fácil de programar.

En la figura 2.10, se distingue 4 capas del sistema operativo Android, en la primera capa se encuentra el Kernel de Linux, en la segunda se encuentran las librerías para el desarrollo de aplicaciones, la tercera capa organiza los diferentes administradores de recursos y por último se encuentra la capa de aplicaciones a las cuales se tiene acceso [19].

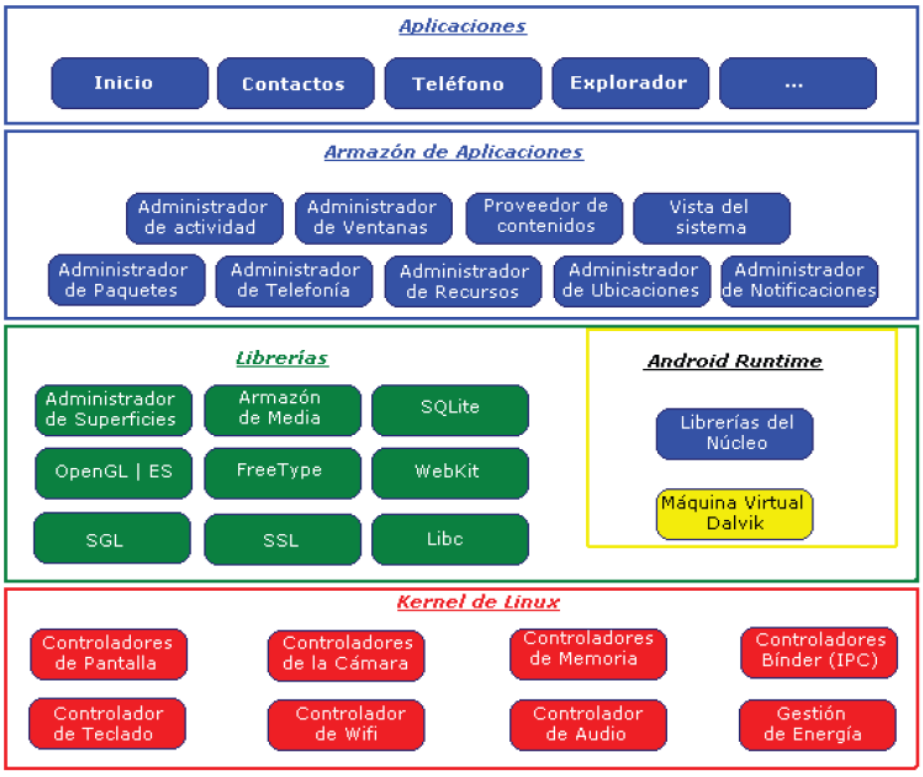


Fig. 2.10 Capas del sistema operativo Android
Fuente: http://faciluso.es/guias-rapidas/primeros-pasos-en-android

2.2.5 Eclipse

La plataforma Eclipse es un Entorno de Desarrollo Integrado como se observa en la figura 2.11 (IDE, *Integrated Development Environment*) desplegable y abierto. Eclipse sirve como IDE de Java con numerosas herramientas de desarrollo, proporciona soporte a otros tipos de lenguaje como: C/C++, Cobol, Fortran, PHP o Python. También se le puede añadir extensiones (plugins).

Fig. 2.11 Logo de Eclipse
Fuente: http://www.moelia.com/what-is-eclipse-rcp-and-why-should-i-care/

En la actualidad Eclipse tiene una versión estable que se encuentra disponible para los sistemas operativos Windows, Linux, Solaris, AIX, HP-UX y Mac OSX. Todas las versiones de Eclipse necesitan tener instalado en el sistema una máquina virtual Java (JVM), JRE (*Java Runtime Environment*) o JDK (*Java Developer Kit*). Esta plataforma se ha usado típicamente para desarrollar entornos de desarrollo integrados, como el IDE de Java llamado *Java Development Toolkit* (JDT) [20].

2.2.6 Hardware Libre

Se conoce hardware libre (electrónica libre) a los dispositivos cuyos diagramas esquemáticos y especiaciones son de acceso público. El término hardware libre se ha usado principalmente cuando se usa software libre con el hardware y proporcionan la información respectiva del hardware (diagramas esquemáticos, diseños, tamaños e información adicional). Cualquier persona o empresa puede fabricar el dispositivo y comercializarlo siempre y cuando este también sea libre [21].

2.2.7 Sistemas Embebidos

Un sistema embebido consiste en un sistema de computación cuyo software y hardware están específicamente diseñados y optimizados para resolver un problema en concreto. El término embebido (empotrado) es una parte integral del sistema (electrónica o sistema electrónico de control) en el que se encuentra.

Los sistemas embebidos se diferencian de los demás sistemas electrónicos ya que al estar insertados dentro del dispositivo que controlan, están sujetos en gran medida a cumplir requisitos como: tamaño, fiabilidad, consumo y costo [22].

En la figura 2.12 se muestra un diagrama de bloques del cual está compuesto un sistema embebido.

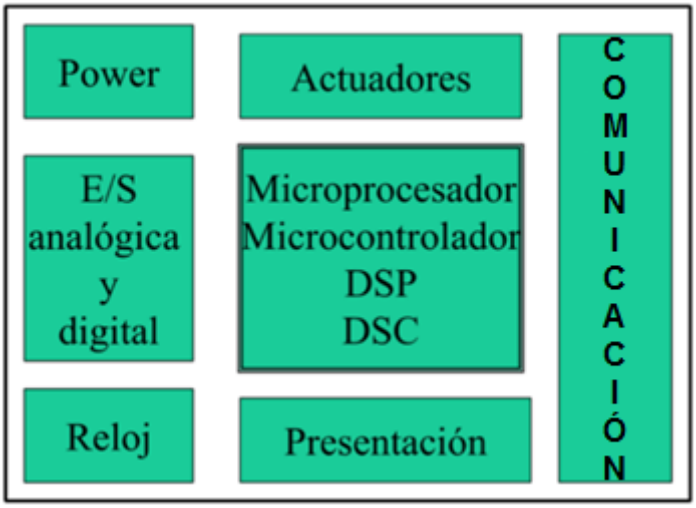


Fig. 2.12 Diagrama de bloques simplificado de un sistema embebido
Fuente: http://ocw.um.es/ingenierias/sistemas-embebidos/material-de-clase-1/ssee-t01.pdf

A continuación se describe brevemente cada bloque del sistema embebido.

- **Microprocesador, microcontrolador, DSP o DSC**

Son los elementos encargados de darle la inteligencia al sistema. La CPU puede ser uno o varios microprocesadores, microcontroladores, DSP o DSC.

La localidad de estos chips puede encontrarse dentro del propio dispositivo, de forma externa e incluso pueden ser de ambas formas.

- **Comunicación**

La comunicación es algo muy importante para el sistema ya que permite transferir la información hacia otros dispositivos o al computador, los sistemas embebidos tienen interfaces estándar de comunicación, por cable o inalámbricas. Entre los más importantes se tiene: RS-232, USB, Ethernet, Fibra Óptica, Wifi, Bluetooth, GPS, etc.

- **Presentación**

Por lo general la presentación suele ser una pantalla gráfica, táctil, LCD, diodos LED, etc. La pantalla táctil es la presentación más aceptada por los usuarios pero el inconveniente es que lleva un desarrollo de software más complejo y se necesita más potencia en los procesos de cálculo en la CPU.

- **Actuadores**

Son los dispositivos encargados de recibir la información enviada por la CPU y llevarlas hacia el exterior. Se tienen diferentes actuadores los cuales pueden ser: corriente, relés, conmutadores, etc.

- **Pines de E/S analógicos y digitales**

El bloque de entradas y salidas (I/O) se encarga de enviar o recibir las señales ya sea análogas o digitales a todos los circuitos encargados de su generación y procesamiento. Por ejemplo la conversión, la activación de actuadores, conocer si un interruptor o pulsador está abierto o cerrado presionado o suelto, la conversión de la señal de un sensor para poder procesarla, el estado de un LED, etc.

- **Reloj**

Es el encargado de generar las distintas señales de reloj (clock) necesarias para la temporización de todos los circuitos digitales. Generalmente consta de un solo oscilador principal, sus características son muy importantes para el buen funcionamiento de aplicaciones.

- **Módulo de alimentación (Power)**

El módulo de alimentación es el encargado de proporcionar las tensiones y corrientes necesarias para que el sistema embebido funcione. Se lo puede alimentar de diferente forma, con baterías por medio de un convertidor de AC/DC o por medio de un cable (USB), igualmente existen dispositivos los cuales varían la tensión de entrada de acuerdo a las necesidades, entre ellos se tiene: 5, 9, 12 y 24 voltios continuos (Vdc) [23].

2.2.8 Raspberry Pi

La Raspberry Pi como se observa en la figura 2.13 es una mini computadora basada en la arquitectura ARMv6, un GPU de Broadcom, Decodifica videos en Full HD, consta de puertos USB 2.0, tiene salida HDMI, puerto Ethernet, un lector de memorias y por último tiene una salida de audio estándar de 3.5 mm. Corre una distribución de Linux especial para el dispositivo con entorno gráfico. Fue pensada para presentar a los estudiantes jóvenes y niños la programación de una computadora [24].

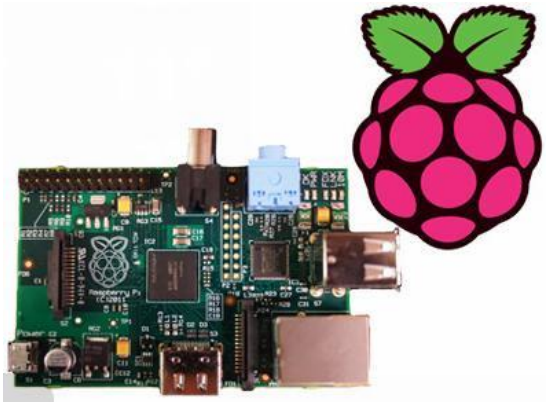


Fig. 2.13 Logo y placa de Raspberry pi
Fuente: http://www.unocero.com/2012/07/16/para-quien-quiera-comprar-su-raspberry-pi/

Hardware Raspberry Pi

La Raspberry Pi modelo B tienen unas dimensiones de 8.5cm por 5.3cm en el centro de la placa tiene un chip Broadcom BCM2835 con un procesador ARM11, proporciona frecuencias variables y brinda la posibilidad de subirla (overcloking) hasta 1 GHz, tienen un procesador gráfico VideoCore IV y memoria RAM de distintas cantidades como se muestra en la figura 2.14.

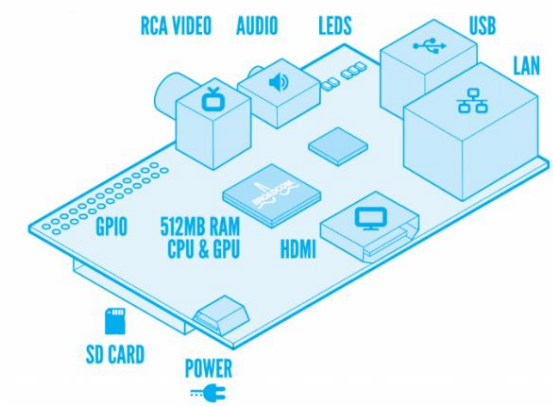


Fig. 2.14 Raspberry Pi modelo B
Fuente: http://www.raspberrypi.org/help/faqs/#introWhatIs

Las últimas Raspberry Pi tienen una memoria RAM de 512 MB la cual equivale a entornos gráfico como el XBOX de Microsoft y la reproducción de video de 1080p. Tiene salida de video y audio por medio de un conector HDMI pudiendo conectarlo a una tv o a un proyector, además cuenta con salida de video compuesto y una salida de audio (minijack). Conexión Ethernet 10/100, pero si se necesita conexión Wifi solo se debe conectar a cualquier puerto USB para tener internet inalámbrico. Los puertos tienen una corriente limitada ya que si se desea conectar un hub se necesita de una alimentación externa para los dispositivos.

Software Raspberry Pi

Raspbian OS es el sistema operativo otorgado por Raspberry Pi como se muestra en la figura 2.15. Para poder instalarlo se debe colocar en una tarjeta SD la cual se introduce en la ranura Raspberry Pi.Risc Os Pi, el sistema operativo se obtiene descargando desde la página oficial y para poder instalarlo en la tarjeta se necesita de un programa como Fedora Remix. Igualmente existen otros programas que se encarga de realizar todo el trabajo de instalación desde la Raspberry Pi [25].

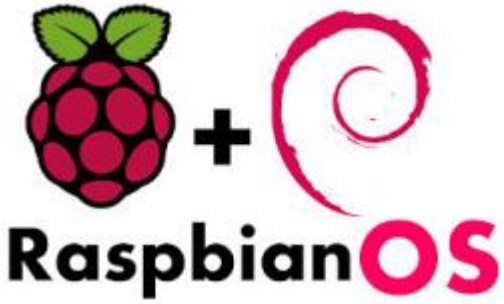


Fig. 2.15 Sistema operativo Raspbian
Fuente: http://www.fluceando.com/blog/presentacion-de-la-web/

2.2.9 Arduino

Arduino como se observa en la figura 2.16 es una placa electrónica para la creación de prototipos las cuales están basadas en software y hardware libre lo que le hace que sea una plataforma flexible y fácil de utilizar. Arduino fue diseñado para desarrolladores, aficionados y cualquier persona interesada en crear proyectos interactivos.

Fig. 2.16 Arduino Uno
Fuente: http://media.digikey.com/Photos/Arduino/A000046.jpg

Arduino es una plataforma que toma información de su alrededor por medio de sus pines de entrada, esto se logra por medio de sensores los cuales accionan actuadores pudiendo controlar luces, motores, etc. La placa Arduino se programa con el lenguaje de programación proporcionado por la misma placa el cual está basado en Wiring y consta de un entorno de desarrollo basado en Processing. Los programas realizados en Arduino se pueden ejecutar sin necesidad de que esté conectado al ordenador, además puede comunicarse con diferentes tipos de software: MaxMSP, Processing, Flash, etc. [26].

Tipos de Arduinos

En la actualidad existe una gran variedad de Arduinos para todas las necesidades tales como:

- Arduino Uno
- Arduino Mini
- Arduino Leonardo
- Arduino Mega
- Arduino Due

Estos dispositivos son los más usados pero también existen otros dispositivos llamados shield los cuales les proporcionan más capacidades y más aplicaciones para los diferentes Arduinos.

2.2.10 UDOO

UDOO es un mini PC que se puede utilizar tanto con Android y el sistema operativo Linux, con un tablero compatible con Arduino Due incrustado en la misma placa. UDOO es una placa de prototipos de gran alcance para desarrollo y diseño de software, es fácil de usar y permite el desarrollo de proyectos con conocimientos mínimos de hardware.

Puede funcionar con diferentes entornos informáticos juntos, cada uno tiene sus puntos propios de fuerzas y debilidades, todos ellos son útiles en la vida de hoy en día para los propósitos educativos y la realización de prototipos de una manera rápida. UDOO es un hardware abierto, plataforma de bajo coste equipado con un Procesador ARM i.MX6 Freescale y una sección compatible con Arduino Due basado en un Procesador ATMEL SAM3X ARM, todo esto disponible en la misma placa como se indica en la figura 2.17.

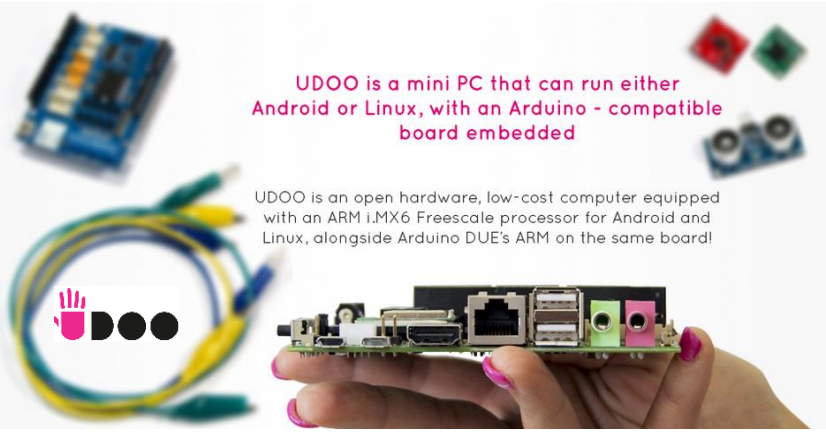


Fig. 2.17 Placa UDOO
Fuente: http://www.udoo.org/

Además UDOO combina un Arduino Due con 4 Raspberry Pi, con 1 GB de RAM, con salida HDMI, conectividad WiFi, Mini USB y MicroSD. La máquina arranca de la tarjeta MicroSD, lo que significa que para arrancar con otro sistema operativo simplemente hay que cambiar de tarjeta. En el ANEXO 3 se puede observar los puertos que tiene la placa y la distribución de los pines del Arduino DUE.

Especificaciones

UDOO tiene una línea que está compuesto de tres modelos, compartiendo muchas de las características, diferente sólo para la conectividad y de los procesadores. Los tres modelos cuentan con un Arduino incrustado con una sección compatible basada en Arduino Due. El tamaño de UDOO es de 11cm x 8,5cm. Dispone de varios pines de E/S que funcionan solo con 3.3V.

Tipos de UDOO

A continuación se describen los tres modelos de la placa UDOO con sus respectivas características técnicas:

- **UDOO Dual Basic**

Es la primera versión de la placa UDOO consta con casi todas las características de las otras versiones como se indica en la figura 2.18. El mayor inconveniente de esta placa son sus microprocesadores los cuales son un poco lentos lo que dificulta el proceso de visualización de la cámara y la trasferencia de datos entre un microprocesador y otro.

Fig. 2.18 Placa UDOO Dual Basic
Fuente: http://shop.udoo.org/other/product/udoo-dual-basic.html

Entre sus características técnicas se tiene:

- Freescale i.MX6DualLite, 2x ARM Cortex-A9 núcleo a 1 GHz con sistema de instrucción ARMv7A.
- GPU Vivante GC 880 para 3D y 2D (gráficos vectoriales) + Vivante GC 320 para 2D (composición).
- CPU Atmel SAM3X8E ARM Cortex-M3 (igual que el Arduino Due).
- RAM DDR3 1GB.
- 76 GPIO totalmente disponible y compatible con Arduino R3 1.0.
- HDMI y LVDS + Touch.

- **UDOO Dual**

Esta placa es muy parecida a la UDOO Dual Basic tiene una ventaja ante la otra y es que posee un módulo Wifi instalado como se muestra en la figura 2.19. Pero el problema sigue siendo el mismo que el anteriormente mencionado tiene unos microprocesadores un poco lentos.

Fig. 2.19 Placa UDOO Dual
Fuente: http://shop.udoo.org/other/product/udoo-dual.html

Sus características técnicas son las siguientes:

- Freescale i.MX6DualLite, 2x ARM ® Cortex-A9 núcleo a 1 GHz con un sistema de instrucciones ARMv7A.
- GPU Vivante GC 880 para 3D y 2D (gráficos vectoriales) + Vivante GC 320 para 2D (composición).
- CPU Atmel SAM3X8E ARM Cortex-M3 (igual que el Arduino Due).
- RAM DDR3 1GB.
- 76 GPIO totalmente disponible y compatible con Arduino R3 1.0.
- HDMI y LVDS + Touch.
- 2 Micro USB (1 OTG).
- 2 USB 2.0 Tipo A y 1 Conector macho interno USB 2.0 (requiere cable adaptador).
- Audio analógico y Mic jacks.
- Cámara CSI Conexión.
- Lector de tarjetas Micro SD (dispositivo de arranque).
- Fuente de alimentación (6-15V) y el conector de batería externa.

- Ethernet RJ45 (10/100/1000 Mbit).
- Módulo WiFi.

➢ **UDOO Quad**

Por último se tiene la UDOO Quad la ventaja que posee ante sus antecesores es la de ser dos veces más rápida lo cual hace que la información entre los dos microprocesadores sean más rápidos y exactos como se observa en la figura 2.20.

Fig. 2.20 Placa UDOO Quad
Fuente: http://shop.udoo.org/other/product/udoo-quad.html

Sus características técnicas son las siguientes:

- Freescale i.MX6Quad, 4 x ARM Cortex-A9 núcleo a 1 GHz con sistema de instrucción ARMv7A.
- GPU Vivante GC 2000 para 3D + Vivante GC 355 para 2D (gráficos vectoriales) + Vivante GC 320 para 2D (composición).
- CPU Atmel SAM3X8E ARM Cortex-M3 (igual que el Arduino Due).
- RAM DDR3 1GB.
- 76 GPIO totalmente disponible y compatible con Arduino R3 1.0.
- HDMI y LVDS + Touch.
- 2 Micro USB (1 OTG).
- 2 USB 2.0 Tipo A y 1 Conector macho interno USB 2.0 (requiere cable adaptador).
- Audio analógico y Mic jacks.

- Cámara CSI Conexión.
- Lector de tarjetas Micro SD (dispositivo de arranque).
- Fuente de alimentación (6-15V) y el conector de batería externa.
- Ethernet RJ45 (10/100/1000 Mbit).
- Módulo Wifi.
- Conector SATA con cabezal de alimentación [27].

2.3 PROPUESTA DE SOLUCIÓN

Con este proyecto se pretende obtener un sistema para el estacionamiento de vehículos livianos evitando daños y reduciendo el tiempo al momento de estacionarse, teniendo una amplia perspectiva de la parte trasera del automóvil.

CAPÍTULO III

METODOLOGÍA

3.1 MODALIDAD DE LA INVESTIGACIÓN

Se realizó una investigación de campo la cual ayudo a determinar los problemas, con el fin de obtener y recopilar información que fue útil para el desarrollo del proyecto y de los objetivos planteados.

Igualmente se realizó una investigación bibliográfica – documental mediante libros, revistas, artículos y tesis para poder guiarse y profundizar en el tema, esta información sirvió como sustento científico para el proyecto, aplicando teorías y diversos criterios de los autores acorde a los requerimientos del proyecto.

3.2 RECOLECCIÓN DE INFORMACIÓN

Para la recolección de información se buscó en libros, artículos, revistas, tesis e internet, estos permitieron recolectar los datos necesarios requeridos para desarrollar los objetivos específicos.

Para la recolección de datos técnicos se reunió información de acuerdo a la variable medida ya sea este voltaje, corriente o resistencia. Se utilizaron dispositivos comunes como voltímetro, amperímetro o multímetro.

3.3 PROCESAMIENTO Y ANÁLISIS DE DATOS

Para la realización del procesamiento y análisis de datos se llevaron a cabo los siguientes parámetros:

- Recolección de información mediante libros, artículos, tesis, etc.
- Revisión la información indispensable para el proyecto.
- Lectura minuciosa de la información, que ayudó a plantear estrategias para la solución del problema.
- Interpretar los resultados

3.4 DESARROLLO DEL PROYECTO

Para el desarrollo del proyecto se llevaron a cabo los siguientes pasos:

- Investigación de los sistemas de estacionamientos que se encuentran instalados actualmente en los vehículos.
- Análisis y selección de los diferentes tipos de sensores que existen, como funcionan, sus características y para qué sirven cada una de ellas.
- Selección de la ubicación correcta de los sensores que se van a colocar en el vehículo.
- Acondicionamiento y procesamiento de las señales de los sensores.
- Diseño de una interfaz para visualizar la distancia medida por los sensores.
- Implementación del prototipo
- Pruebas del sistema

CAPITULO IV

DESARROLLO DE LA PROPUESTA

4.1 ANTECEDENTES

Cada vez más se incrementa el número de automóviles en la ciudad de Ambato, en el año 2012 se han matriculado 56 mil vehículos, 9 mil más que el año 2011 es una de las ciudades con más vehículos llegando en el 2013 a tener en circulación 65 mil autos. Además el 89.4% de esta cifra son vehículos livianos. Con el incremento de los vehículos los parqueaderos no pueden abastecer la demanda de los vehículos y los conductores se ven en la necesidad de estacionarse en lugares muy reducidos provocando accidentes y choques [28].

Este sistema de ayuda para el estacionamiento permite tener una visión de la parte trasera del vehículo el cual ayuda al conductor a evitar obstáculos o a determinar en qué momento hay que detener el vehículo para evitar un choque. En el Ecuador algunos de los automóviles modernos poseen ya con un asistente para estacionarse, igualmente existen unos kits para colocarlos en algunas marcas de carros pero no en todos. Este sistema consta de sensores ultrasónicos, una cámara, un zumbador y una placa la cual procesa los datos recibidos y los envía a una pantalla para poder ser vistos por el conductor, este proyecto se basa en software y hardware libre lo cual da soporte para que cualquier persona pueda utilizarlo, construirlo e incluso rediseñarlo de acuerdo a sus necesidades.

4.2 ÁNALISIS DE FACTIBILIDAD

4.2.1 Factibilidad Técnica

El proyecto de investigación técnicamente es factible debido a que la mayoría de los elementos empleados se encuentran en el país, mientras que los dispositivos faltantes se pueden conseguir por medio de internet sin ningún problema.

4.2.1 Factibilidad Económica

El desarrollo del proyecto es económicamente factible debido a que toda la parte económica fue financiada por el investigador.

4.2.1 Factibilidad Bibliográfica

El proyecto es factible puesto que existe bastante información en libros, revistas y especialmente en internet.

4.3 ANÁLISIS DE REQUERIMIENTOS

Después de haber investigado los sistemas de estacionamiento tanto los que se comercializan como los propios de cada marca de vehículo se ha determinado que estos sistemas disponen de tres partes fundamentales que son: el cerebro del sistema, la parte del sensado y la presentación.

De acuerdo a esto se ha llegado a tener los requerimientos que necesita el proyecto.

4.3.1 Software

Para llevar a cabo el proyecto se necesita de un sistema operativo libre que permita instalar aplicaciones o programas propios, sea fácil de utilizar, asimismo que pueda interactuar con dispositivos electrónicos. Igualmente se necesita de un entorno de desarrollo potente en donde realizar la aplicación el cual permita utilizar todos los recursos que tenga el sistema operativo a emplear.

Además se requiere de un programa que contenga todas las características necesarias para programar los dispositivos electrónicos y pueda comunicarse con el sistema operativo elegido para realizar la aplicación.

4.3.2 Hardware

Para la instalación del sistema de estacionamiento en un vehículo liviano se necesitan de los siguientes dispositivos:

- **Una placa central:** esta placa debe ser potente la cual permita instalar un sistema operativo, pueda conectarse con diferentes tipos de sensores, una cámara de alta resolución y una pantalla, que sea rápida para enviar y recibir información entre los dispositivos externos y el sistema operativo, sea pequeña para ubicarlo en la cajuela del vehículo.

- **Sensores que midan distancias:** estos sensores deben tener un margen de error mínimo para evitar fallos al momento de obtener la medida, pueda medir distancias pequeñas o grandes, sea pequeño para poder instalarlo en el vehículo y lo más importante sea compatible con la placa elegida.

- **Cámara:** una cámara de alta resolución para tener una presentación fluida de video, al igual que los sensores, sea compatible con la placa y su tamaño sea pequeña para ubicarla en la parte trasera del automóvil.

- **Pantalla:** una pantalla pequeña de 7 a 9 pulgadas para que pueda ser ubicada en el tablero del vehículo y sea compatible con la placa central.

4.2 COMPARACIÓN DE PLACAS EMBEBIDAS

Tabla 4.1 Características técnicas de cada placa embebida

	Arduino Yum	**BeagleBone Black**	**UDOO Quad**
Precio	$73	$49	$135
Tamaño	7,3 x 5,3 cm	8,6 x 5,4 x 1,7 cm	11 x 8,5 x 2 cm
RAM	64 MB DDR2	512MB DDR3	1GB
Velocidad de reloj	16 MHz	400 MHz	1GHz
Multitarea	No	Si	Si
Voltaje de entrada	5 V	5 V	6 - 15 V
Memoria Flash	32KB	2 GB	Micro SD(dispositivo de arranque)
CPU	Ninguna	AM335x 1GHz ARM Cortex-A8	Freescale i.MX 6 ARM Cortex-A9 Quad core
Conectividad de red	10/100Mbit/s, Wifi	10/100Mbit/s	Ethernet RJ45 (10/100/1000 MBit), WiFi
Puertos USB	Una USB Tipo-A 2.0	Una USB Tipo-A 2.0	Dos USB tipo A(x2)
Sistema operativo	OpenWrt-Yun (Linux)	Linux/Android 4.0	Distribuciones de Linux y Android
IDE	Arduino	Cloud9 IDE en Node.JS w	Arduino, Scrath, IDLE, Linux
Microcontrolador	ATmega32u4	ARM Cortex-M3	Atmel SAM3X8E ARM Cortex-M3
Entradas/Salidas GPIO	Ninguna	67 GPIO disponibles	76 GPIO disponibles
Entradas/Salidas digital	20 (7 como salida PWM)	65 (3.3V)	62
I/O analógica	12 entradas	7	14

Fuente: Investigador

Después de haber investigado cada uno de las placas y la que más se acerca a los requerimientos del sistema de estacionamiento es la placa UDOO Quad puesto que consta con una memoria de 1G, trabajo a una frecuencia de 1GHz, tiene un Arduino en la misma placa, corre un sistema operativo Linux o Android lo que hace que interactúe el hardware y software a la vez.

4.4 COMPARACIÓN DE LOS SISTEMAS OPERATIVOS UTIZADOS EN LA PLACA UDOO

4.4.1 UDOObuntu (Oficial Distribución Lubuntu 12.04 LTS)

Lubuntu es un sistema operativo rápido y ligero. El núcleo del sistema está basado en Linux y Ubuntu. Consta de una gran velocidad y eficiencia energética. Debido a esto, Lubuntu tiene requisitos muy bajos de hardware. Lubuntu fue fundada por Mario Behling y actualmente es desarrollado principalmente por Julien Lavergne. [29].

4.4.2 Android 4.3 Jelly Bean

El sistema operativo Android es actualmente la plataforma más popular del mundo para móviles inteligentes (representan el 57,6% del mercado), igualmente permite realizar aplicaciones y comercializarlas.

Tiene un gran soporte, una comunidad que proporciona actualizaciones y ayudan a resolver problemas que se presentan al desarrollar una aplicación. Android es una plataforma móvil y flexible que se utiliza en *Smartphone, tablets*, TV, electrodomésticos, computadoras, etc. Utiliza un lenguaje de programación muy conocido como es JAVA, por lo tanto ya se puede tener una base para comenzar a programar en Android. [30].

Se ha decidido utilizar Android para realizar la aplicación puesto que es compatible con la Placa UDOO, es de fácil uso y puede interactuar con dispositivos electrónicos.

4.5 ENTORNOS DE DESARROLLO MÁS UTILIZADOS

Los IDE's (Entorno Integrado de Desarrollo) más utilizados por los desarrolladores son: Eclipse y NetBeans.

4.5.1 NetBeans

NetBeans es de fácil uso para realizar entornos gráficos simplemente arrastrando y pegando componentes ya viene con plugins y módulos integrados, evitando tener que configurar el ambiente, dando todo el entorno listo para trabajar.

4.5.2 Eclipse

Es uno de los entornos Java más utilizados a nivel profesional constituye con un IDE completo y adaptable, permite configurar el ambiente de desarrollo lo cual hace que sea versátil y altamente personalizable, vincula herramientas con una gran cantidad de plugins como módulos independientes [31].

En si los dos IDE's son muy utilizados para desarrollar aplicaciones sin embargo se ha optado por utilizar el entorno de desarrollo Eclipse debido a que el plugin de Android para NetBeans todavía no funciona correctamente. Eclipse proporciona un ambiente integrado sin ser necesaria su instalación, facilita la instalación o vinculación de diferentes plugins, da la libertad de realizar un código propio sin depender de lo que el IDE genere como es NetBeans por lo tanto consume menos memoria y además de que existe una gran variedad de información para poder programar una aplicación en Android empleando Eclipse.

4.6 COMPARACIÓN DE SENSORES ULTRASÓNICOS

4.6.1 Sensor Ultrasónico HC-Sr04

Es un sensor utilizado especialmente para proyectos de robótica en la medición de distancias, evitar obstáculos y para otras aplicaciones. Una de las ventajas de este sensor es que incluye toda la circuitería en el mismo módulo como se ve en la figura 4.1, es potente pero barato compatible con Arduino y otras plataformas.

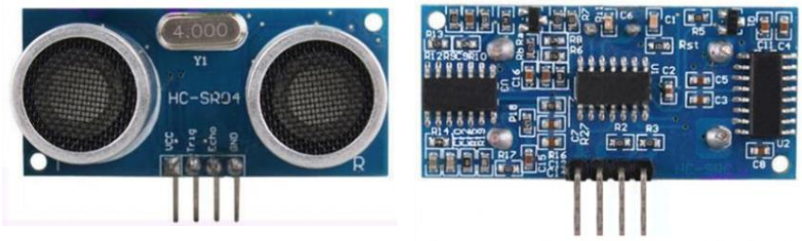


Fig. 4.1 Sensor HC-Sr04
Fuente: https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit?pli=1

Es uno de los sensores ultrasónicos más económicos y pequeños que existe en el mercado y sigue siendo uno de los mejores por su rendimiento, es estable, el error es tan bajo llegando a los 3mm, tiene una alta precisión y funciona con 5V (DC).

Tiene 4 pines los cuales son: “VCC” conectado a 5V, “Trig” conectado a un pin digital de la placa encargado de enviar el pulso ultrasónico, “Eco” conectado a un pin de entrada digital que recibirá el eco de dicho pulso y “GND” conectado a tierra, trabaja con una frecuencia de 40KHz con un ángulo de medición total de 60 grados [32].

4.6.2 Sensor LV-MaxSonar

Es un pequeño sensor por ultrasonido que detecta la presencia de un objeto y mide la distancia en un rango corto y largo. Estos sensores tienen integrado el transmisor y el receptor en la misma placa. Se tiene tres tipos de salidas: analógica, serial y PWM se alimenta con 5V DC, es compatible con Arduino, trabaja con una frecuencia de 42KHz, es el más pequeño

dentro de los sensores ultrasónicos, dependiendo de la clase del sensor (Z0, Z1, etc.) cuesta unas 4 o 5 veces más que el sensor HC-Sr04 como se observa en la figura 4.2 [33].

Fig. 4.2 LV-MaxSonar
Fuente: http://store.jdrones.com/product_p/mb1000.htm

4.6.3 Sensor TP-US03

Es un sensor ultrasónico diseñado exclusivamente como sensores de estacionamiento como se ve en la figura 4.3, en la actualidad existe una gran variedad de estos sensores para las diferentes marcas de autos pero en si todos funcionan de la misma manera, es de bajo costo, el transductor que lleva dentro funciona tanto como transmisor y receptor, funciona a 40KHz y es de tamaño pequeño, tiene un ángulo de cobertura total de 90 grados.

Fig. 4.3 Sensor TP-US03
Fuente: http://www.timberport.com/product.php?cat=4&id=124&m=3&b=10

La desventaja de este tipo de sensores es que necesita un circuito completo el cual realice todo el proceso de amplificación, comparación y multiplexación para que la señal obtenida entre a un microcontrolador y pueda ser utilizada [34].

En la tabla 4.2 se muestran las características más importantes de los sensores ya mencionados.

Tabla 4.2 Características técnicas de los sensores ultrasónicos

	HC-Sr04	**LV-MaxSonar Z0**	**TP-US03**
Precio	$ 10	$ 45	$ 5
Dimensiones	(45 x 20 x 15) mm	(19,9 x 21,1 x 16,4) mm	(22 x 25 x 20,5) mm
Frecuencia	40KHz	42KHz	40KHz
Ángulo de detección total	60º	60º	90^0
Voltaje de alimentación	5V	5V	12V
Corriente de trabajo	15mA	15mA	15mA
Rango mínimo detectable	2 cm	15,2 cm	20 cm
Rango máximo detectable	450 cm	600 cm	150 cm
Resolución (Error)	3 mm	2,54 cm	9 mm

Fuente: Investigador

Una vez comparado cada uno de los sensores e investigado sus características el sensor que más conviene para el proyecto de acuerdo a los requerimientos es el sensor ultrasónico HC-Sr04.

4.7 ELECCIÓN DE LA CÁMARA

En la tabla 4.3 se muestran las características más importantes de cada cámara.

Tabla 4.3 Características técnicas de las cámaras

	Cámara raspberry pi	**Cámara de UDOO**	**FaceCam 320X**
Precio	$ 25	$ 39	$ 10
Resolución	5 megapíxeles	5 megapíxeles	3 megapíxeles
Tamaño	(20x25x9) mm	(23x24x9) mm	(74x25x61) mm
Video	640x480p	VGA (320x480), (640x480) / QSXGA (2592x1944)	640x480
Soporta	1080p / 720p	1080p / 720p / 1280x960	-
Compatibilidad con Android	No	Si	No
Compatibilidad con UDOO	No	Si	Si
Otros	2592 x 1944 píxeles Imágenes estáticas	Mejora la relación señal a ruido (SNR), Sensibilidad de 600 mV / lux-sec, Contornos y colores más nítidos	-

Fuente: Investigador

Luego de haber analizado las características de cada cámara y de acuerdo con los requerimientos del sistema de estacionamiento se eligió la cámara de UDOO puesto que tiene una buena resolución, es compatible con la Placa UDOO y lo más importante basta con conectar la cámara en la placa y el sistema operativo Android lo reconoce automáticamente.

4.8 COMPARACIÓN DE PANTALLAS

En la tabla 4.4 se detallan las características de cada pantalla.

Tabla 4.4 Características técnicas de las pantallas

	KIT LCD 7" Touch	**KIT LCD 15,6"**	**KIT LCD 15,6" Touch**
Tamaño	(165x106,4x8,2)mm	(363,3x215,4x10,4)mm	(363,3x215,4x10,4)mm
Display	7 "TFT RGB	15,6 "	15,6 "
Resolución	800X480	1366X768 24bit	1366X768 24bit
Cable de Video	UDOO_VK-7T para UDOO	UDOO_VK-15 para UDOO	UDOO_VK-15T para UDOO y Cable USB para un tercer UDOO'USB
Otros	Junta adaptador LCD, Pantalla táctil I2C, Dual Touch	-	USB pantalla táctil capacitiva y controlador de presión Touch

Fuente: Investigador

Cada una de las pantallas tiene sus ventajas y desventajas sin embargo la pantalla escogida de acuerdo a sus características técnicas y requerimiento del proyecto es la pantalla de UDOO de 7 pulgadas.

4.9 DESCRIPCIÓN GENERAL DEL SISTEMA

El sistema electrónico de ayuda de estacionamiento tiene como finalidad alertar al conductor si existe algún objeto en la parte trasera y delantera del vehículo y avisarle por medio de una pantalla y de un zumbador que tan cercano se encuentra del objeto. El sistema tiene tres partes fundamentales que son: seis sensores con una cámara y un zumbador, una placa la cual es el cerebro del sistema y una pantalla. Cuando el sistema entra en funcionamiento los sensores

comienzan a detectar los objetos que se encuentras en la parte trasera y delantera del vehículo, estas señales entran a la placa y determina que objeto se encuentra más cercano al vehículo y lo presenta en la pantalla, además para darle una mejor perspectiva al conductor el sistema tiene un zumbador el cual suena con mayor o menor frecuencia dependiendo de la cercanía de los objetos y por último de una cámara trasera la cual completa el sistema dándole una visión del objeto detectado. El sistema comienza a funcionar al presionar el botón de parqueo que se encuentra en el tablero del vehículo.

En la figura 4.4 se observa el diagrama de bloques de funcionamiento del sistema de estacionamiento.

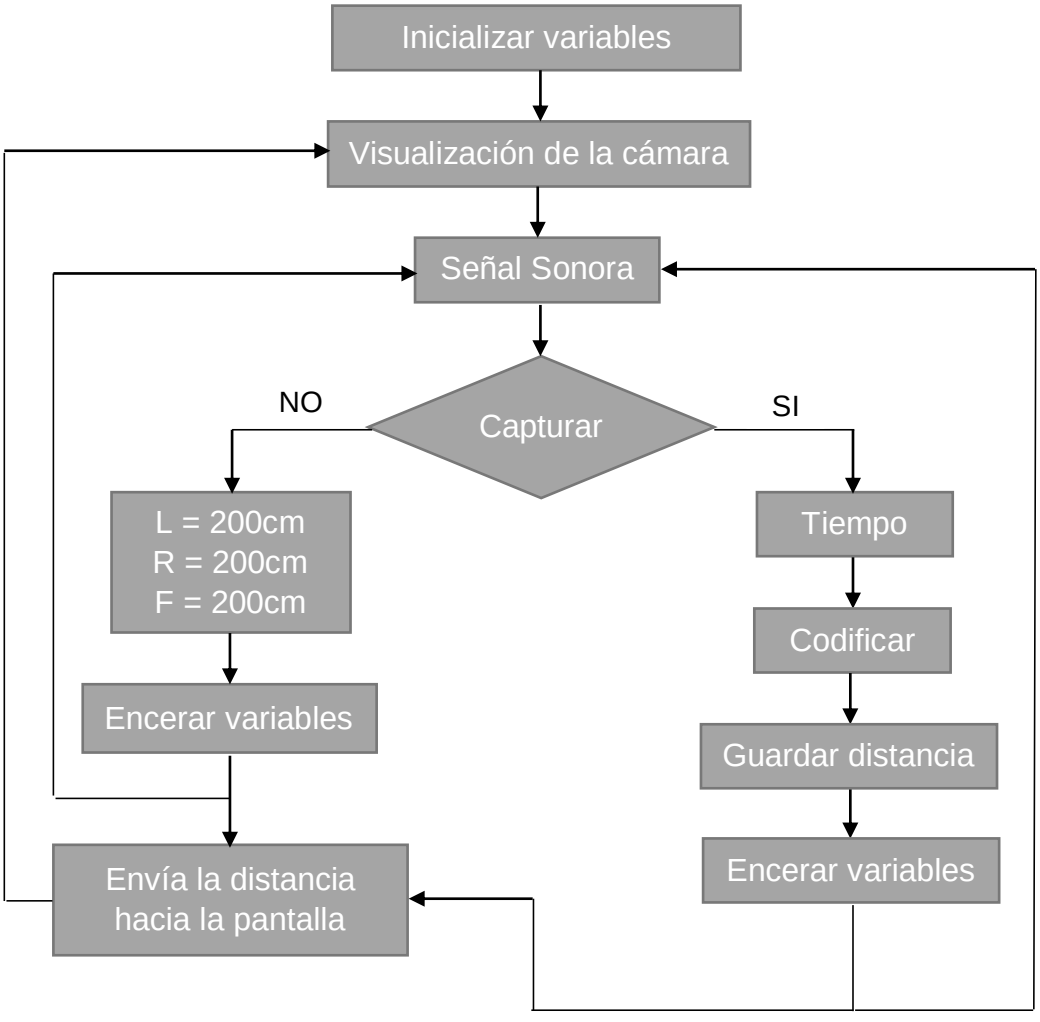


Fig. 4.4 Diagrama de bloques de funcionamiento del sistema de estacionamiento
Fuente: Investigador

4.10 INSTALACIÓN DEL SISTEMA OPERATIVO (ANDROID) EN LA PLACA UDOO

Primero se dispone de una tarjeta micro SD en la cual se coloca la imagen SD de Android 4.3 (Jelly Bean) de la página oficial de UDOO, luego de haber descargado y extraído se crea el sistema operativo de arranque desde la imagen. Dependiendo del sistema operativo a emplear ya sea Linux, Mac o Windows se deberá seguir los pasos correspondientes que indican en la página de UDOO.

4.10.1 Creación de una imagen en una tarjeta micro SD usando Windows

Para realizar una tarjeta micro SD de arranque se debe seguir los siguientes pasos:

1. Descargar el software Win32DiskImager
2. Descomprimir y obtener una carpeta llamada "win32diskimager-v0.7-binario"
3. Si la computadora tiene una ranura para tarjetas SD (micro SD adaptador necesario), se inserta la tarjeta. Si no es así, insertar la tarjeta en un lector de tarjetas SD, y luego conectar el lector al ordenador
   Nota: debe ser formateado en FAT32
4. Ejecutar el archivo llamado Win32DiskImager.exe (Windows Vista, 7 y 8 hacer clic derecho en el archivo y seleccionar "Ejecutar como administrador")
5. Si no se encuentra la tarjeta micro SD (*Device*) que se esté utilizando de forma automática hacer clic en el menú desplegable de la derecha y seleccionar la letra de la tarjeta micro SD que se acaba de conectar (por ejemplo, [H: \])
6. Tener cuidado al seleccionar la unidad correcta, al no hacerlo, puede destruirse los datos de cualquier disco duro de la computadora
7. En el cuadro de archivo de imagen, seleccionar el archivo img. descargado y presionar en "*Write*". Nota: si aparece un mensaje de advertencia, hacer clic en Yes

La tarjeta micro SD está lista para ser utilizada. Insertar en la UDOO y arrancará el sistema operativo Android [35].

4.11 CONFIGURACIÓN DE LA PLACA ARDUINO CON UDOO

Si se utiliza una máquina externa (Windows 7) para programar se debe seguir unos sencillos pasos los cuales son:

1. Verificar si una tarjeta SD de trabajo está presente en la placa UDOO
2. Desconectar el Puente J18 como se observa en la figura 4.5. Esto permitirá la comunicación entre el ordenador y el puerto de programación del SAM3X

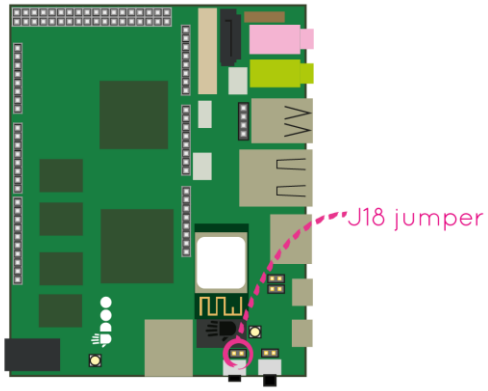


Fig. 4.5 Puente J18 de la UDOO
Fuente: http://www.udoo.org/ProjectsAndTutorials/get-the-arduino-ide-ready-to-program- udoo/?portfolioID=1394

3. Conectar el cable micro USB en el puerto Sam3x como se muestra en la figura 4.6.

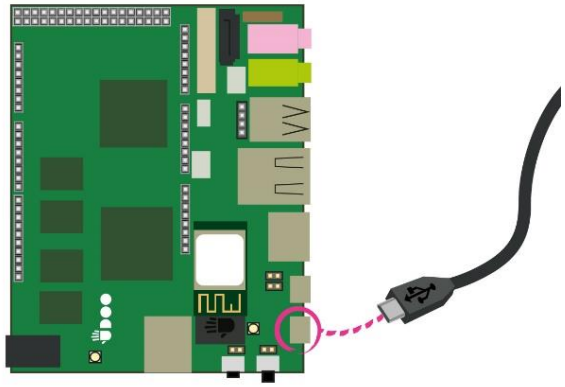


Fig. 4.6 Conexión de la mini USB
Fuente: http://www.udoo.org/ProjectsAndTutorials/get-the-arduino-ide-ready-to-program- udoo/?portfolioID=1394

Si se cumplen todos los requisitos anteriores, se está listo para comenzar. Ahora se configura el estándar de Arduino-IDE con el fin de hacer que se comuniquen el Sam3x de UDOO, que es el microcontrolador compatible con Arduino con el programador del entorno integrado de desarrollo y se sigue con los últimos pasos que son:

4. Descargar e instalar la versión Arduino IDE 1.5 para el sistema operativo específico desde la página oficial (otras versiones no funcionará, debido a que UDOO necesita el software de programación Arduino 2 compatible)
5. Descargar el parche de acuerdo al sistema operativo utilizado en este caso: Bossac Windows
6. Extraer los archivos y colocarlos en las siguientes rutas, sobrescribiendo los archivos existentes anteriores:
   En Windows 32 bits:
   C: \ Archivos de programa \ Arduino \ hardware \ tools
   En Win 64 bits:
   C: \ Archivos de programa (x86) \ Arduino \ hardware \ tools
7. Descargar el controlador serie: CP210x USB to UART Bridge VCP Drivers
8. Instalar la versión adecuada para el sistema operativo:
   CP210xVCPInstaller_x86.exe para sistemas de 32 bits
   CP210xVCPInstaller_x64.exe para el sistema de 64 bits

Con este parche y driver ahora se es capaz de subir los programas seleccionando el Arduino Due (Puerto de Programación) de: Herramientas - Placa - Arduino DUE (*Programming Port*).

Una vez realizado todo este proceso se podrá utilizar la UDOO como un Arduino normal, hay que tomar en cuenta que el Arduino DUE soporta solo voltajes de entrada y salida de hasta 3.3V [36].

4.12 INSTALACIÓN DEL ENTORNO DE DESARROLLO ECLIPSE

En la actualidad para realizar una aplicación en Android existen varios programas como: Android Studio, App Inventor, Plataforma Eclipse, etc. Dependiendo de la dificultad o de las propiedades de cada programa, la aplicación se podrá realizar o no, para la aplicación de este sistema se ha utilizado la plataforma Eclipse puesto que se necesita utilizar permisos para manipular la cámara y comunicar Android y Arduino a la vez.

Para realizar una aplicación de Android primero se debe instalar la plataforma Eclipse para ello se debe seguir los siguientes pasos:

1. Descargar el paquete completo del link https://developer.android.com/sdk/index.html el cual proporciona la plataforma Eclipse, SDK Android y ADT plugin.
2. Después de haber descargado y extraído el archivo zip se obtendrá una carpeta que en este caso está colocado en el Disco Local C como se indica en la figura 4.7.

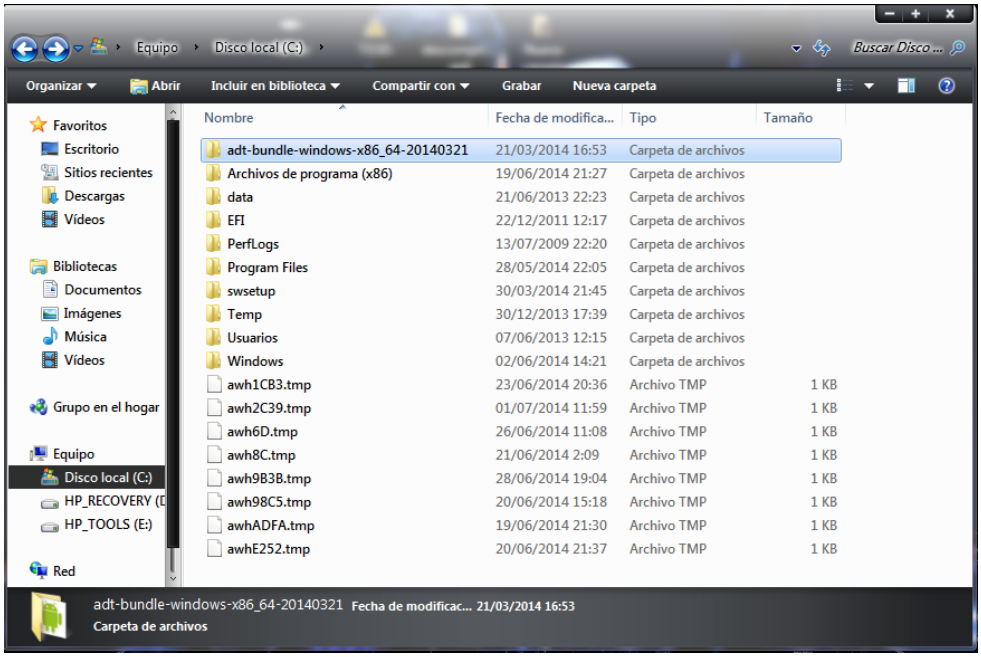


Fig. 4.7 Carpeta extraída a la PC
Fuente: Investigador

3. Se abre el programa Eclipse que se encuentra dentro de la carpeta llamada Eclipse como se observa en la figura 4.8.

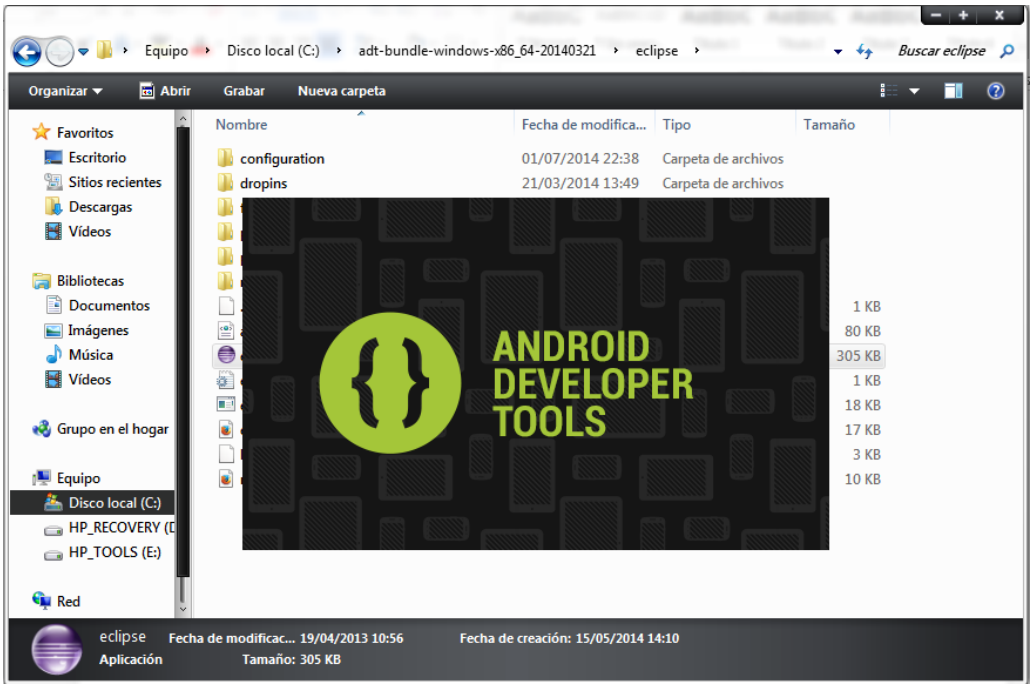


Fig. 4.8 Eclipse iniciando
Fuente: Investigador

4. Al arrancar Eclipse comenzará preguntando en que carpeta se quiere utilizar como *workspace* como se ve en la figura 4.9. En esta carpeta se almacenarán los proyectos creados en Eclipse. Es muy importante conocer su ubicación para poder realizar copias de seguridad de los proyectos. Presionar ok y está listo trabajar con Eclipse.

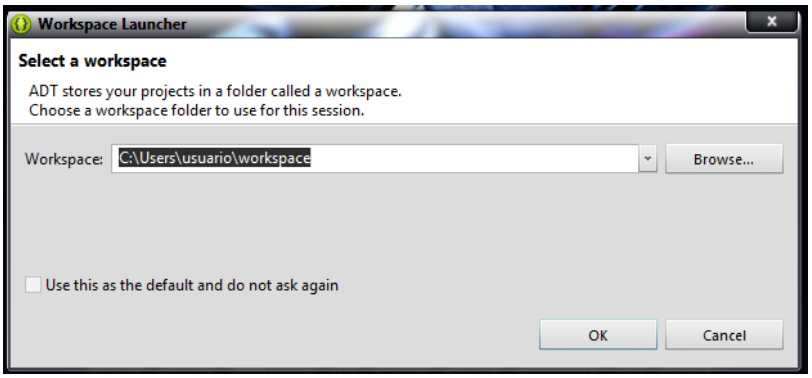


Fig. 4.9 Imagen donde se guardará los archivos creados en Eclipse
Fuente: Investigador

4.13 CREACIÓN DE UN PROYECTO ANDROID EN ECLIPSE

Para la creación de un proyecto se procese a realizar los siguientes pasos:

1. Seleccionar *File* > *New* > *Proyect* > *Android Application Proyect*, colocar el nombre del proyecto, seleccionar la versión SDK mínima en donde podrá correr la aplicación como se ve en la figura 4.10.

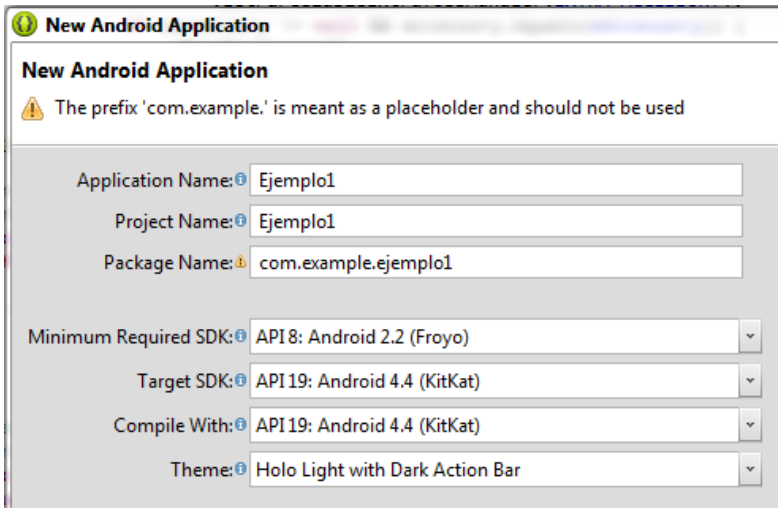


Fig. 4.10 Imagen en donde se ingresa los parámetros del proyecto
Fuente: Investigador

2. Click en Next y por último Finish y se obtendrá la siguiente pantalla como se muestra en la figura 4.11.

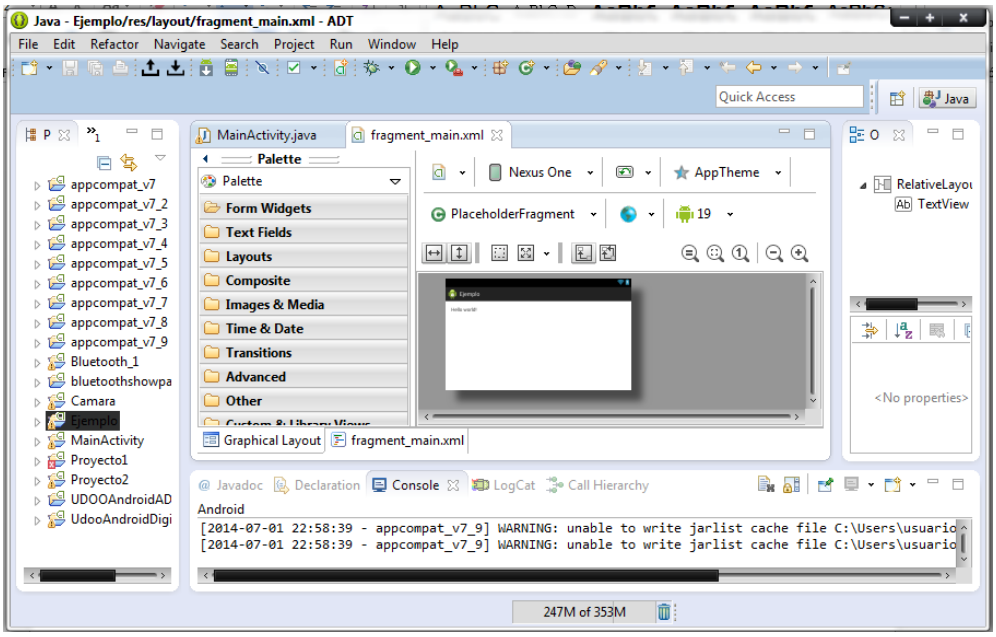


Fig. 4.11 Imagen después de haber creado un archivo
Fuente: Investigador

Existen dos iconos muy importantes en Eclipse el primero con una flecha de descarga donde se abrirá una pantalla en la cual se puede descargar cualquier Android disponible (API) como se muestra en la figura 4.12.

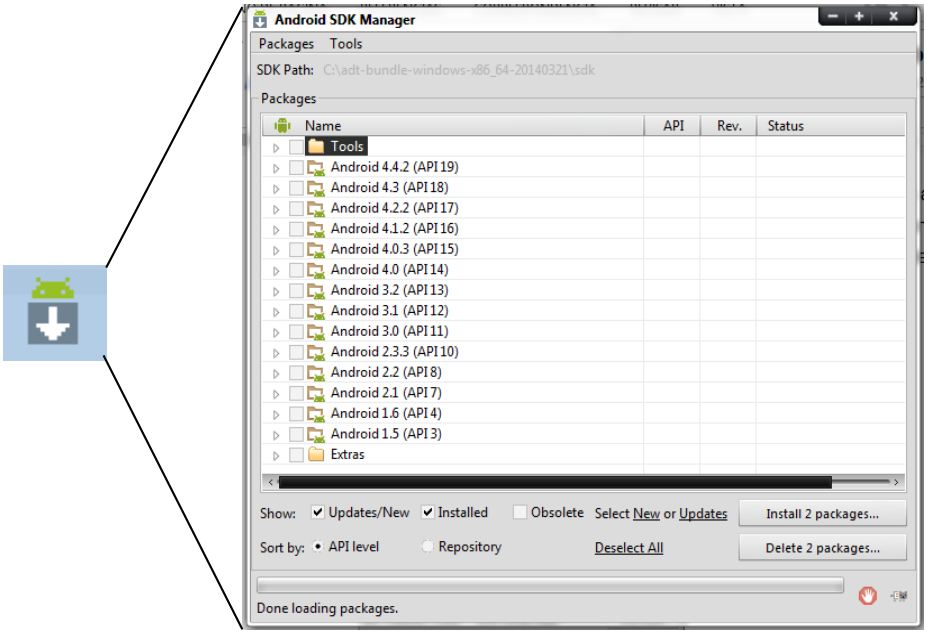


Fig. 4.12 API'S instaladas
Fuente: Investigador

El segundo proporciona un simulador de Android para poder cargar la aplicación y comprobar su funcionamiento como se indica en la figuras 4.13.

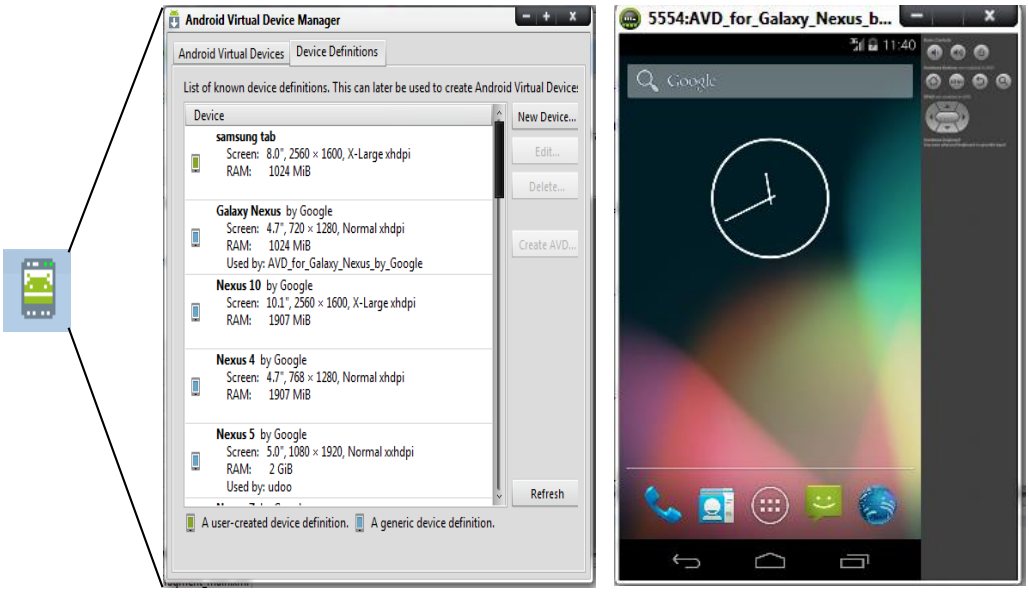


Fig. 4.13 Emulador de Android
Fuente: Investigador

4.14 CARACTERÍSTICAS DEL SENSOR ULTRASÓNICO HC-SR04

Este sensor incluye el transmisor ultrasónico, el receptor y el circuito de control.

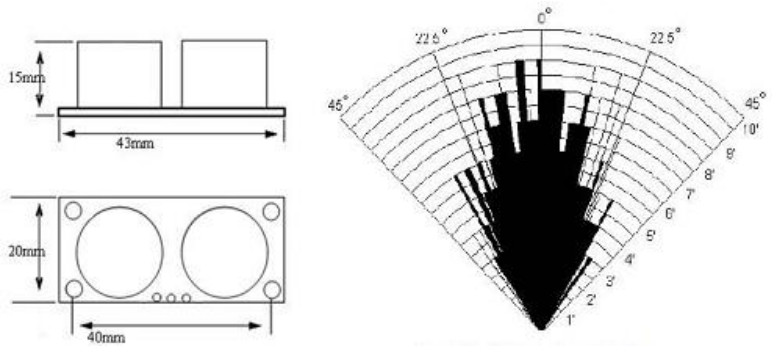


Fig. 4.14 Rango y medida del sensor HC-Sr04
Fuente: http://www.famosastudio.com/ultrasonic-range-sensor-hc-sr04

Como se observa en la figura 4.14 es un sensor pequeño y tiene un ángulo de cobertura de 60 grados a 200 centímetros mientras mayor sea la distancia menor será el ángulo de cobertura.

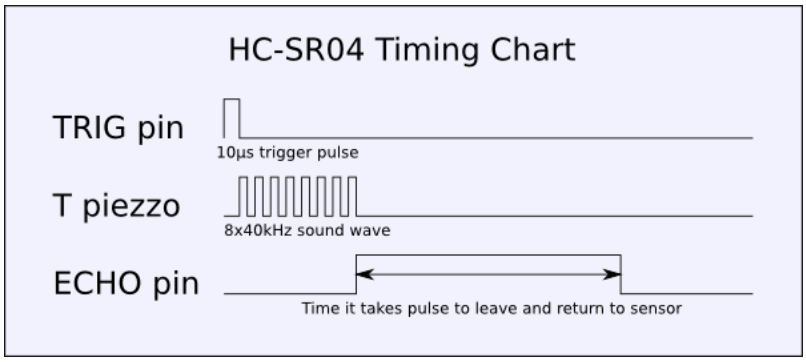


Fig. 4.15 Formas de onda enviadas por el sensor HC-Sr04
Fuente: http://www.famosastudio.com/ultrasonic-range-sensor-hc-sr04

En la figura 4.15 se observa las formas de onda la señal del Trigger, la frecuencia y el pin ECO.

4.15 FUNCIONAMIENTO DEL SENSOR ULTRASÓNICO HC-SR04

Para que el sensor ultrasónico pueda detectar un objeto debe ejecutar algunos parámetros que son:

- Primero envía un pulso "1" de 10uS por el Pin Trigger.
- Luego envía 8 pulsos de 40KHz y coloca su salida Echo a alto (seteo), en ese instante se detecta este evento e inicia un conteo de tiempo.

- La salida Echo se mantiene en alto hasta recibir el eco reflejado por el obstáculo a lo cual el sensor pondrá su pin Echo en bajo, es decir, termina de contar el tiempo.
- La distancia es proporcional a la duración del pulso y se puede calcular con la fórmula (1).

$$d = \frac{v*t}{2} \qquad \text{Fórmula (1)}$$

Donde:

V = velocidad del sonido (345m/s) o que es lo mismo 0.0345cm/microsegundos

T = tiempo del pulso del trigger 10 ms

La distancia total es la distancia que tarde en ir y regresar la señal por lo tanto al dividirlo para dos nos proporciona la distancia real del objeto detectado.

4.16 UBICACIÓN CORRECTA DE LOS SENSORES ULTRASÓNICOS Y DE LA CÁMARA

Como se observa en la figuras 4.16 el ángulo de visión del sensor y de la cámara es de 60 y 70 grados respectivamente con un área de detección máxima de 200cm determinado por el fabricante [37].

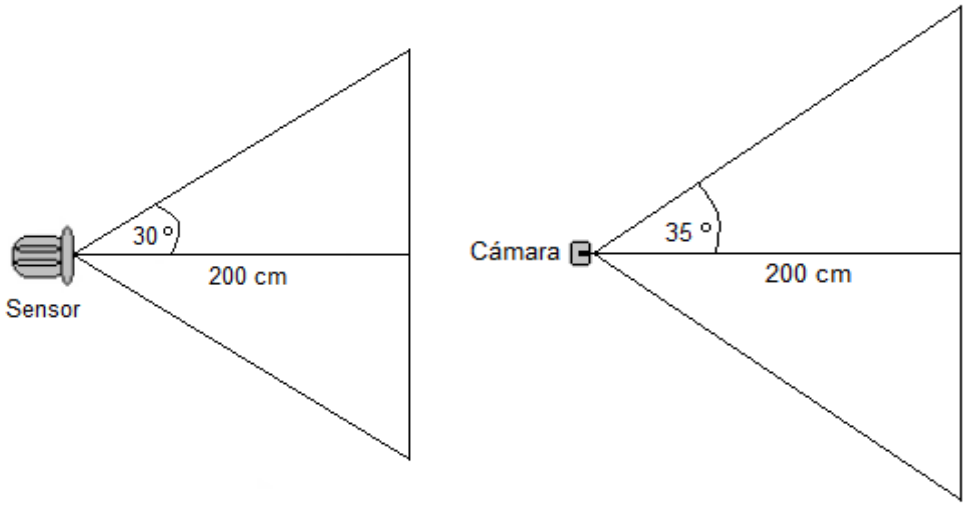


Fig. 4.16 Rango de detección del sensor HC-Sr04 y de la cámara UDOO
Fuente: Diseño e implementación de un sistema ultrasónico de ayuda para parqueo de vehículos.pdf

- **Ubicación de los sensores en la parte delantera del vehículo**

La distancia entre los sensores se determinó con la fórmula (2).

$$D = \frac{w}{n}$$ Fórmula (2)

Donde:

D = distancia entre sensores

w = ancho del vehículo 180cm (Calculado en el Anexo 4)

n = número de sensores 2

Por lo tanto la distancia entre sensores es *90 cm*, entonces de la parte central de vehículo se mide *45 cm* a la izquierda y *45 cm* a la derecha y se ubican los sensores como se observa en la figura 4.17.

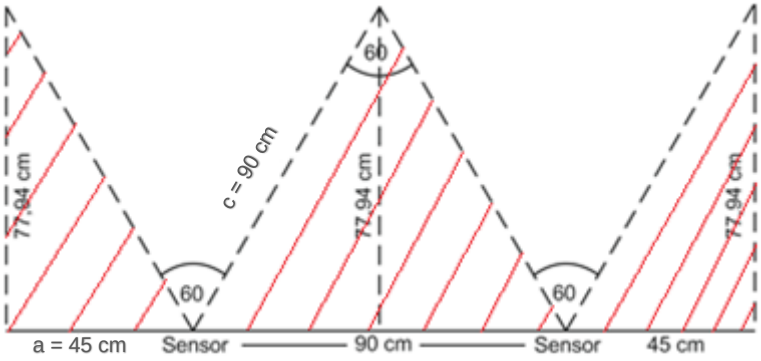


Fig. 4.17 Distancia de los sensores de la parte delantera del vehículo
Fuente: Investigador

La distancia entre el vehículo y el punto más lejano a detectar se determinó con la fórmula (3).

$$c = \sqrt{a^2 + b^2}$$ Fórmula (3)

Donde:

c = Hipotenusa (90 cm)

a = primer cateto (45 cm)

b = segundo cateto

Despejando *b* de la fórmula (3) se obtiene que *b = 77,94 cm.*

Como se observa en la figura 4.17 al ubicar los dos sensores se forma un triángulo (líneas rojas) en el cual no se podrá detectar ningún objeto ya que el ángulo de visión del sensor es de 60º, además se describe la distancia entre el vehículo y el punto más lejano donde no se podrá detectar ningún obstáculo.

Entonces la ubicación correcta de los sensores en la parte delantera del vehículo se muestra en la figura 4.18.

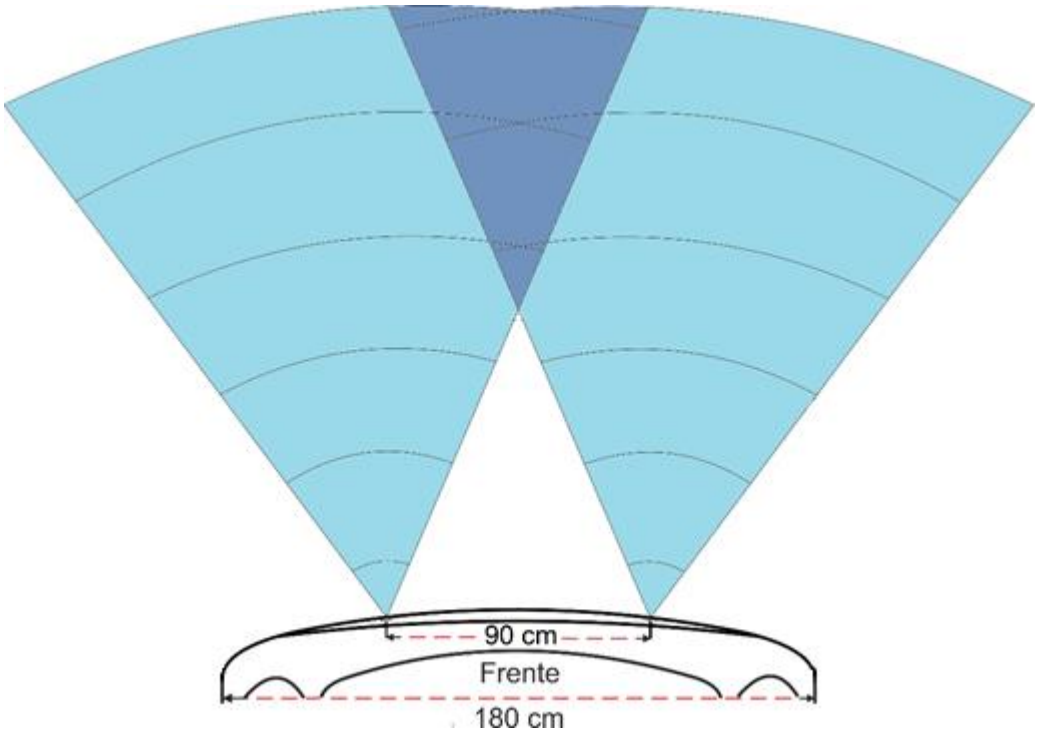


Fig. 4.18 Diagrama de detección de los sensores en la parte delantera del vehículo
Fuente: Investigador

- **Ubicación de los sensores y de la cámara en la parte trasera del vehículo**

Se realiza el mismo proceso que la parte delantera utilizando la fórmula (2), solo que este caso el número de sensores es *4* y se obtuvo que *D=45cm*. Para obtener medidas cerradas se ubicaron los sensores entre sí a *40 cm,* además la cámara se ubicó en la parte central del vehículo,

gracias a esto los sensores del centro se pueden separarse *10 cm* entre ellos logrando así cubrir más zona de detección como se ve en la figura 4.19.

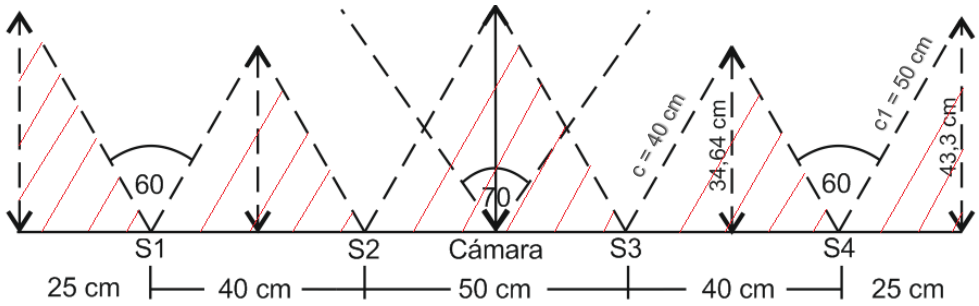


Fig. 4.19 Distancia de los sensores de la parte trasera del vehículo
Fuente: Investigador

Utilizando la fórmula (3) se obtuvo que *b = 34,64cm* y *b1 = 43,3cm*, *b* es la distancia de los sensores centrales hacia el objeto a detectar y *b1* es la distancia de los sensores laterales hacia el objeto a detectar.

De esta manera la ubicación correcta de los sensores y de la cámara en la parte trasera del vehículo se muestra en la figura 4.20.

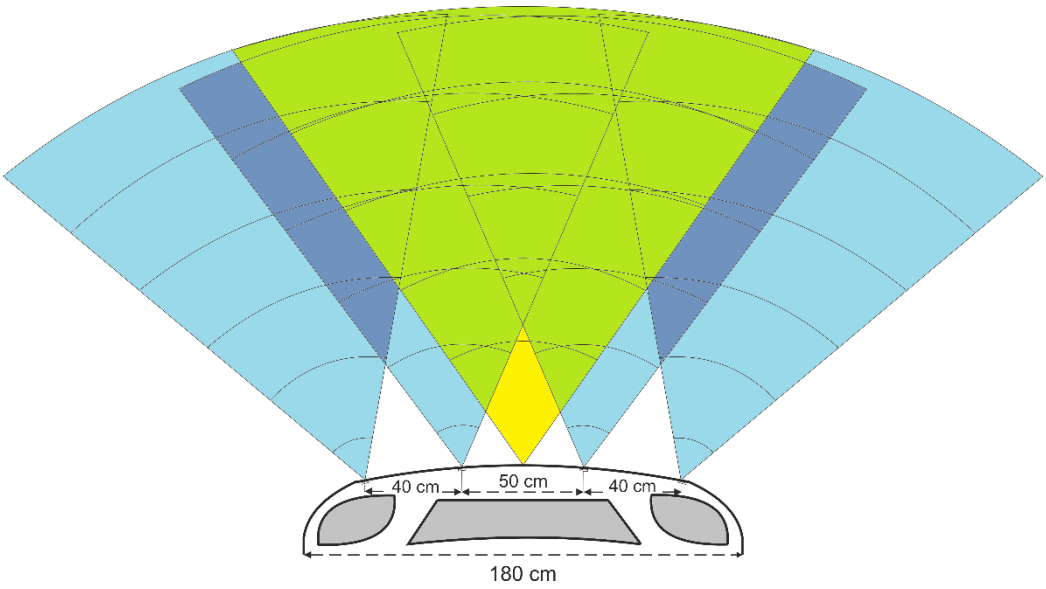


Fig. 4.20 Diagrama de detección de los sensores y de la cámara en la parte trasera del vehículo
Fuente: Investigador

Cabe mencionar que los sensores deben colocarse en línea horizontal y paralelo al suelo para evitar errores de medición.

La ubicación de los sensores desde el suelo hasta su posición está dentro de los (40 – 50)cm como se observa en la figura 4.21.

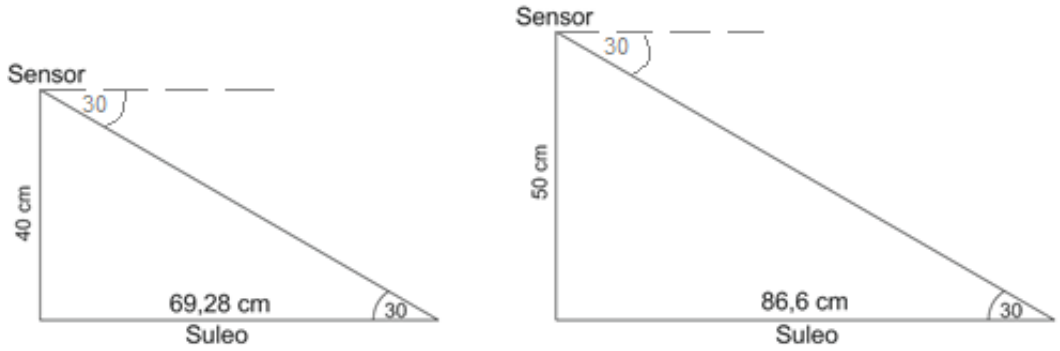


Fig. 4.21 Imagen de detección de los sensores en forma vertical
Fuente: Investigador

Para calcular la distancia (Suelo) se utilizó la fórmula (4).

$$tang(30) = \frac{40\ cm}{Suelo} \qquad \text{Fórmula (4)}$$

Donde las distancia del vehículo hasta un posible obstáculo a una altura de 40cm será 69,28cm, mientras que si desea colocar a 50 cm utilizando la fórmula (4) se obtiene 86,6cm.

Estas distancias son las más adecuadas para colocar los sensores debido a que si se ubica más abajo puede que los sensores detecte al suelo como un obstáculo y si se coloca más arriba no podrá detectar objetos que se encuentran cerca del suelo.

La visualización de la cámara y la presentación medida por los sensores se observará en un pantalla de 7 pulgadas. Además el sistema posee de un zumbador el cual sonará a diferente frecuencia dependiendo de la cercanía de un objeto.

4.17 DISEÑO DE LA INTERFACE

Antes de comenzar a diseñar la interface se carga la pantalla de 7 pulgadas para que la aplicación funcione con ella, para esto se debe conectar un cable en la mini USB de la UDOO y conectar en un puerto USB de una computadora.

Se procede a ejecutar *Putty* que es un programa que permite configurar remotamente la UDOO, se especifica el *COM* y la velocidad que este caso es 115200 se carga el terminal luego se alimenta a la placa UDOO y se introduce el siguiente código en el terminal.

setenv bootargs console=ttymxc1,115200 init=/init video=mxcfb0:dev=ldb,LDB-WVGA,if=RGB666,bpp=32 video=mxcfb1:off video=mxcfb2:off fbmem=28M vmalloc=400M androidboot.console=ttymxc1 androidboot.hardware=freescale mem=1024M

Por último se escribe *saveenv* se vuelve apagar y prender la UDOO y ahora se puede utilizar la pantalla sin ningún problema como se observa en la figura 4.22.

Fig. 4.22 Pantalla Táctil de 7'' para la placa UDOO
Fuente: http://www.udoo.org/ProjectsAndTutorials/how-to-connect-lvds-displays-to-udoo-with-ubuntu-and-android/?portfolioID=1394

Para comenzar con el diseño de la interface en Eclipse primero se dieron permisos necesarios para utilizar la cámara y el accesorio USB.

<uses-feature android:name="android.hardware.usb.accessory" />

<uses-feature android:name="android.hardware.camera" />

<uses-permission android:name="android.permission.CAMERA" />

Después se introdujo un archivo *xml* en la carpeta *res* en la cual se describe algunos parámetros para que Arduino pueda comunicarse con Android.

<usb-accessory manufacturer="UNIVERSIDAD_TÉCNICA_DE_AMBATO" model="ANDROID_ARDUINO" version="1.0" />

En la ventana principal se colocó toda la programación para trasferir datos desde Arduino a Android y se creó un *fragment* en donde se activa la visualización de la cámara.

- Flujo de entrada y salida de datos de Android

```
private void openAccessory(UsbAccessory accessory) {
mFileDescriptor = mUsbManager.openAccessory(accessory);
if (mFileDescriptor != null) {
mAccessory = accessory;
FileDescriptor fd = mFileDescriptor.getFileDescriptor();
                mInputStream = new FileInputStream(fd);
                mOutputStream = new FileOutputStream(fd);
Thread thread = new Thread(null, this, "UDOO_ADK_readfrom");
                        thread.start();

Toast.makeText(getApplicationContext(), "Conectado",
                                        Toast.LENGTH_SHORT).show();
                Log.i(TAG, "openaccessory");
        } else {
Toast.makeText(getApplicationContext(), "Desconectado",
Toast.LENGTH_SHORT).show();
        }

}
```

- Código para que Android puede leer los datos que proporciona Arduino

```
public void run() {
int ret = 0;
byte[] buf = new byte[8];
running = true;

while (running) {
        try {
                ret = mInputStream.read(buf);

        } catch (IOException e) {
                break;
        }
        m = Message.obtain(mHandler);

        if (ret != 0) {
                m.arg1 = unsignedByteToInt(buf[1]);

                ret = 0;
        }
        mHandler.sendMessage(m);

        }
}

private int unsignedByteToInt(byte b) {
        return b & 0xFF;
}

static Handler mHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
                izquierda.setText(msg.arg1 + " cm");
        }

};
```

- *Fragment* para la visualización previa de la cámara

Dentro del *fragment* se creó la visualización de la cámara con un *Surfaceview*.

```
//Crea la previsualización de la cámara
@Override
public void surfaceCreated(SurfaceHolder holder) {
    this.surfaceHolder = holder;

      startCameraPreview();
  }

//Cambia la previsualización de la cámara
@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width,
int height) {

  }

//Destruye la previsualización de la cámara
@Override
public void surfaceDestroyed(SurfaceHolder holder) {

  }
```

Estos son los pasos más importantes para la creación de la interface además en la carpeta *res – values* se coloca el nombre de la aplicación y el texto que aparecerá en la interface. En la figura 4.23 se muestra la interface terminada. Para observar todo el código de programación ir al ANEXO 1 y ANEXO 2.

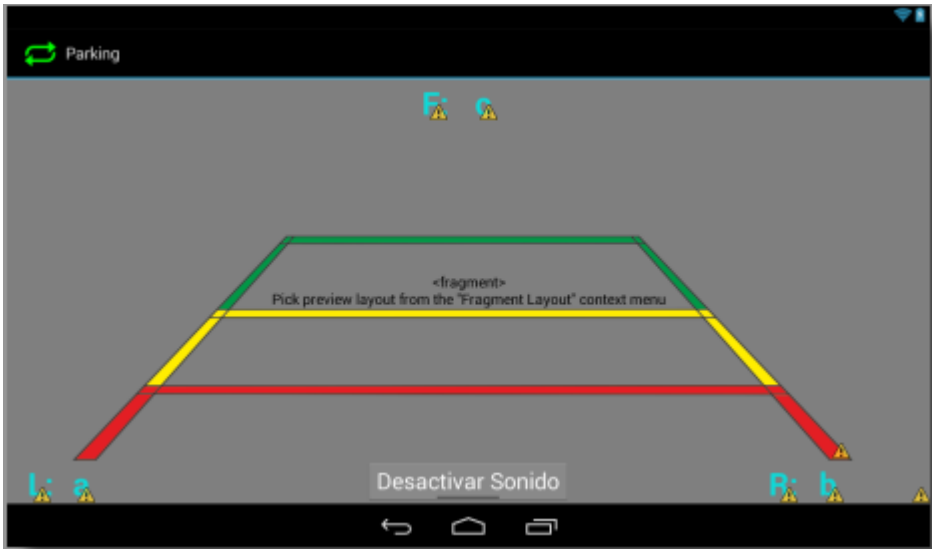


Fig. 4.23 Interface terminada
Fuente: Investigador

Adicionalmente se colocó una imagen que ayudará al conductor a tener una mejor perspectiva de la distancia del objeto al vehículo.

4.18 DISEÑO DEL CIRCUITO ELECTRÓNICO

A continuación en la figura 4.24 se presenta el circuito de conexión del sistema de estacionamiento propuesto.

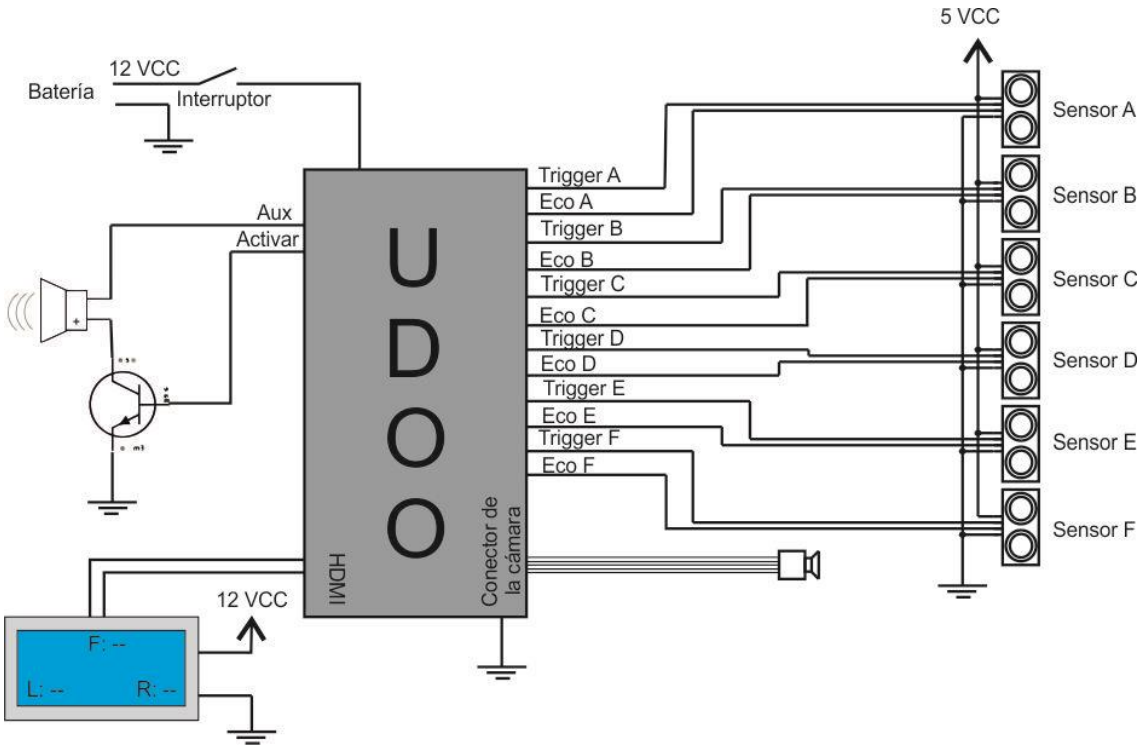


Fig. 4.24 Circuito electrónico del sistema de estacionamiento
Fuente: Investigador

El sistema de estacionamiento entra en funcionamiento cuando el interruptor es presionado, se espera un momento hasta que arranque el sistema operativo, luego automáticamente la aplicación se inicia y se comienza a visualizar la parte trasera del vehículo en la pantalla por medio de la cámara, el zumbador empieza a sonar y los sensores comienzan a detectar si existe algún objeto dentro del alcance de medición y proporcionan la distancia medida del objeto al conductor en la pantalla. El sistema no necesita de baterías adicionales, su alimentación es proporcionada por la misma batería del auto.

4.19 INSTALACIÓN DEL SISTEMA

4.19.1 Sensores y cámara

La ubicación de los sensores A, B, C, D, E, F y de la cámara como se observa en la figura 4.25 y 4.26 deben colocarse a las mismas alturas para poder evitar algún fallo en la medición.

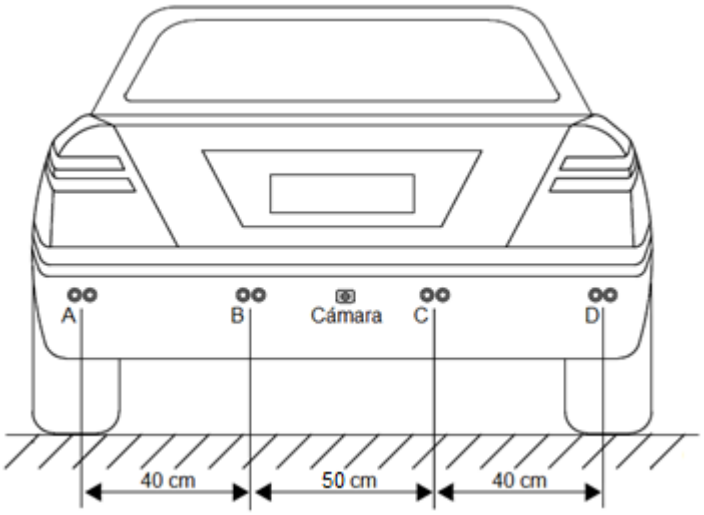


Fig. 4.25 Ubicación correcta de los sensores en la parte trasera del vehículo
Fuente: http://parking-sensors.hyt.com/doc/HYT-F4R4-BEEPER.pdf

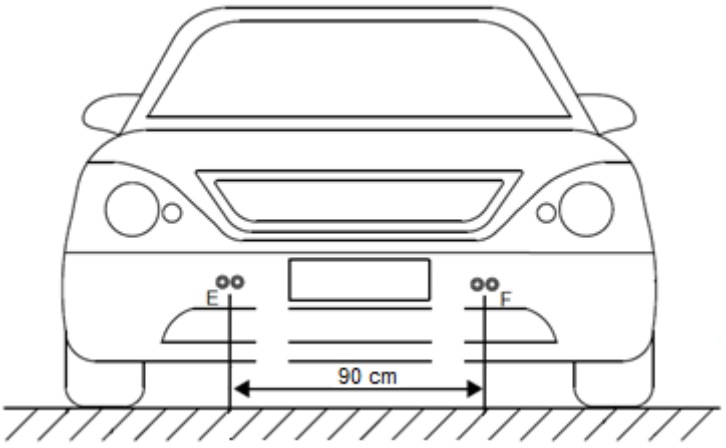


Fig. 4.26 Ubicación correcta de los sensores en la parte delantera del vehículo
Fuente: http://parking-sensors.hyt.com/doc/HYT-F4R4-BEEPER.pdf

La altura va a depender del diseño del vehículo, se recomienda colocar entre 40 cm y 50 cm por el ángulo de detección de los sensores como se observa en la figura 4.27.

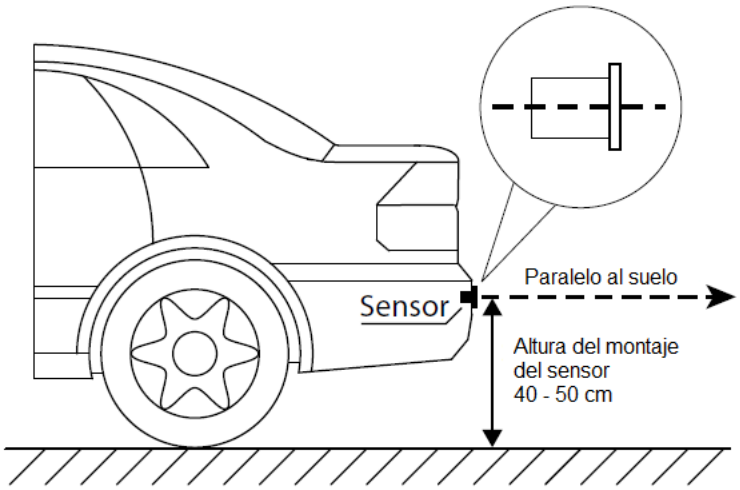


Fig. 4.27 Altura adecuada para la instalación de los sensores
Fuente: http://parking-sensors.hyt.com/doc/HYT-F4R4-BEEPER.pdf

4.19.2 Pantalla

La ubicación de la pantalla de 7 pulgadas va a depender del conductor y del vehículo como se muestra en la figura 4.28.

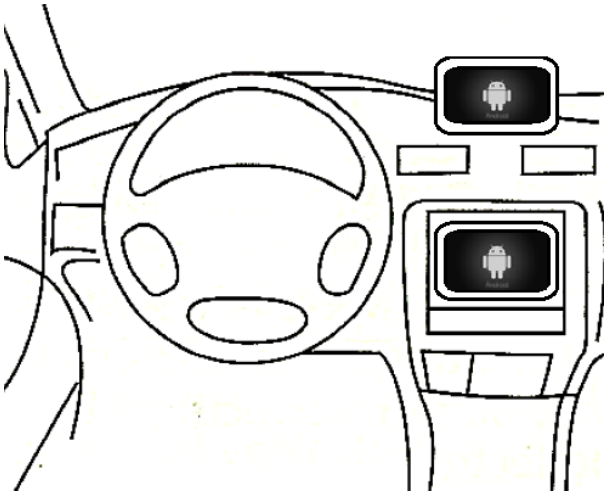


Fig. 4.28 Ubicación correcta de la pantalla táctil
Fuente: http://alarmasgenius.com/documentos/Manuales Genius Web/Sensores Genius de Retroceso4.pdf

4.19.3 Zumbador

El zumbador se coloca en la parte central del parabrisas posterior como observa en la figura 4.29.

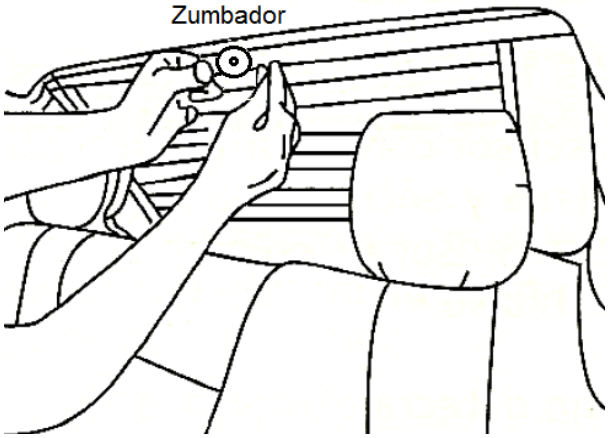


Fig. 4.29 Ubicación del zumbador
Fuente: http://alarmasgenius.com/documentos/Manuales Genius Web/Sensores Genius de Retroceso4.pdf

4.19.4 Placa UDOO

Encontrar un espacio adecuado para ubicarlo fijamente la placa UDOO dentro del área del baúl como se observa en la figura 4.30.

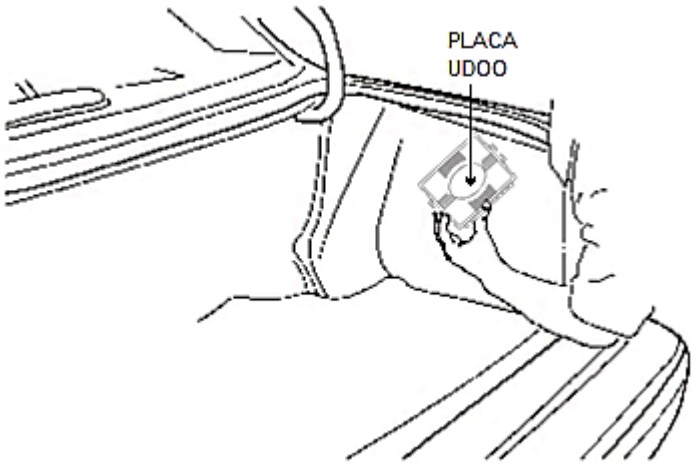


Fig. 4.30 Ubicación de la placa UDOO
Fuente: http://www.b52audio.com/esp/recursos/uploads/productos/PS-100_esp.pdf

4.20 COSTO DEL PROYECTO

El costo de los materiales del sistema de estacionamiento en un vehículo liviano se detalla en la tabla 4.5.

Tabla 4.5 Costo de los materiales para el proyecto

Descripción	Cantidad	Valor Unit.	Valor Total
Placa UDOO Quad	1	$ 200	$ 200
Cámara de UDOO	1	$ 70	$ 70
Pantalla de 7 pulgadas	1	$ 160	$ 160
Sensor ultrasónico Hc-Sr04	6	$ 10	$ 60
Zumbador	1	$ 0,30	$ 0,30
Transistor 3904	1	$ 0,35	$ 0,35
Fusible	1	$ 0,30	$ 0,30
Interruptor	1	$ 0,40	$ 0,40
Cable para los sensores y el zumbador (4 metros)	7	$ 0,80	$ 5,60
Cable para la pantalla (4 metros)	1	$ 10	$ 10
Conectores	9	$ 0,30	$ 2,70
		TOTAL	$ 509,65

Fuente: Investigador

El costo de instalación y de la mano de obra del sistema de estacionamiento se obtuvo consultando a las personas que instalan este tipo de sistemas y el tiempo que se demora en hacerlo que es aproximadamente un día. De lo cual se obtuvo que es 10% del costo del sistema. Además se requiere de algunos materiales adicionales como pegamento, tornillos, taladro entre otras cosas para sujetar los dispositivos en el vehículo que sería de unos $5 adicionales. Cabe mencionar que se ha empleado software libre el cual no tiene ningún costo al momento de utilizarlo. Con lo cual el costo total del sistema de estacionamiento instalado en un automóvil es: $ 565,62.

4.21 IMPLEMETACIÓN DEL SISTEMA DE ESTACIONAMIENTO EN UNA CAMIONETA MAZDA DOBLE CABINA

En la figura 4.31 y 4.32 se muestran como quedaron instalados los sensores y la cámara en la parte trasera y delantera del vehículo respectivamente.

Fig. 4.31 Instalación de los sensores y de la cámara en la parte trasera del vehículo
Fuente: Investigador

Fig. 4.32 Instalación de los sensores en la parte delantera del vehículo
Fuente: Investigador

En la figura 4.33 se observa la instalación de la pantalla y del interruptor del encendido del sistema de estacionamiento.

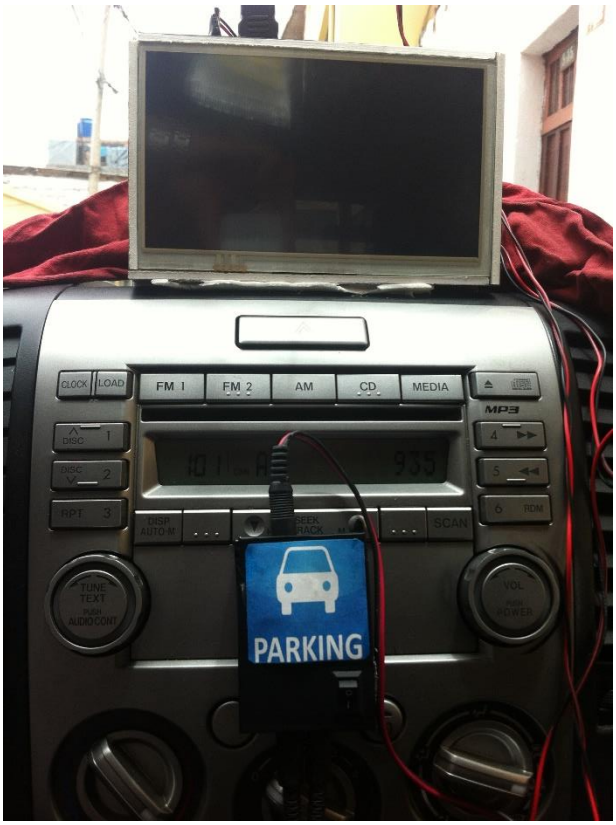


Fig. 4.33 Instalación de la pantalla y del interruptor para el encendido del sistema de estacionamiento
Fuente: Investigador

Por último en la figura 4.34 se indica la ubicación del zumbador.

Fig. 4.34 Instalación del zumbador
Fuente: Investigador

4.22 PRUEBAS DE FUNCIONAMIENTO DEL SISTEMA DE ESTACIONAMIENTO

De la figura 4.35 hasta la figura 4.37 se indica el funcionamiento de la cámara y de los sensores ubicados en la parte trasera del vehículo

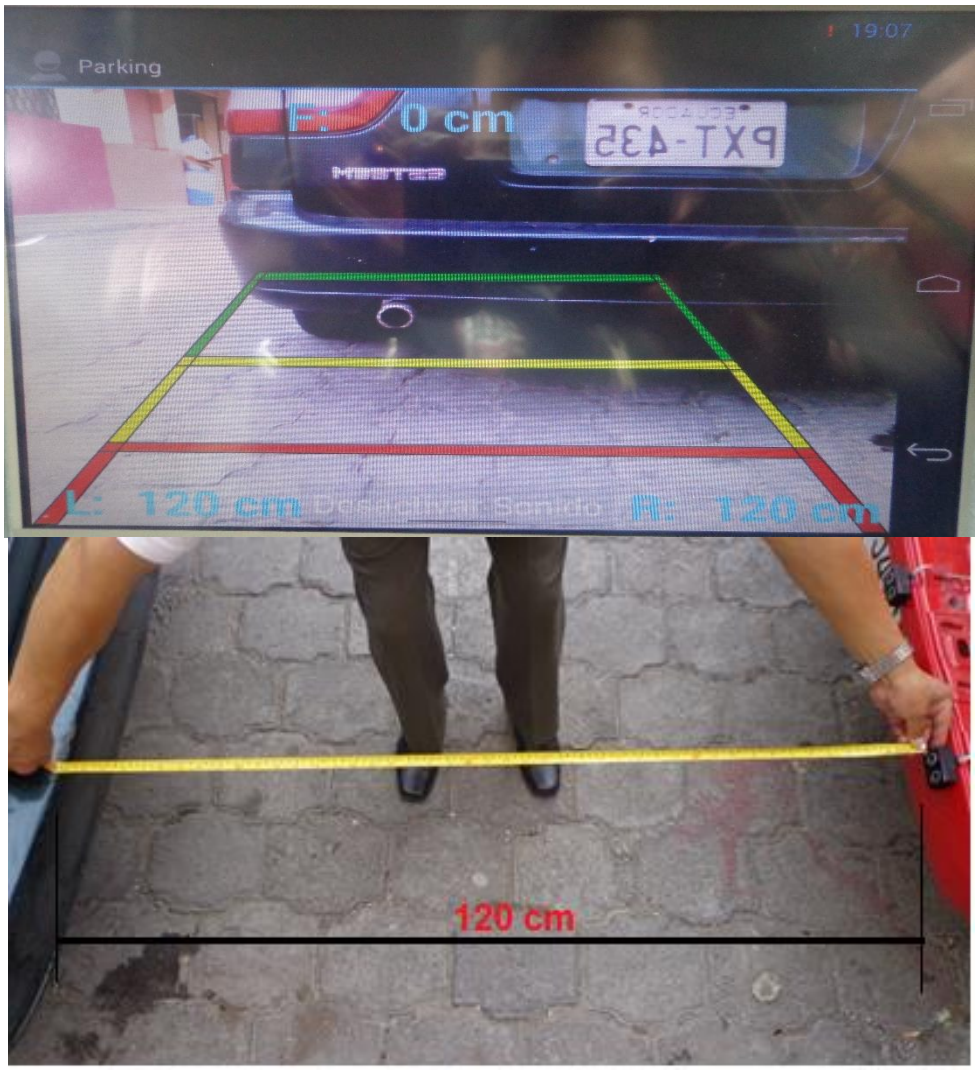


Fig. 4.35 Vehículo a 120 cm de distancia de otro en la parte trasera
Fuente: Investigador

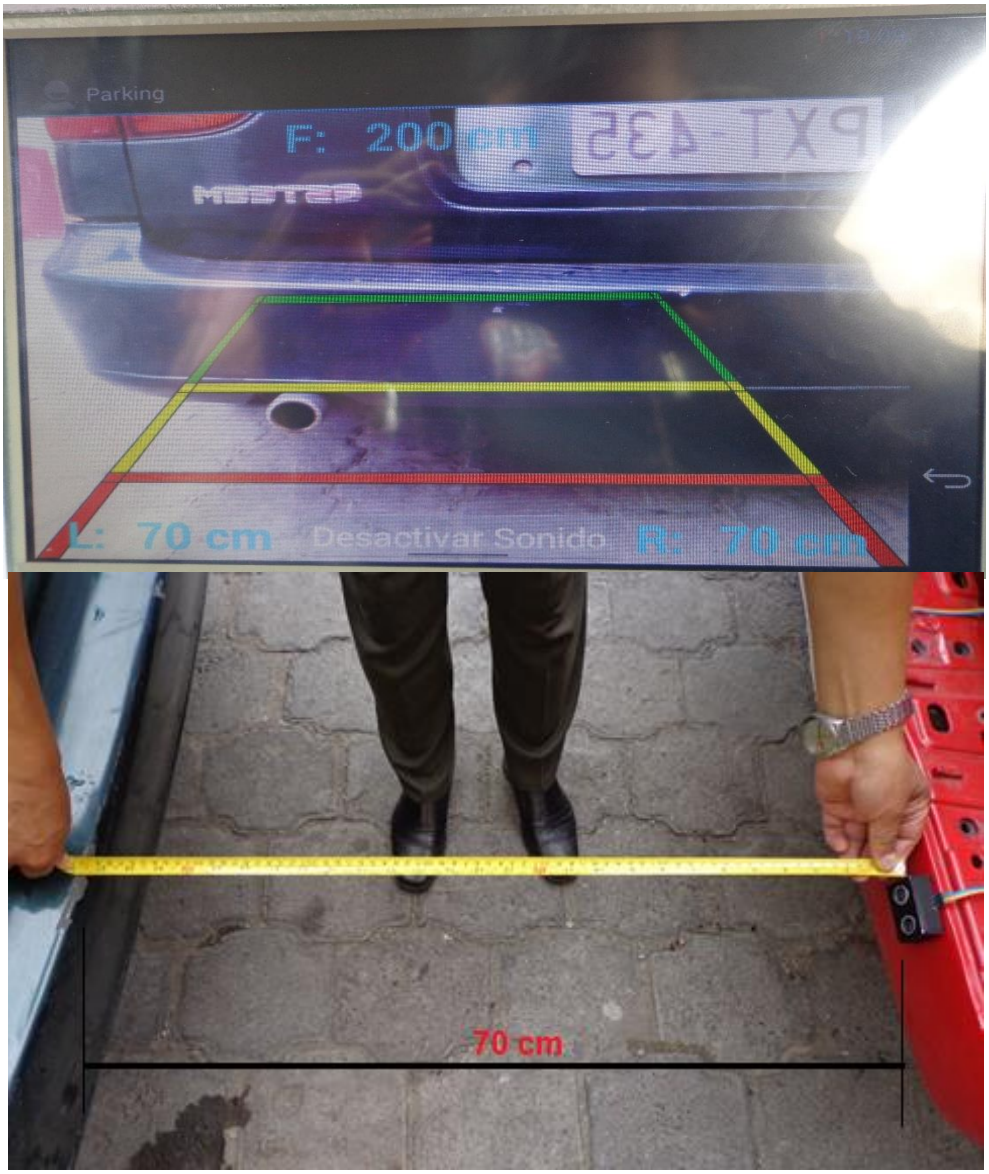


Fig. 4.36 Vehículo a 70 cm de distancia de otro en la parte trasera
Fuente: Investigador

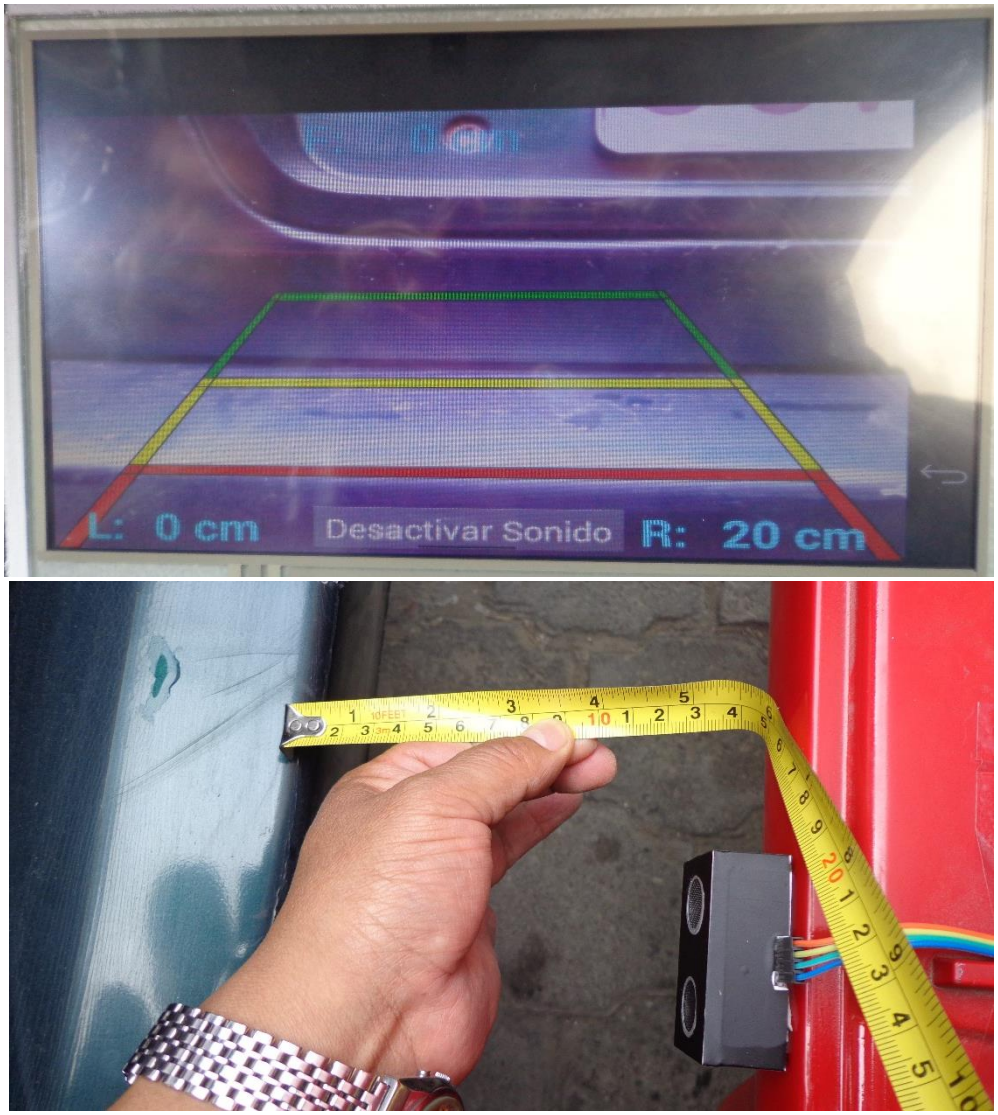


Fig. 4.37 Vehículo a menos 20 cm de distancia de otro en la parte trasera
Fuente: Investigador

En la figura 4.38 y 4.39 muestran el funcionamiento del sistema de los sensores ubicados en la parte delantera del vehículo

Fig. 4.38 Vehículo a 90 cm de distancia de otro en la parte delantera
Fuente: Investigador

Fig. 4.39 Vehículo a 30 cm de distancia de otro en la parte delantera
Fuente: Investigador

CAPÍTULO V

CONCLUISONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Al analizar los diferentes tipos de sistemas de estacionamiento actualmente instalados en los automóviles, se determinó que el uso de 6 sensores ultrasónicos, el empleo de una cámara y una pantalla son suficientes para cubrir el área de estacionamiento del vehículo.

- Los cuatro sensores en la parte trasera del vehículo tienen una separación de 40 cm entre ellos para obtener un área de detección equitativa, con el mismo propósito se ubicó dos sensores en la parte frontal del automóvil los cuales se encuentran separados a una distancia de 90 cm.

- Se determinó la fórmula para ubicar correctamente los sensores ya sea para la parte trasera como delantera del vehículo la cual es: dividir el ancho del vehículo por el número de sensores a utilizar.

- El diseño de la interface consta de dos *layout* para que no exista conflicto entre los procesos utilizados como son lectura y escritura de datos y la previsualización de la cámara.

5.2 RECOMENDACIONES

- Instalar todos los dispositivos del sistema de estacionamiento propuesto debido a que si falta algún elemento del mismo podría funcionar de forma errónea.

- Ubicar los sensores uno a continuación de otro a una distancia igual, si se coloca a diferentes distancias existirán áreas más grandes donde los sensores no podrán detectar ningún objeto.

- Es necesario mencionar que la fórmula obtenida para colocar correctamente los sensores sirve tanto para vehículos livianos como para vehículos pesados.

- Al utilizar dos layout en la interfaz recordar que el layout de la cámara debe estar ubicado en la parte posterior de la interfaz puesto que si se ubica adelante no se podrá observar los datos enviados por los sensores.

BIBLIOGRAFÍA

[1] "Breve historia del automóvil" [online], (2012), Disponible en: http://www.slideshare.net/pulga26/breve-historia-del-automvil#btnNext

[2] "Agencia Nacional de Tránsito" [online], (2014), Disponible en: http://www.ant.gob.ec/index.php/noticias/estadisticas#.UvKMXrRWS7V

[3] J. W. Bonilla Nieto, "Diseño y adaptación de un sistema de seguridad activo para estacionamiento vehicular y monitoreo continuo", Universidad de las Fuerzas Armadas-ESPE, 2012.

[4] P. Suárez, E. Carlos, "Desarrollo de un sistema prototipo de entrenamiento automatizado para estacionar vehículos con interfaz gráfica, auditiva y modelo escala de automóvil, para usuarios en proceso de aprendizaje de conducción", Universidad Nueva Esparta, 2011.

[5] S. Mayanza Lema, "Diseño e implementación de un sistema ultrasónico de ayuda para parqueo de vehículos automotrices de un banco de prueba para la escuela de ingeniería automotriz", Escuela Superior Politécnica de Chimborazo, 2012.

[6] F. González, W. Chacón, "Diseño e implementación de un sistema de asistencia para estacionamiento en un vehículo Toyota Corolla, Universidad Técnica Salesiana, 2009.

[7] O. Matza, "Sistema electrónico de control de velocidad de autobuses, para la Cooperativa de transportes Santa", Universidad Técnica de Ambato, 2014.

[8] H. Zorrilla, "Mercedes crea un sistema de radar para la Clase CL" [online], (2006), Disponible en: http://www.deautomoviles.com.ar/articulos/tecnologia/radar.html

[9] “Active Park Assist de Ford” [online], (2009), Disponible en: http://www.tuexperto.com/2009/01/09/activepark-assist-de-ford-un-nuevo-sistema-de-ayuda-al-aparcamiento/

[10] “Sistema VW (Park Assist Vision)” [online], (2008), Disponible en: http://es.autoblog.com/2008/04/24/vw-presentaun-nuevo-sistema-de-estacionamiento-que-no-necesita/

[11] “Sistema Automático BMW” [online], (2012), Disponible en: http://www.eldiferencial.com.mx/index.php?option=com_content&task=view&id=1901&Itemid=88

[12] “Hyundai actualiza ligeramente al Elantra” [online], (2013), Disponible en: http://www.autoblog.com.uy/2013/08/pero-si-estas-igual-hyundai-actualiza.html

[13] “Tradición en la aceleración” [online], (2013), Disponible en: http://es.chevrolet.com/ss-sports-sedan.html

[14] Villareal D., “Nissan Qashqai 2014. Un vistazo a su tecnología y el Escudo de Protección Inteligente” [online], (2014), Disponible en: http://www.tecmovia.com/2014/01/15/nissan-qashqai-2014-un-vistazo-a-su-tecnologia-y-el-escudo-de-proteccion-inteligente/

[15] “KIA CEE'D SPORTSWAGON” [online], (2013), Disponible en: http://www.kia.com/es/campaigns-and-redirects/gama-ceed/

[16] “Sistema electrónico” [online], (2014), Disponible en: http://www.maquinariapro.com/sistemas/sistema-electronico.html

[17] E. Mieles, “Software Libre”, [Online], (2004), Disponible en: http://www.iti.es/media/about/docs/tic/04/2004-06-software.pdf

[18] O. Gutiérrez, “Android lidera en los mercados del mundo: estudio” [online], (2014), Disponible en: http://www.cnet.com/es/noticias/android-ios-windowsphone-participacion-mercado/

[19] D. Sanz, M. Saucedo, P. Torralbo, “Introducción a Android” [online], (2012), Universidad Complutense de Madrid, E.M.E. Editorial, Disponible en: http://pendientedemigracion.ucm.es/info/tecnomovil/documentos/android.pdf

[20] J. González, J. Pascual, G. Robles, “Introducción al software libre” [online], (2007), Disponible en: http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/eclipse.html

[21] “Hardware Libre” [online], (2014), Disponible en: http://es.wikipedia.org/wiki/Hardware_libre

[22] B. Úbeda, “Sistemas Embebidos” [online], (2009), Disponible en: http://ocw.um.es/ingenierias/sistemas-embebidos/material-de-clase-1/ssee-t01.pdf

[23] I. Villegas, “Sistemas Embebidos” [online], (2010) Disponible en: ielectronicaunprg.files.wordpress.com/2008/03/sistemas-embebidos1.ppt

[24] E. Machuca, “Raspberry Pi y sus Aplicaciones” [online], (2014), Universidad Católica Nuestra Señora de la Asunción, Paraguay, Disponible en: http://jeuazarru.com/wp-content/uploads/2014/10/raspberrypi.pdf

[25] “Raspberry Pi” [online], (2011), Disponible en: http://www.raspberryshop.es/

[26] “Guía básica de Arduino” [online], (2012), Disponible en: http://tienda.tdrobotica.co/download/Libro_kit_Basico.pdf

[27] “UDOO” [online], (2013), Disponible en: http://udoo.org/download/files/Documents/UDOO_Starting_Manual_beta0.4_11_28_2013.pdf

[28] “56 mil vehículos fueron matriculados en 2012” [online], (2013), Disponible en: http://www.lahora.com.ec/index.php/noticias/show/1101449285/-1/56_mil_veh%C3%ADculos_fueron_matriculados_en_2012.html#.VH-iM8mf5I4

[29] “Lubuntu” [online], (2014), Disponible en: http://lubuntu.net/

[30] Zamalloa W., “¿Porque debería desarrollar en Android?” [online], (2012), Disponible en: http://es.slideshare.net/vlaslo/porque-android

[31] Henao C., “¿Eclipse o Netbeans?“ [online], (2013), Disponible en: http://codejavu.blogspot.com/2013/10/eclipse-o-netbeans.html

[32] “Sensor de Distancia Ultrasónico HC-SR04” [online], Disponible en: http://www.techmake.com/sen-00029.html

[33] “MaxSonar EZ1 sensor ultrasónico” [online], Disponible en: http://www.superrobotica.com/download/S320120/Manula%20S320120%20MaxSonar%20EZ1.pdf

[34] “Sensor ultrasónico TP-US03” [online], Disponible en: http://www.timberport.com/product.php?id=124

[35] “Creación de tarjetas Micro SD de arranque de una imagen” [online], (2014), Disponible en: http://www.udoo.org/getting-started/creating-a-bootable-micro-sd-card-from-image/

[36] “Obtener el Arduino-IDE listo para programar UDOO” [online], (2014), Disponible en: http://www.udoo.org/ProjectsAndTutorials/get-the-arduino-ide-ready-to-program-udoo/?portfolioID=1394

[37] “Rango ultrasónico del módulo del sensor HC-SR04” [online], Disponible en: http://www.famosastudio.com/ultrasonic-range-sensor-hc-sr04

ANÉXOS

ANEXO 1

- **CÓDIGO DE LA APLICACIÓN EN ECLIPSE**

CamaraActivity.java

```
//paquete (carpeta) donde se encuentra todo el programa
package com.androidzeitgeist.mustache.activity;
//utiliza para "importar" o hacer referencia al contenido de los paquetes en java
//Proporciona al sistema un flujo de datos de entradas y salidas
import java.io.FileDescriptor;
//Identificador de flujo de entrada
import java.io.FileInputStream;
//Identificador de flujo de salida
import java.io.FileOutputStream;
//Crea automáticamente
import java.io.IOException;
//Interactua con el usuario
import android.app.Activity;
//Sirve para mantener los datos
import android.app.PendingIntent;
//Recibe informacion del Broadcast
import android.content.BroadcastReceiver;
//Permite el acceso a los recursos y las clases específicas de la aplicación
import android.content.Context;
//lanzamiento de las actividades
import android.content.Intent;
//Empareja acciones, categorías y datos (AndroidManifest.xml)
import android.content.IntentFilter;
//Información que se puede recuperar sobre una actividad determinada
//aplicación o receptor
import android.content.pm.ActivityInfo;
//Representa un accesorio USB
```

import *android.hardware.usb.UsbAccessory;*

//Permite acceder al estado de USB y se comunica con los dispositivos USB

import *android.hardware.usb.UsbManager;*

//Una asignación de valores string a los diferentes tipos Parcelable.

import *android.os.Bundle;*

//Permite enviar y procesar mensajes

import *android.os.Handler;*

//Define un mensaje que contiene una descripción y un objeto de datos arbitrarios que pueden ser enviados a un Handler

import *android.os.Message;*

//FileDescriptor retorna por readFileDescriptor(), le permite al uasuario cerrar cuando haya terminado

import *android.os.ParcelFileDescriptor;*

//API para el envío de la salida del registro

import *android.util.Log;*

//Representa el bloque de construcción básico para los componentes de la interfaz de usuario

import *android.view.View;*

//Muestra una imagen arbitraria, tal como un icono.

import *android.widget.ImageView;*

//Muestra el texto para el usuario y opcionalmente les permite editarlo

import *android.widget.TextView;*

//Es una vista que contiene un pequeño mensaje rápido para el usuario

import *android.widget.Toast;*

//Verifica el estado del boton con un indicador de "luz"

import *android.widget.ToggleButton;*

import *com.androidzeitgeist.mustache.R;*

//Un Fragment representa un comportamiento o una porción de interfaz de usuario en una Activity

import *com.androidzeitgeist.mustache.listener.CameraFragmentListener;*

```
//Actividad que muestra una vista previa de la cámara,
//Runnable declara una función denominada run, que han de definir las clases
que implementen esta interface
public class CameraActivity extends Activity implements
CameraFragmentListener,  Runnable{

    //Utiliza una imagen
    static ImageView iv1;
    //Utiliza ADK (Application Development Kit)
    private static final String TAG = "UDOO_AndroidADKFULL";
    //Acciona el permiso USB
    private static final String ACTION_USB_PERMISSION =
    "org.udoo.androidadkdemobidirect.action.USB_PERMISSION";
    private UsbManager mUsbManager;
    private PendingIntent mPermissionIntent;
    private boolean mPermissionRequestPending;

    //Variables para la utilización del accesorio USB
    UsbAccessory mAccessory;
    ParcelFileDescriptor mFileDescriptor;
    FileInputStream mInputStream;
    FileOutputStream mOutputStream;

    //Variables utilizadas en la interfaz del usuario
    private ToggleButton botonLED;
    private static TextView izquierda;
    private static TextView derecha;
    private static TextView frontal;
   private boolean running = false;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_camera);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDS
CAPE);
        iv1 = (ImageView) findViewById(R.id.parqueo);
        iv1.setVisibility(View.VISIBLE);
        mUsbManager = (UsbManager)
        getSystemService(Context.USB_SERVICE);
        mPermissionIntent = PendingIntent.getBroadcast(this, 0, new
        Intent(ACTION_USB_PERMISSION), 0);
        IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
        filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED
        );
        registerReceiver(mUsbReceiver, filter);
        if (getLastNonConfigurationInstance() != null) {
        mAccessory = (UsbAccessory) getLastNonConfigurationInstance();
                        openAccessory(mAccessory);
                }

        //Utilización de las variables
        botonLED = (ToggleButton) findViewById(R.id.toggleButtonLED);
        izquierda = (TextView) findViewById(R.id.textView_izquierda);
        derecha = (TextView) findViewById(R.id.textView_derecha);
        frontal = (TextView) findViewById(R.id.textView_frontal);
}
//Revisa e intenta conectarse con el accesorio USB conectada
private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
```

```
        if (ACTION_USB_PERMISSION.equals(action)) {
        synchronized (this) {
        UsbAccessory accessory = (UsbAccessory) intent
        .getParcelableExtra(UsbManager.EXTRA_ACCESSORY);
                if
                (intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GR
                ANTED,   false)){
                openAccessory(accessory);
                } else {
        Log.d(TAG, "permiso denegado para el accesorio" + " " + accessory);
                }
        mPermissionRequestPending = false;
                }
        } else if
(UsbManager.ACTION_USB_ACCESSORY_DETACHED.equals(action)) {
                UsbAccessory accessory = (UsbAccessory) intent
                .getParcelableExtra(UsbManager.EXTRA_ACCESSORY);
                if (accessory != null && accessory.equals(mAccessory)) {
                closeAccessory();
                        }
                }
        }
};
@Override
public Object onRetainNonConfigurationInstance() {
        if (mAccessory != null) {
                return mAccessory;
                } else {
                return super.onRetainNonConfigurationInstance();
                }
}
```

```
//Abre el accesorio USB de la lista de accesorios
@Override
public void onResume() {
        super.onResume();
        if (mInputStream != null && mOutputStream != null) {
                return;
                }
        UsbAccessory[] accessories = mUsbManager.getAccessoryList();
        UsbAccessory accessory = (accessories == null ? null :accessories[0]);
        if (accessory != null) {
                if (mUsbManager.hasPermission(accessory)) {
                openAccessory(accessory);
                } else {
                synchronized (mUsbReceiver) {
                        if (!mPermissionRequestPending) {
        mUsbManager.requestPermission(accessory,mPermissionIntent);
        mPermissionRequestPending = true;
                                }
                        }
                }
        } else {
                Log.d(TAG, "Accesorio nulo");
        }
}
@Override
public void onPause() {
        super.onPause();
}
@Override
public void onDestroy() {
        closeAccessory();
        unregisterReceiver(mUsbReceiver);
```

```
        super.onDestroy();
}
//Abre el flujo de entrada y de salida de datos desde el descriptor, también
inicia el hilo conductor que se lee en Arduino
private void openAccessory(UsbAccessory accessory) {
        mFileDescriptor = mUsbManager.openAccessory(accessory);
        if (mFileDescriptor != null) {
                mAccessory = accessory;
                FileDescriptor fd = mFileDescriptor.getFileDescriptor();
                mInputStream = new FileInputStream(fd);
                mOutputStream = new FileOutputStream(fd);
                Thread thread = new Thread(null, this, "UDOO_ADK_readfrom");
                thread.start();
                Toast.makeText(getApplicationContext(), "Conectado",
                Toast.LENGTH_SHORT).show();
                Log.i(TAG, "openaccessory");
                } else {
                Toast.makeText(getApplicationContext(), "Desconectado",
                        Toast.LENGTH_SHORT).show();
        }
}
//Cierra el accesorio USB
private void closeAccessory() {
        Log.i(TAG, "closeaccessory");
        try {
                if (mFileDescriptor != null) {
                        mFileDescriptor.close();
                        }
                } catch (IOException e) {
                } finally {
                mFileDescriptor = null;
                mAccessory = null;
```

```
                running = false;
        }
}
// ToggleButton enviar un mensage a SAM3X (Arduino)
public void blinkLED(View v) {
        if (mAccessory != null) {
                byte[] message = new byte[1];
                if (botonLED.isChecked()) {
                        message[0] = (byte) 1;
                Toast.makeText(getApplicationContext(), "Sonido Desactivado",
                Toast.LENGTH_SHORT).show();
                } else {
                        message[0] = (byte) 0;
                Toast.makeText(getApplicationContext(), "Sonido Activado",
                Toast.LENGTH_SHORT).show();
                        }
                        if (mOutputStream != null) {
                                try {
                                        mOutputStream.write(message);
                                } catch (IOException e) {
                                        Log.e(TAG, "escritura fallida", e);
                                }
                        }
        } else {
                Toast.makeText(getApplicationContext(), "Desconectado",
                Toast.LENGTH_SHORT).show();
        }
}
private static Message m;
private static Message m1;
private static Message m2;
```

```
//Hilo para los datos de lectura de SAM3X (Arduino)
public void run() {
        int ret = 0;
        byte[] buf = new byte[8];
        running = true;
        while (running) {
                try {
                        ret = mInputStream.read(buf);
                        } catch (IOException e) {
                        break;
                }
                m = Message.obtain(mHandler);
                m1 = Message.obtain(m1Handler);
                m2 = Message.obtain(m2Handler);
                if (ret != 0) {
                        m.arg1 = unsignedByteToInt(buf[1]);
                        m1.arg2 = unsignedByteToInt(buf[2]);
                        m2.obj= unsignedByteToInt(buf[3]);
                        ret = 0;
                }
                mHandler.sendMessage(m);
                m1Handler.sendMessage(m1);
                m2Handler.sendMessage(m2);

        }
}
private int unsignedByteToInt(byte b) {
        return b & 0xFF;
}
static Handler mHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
```

```
                izquierda.setText(msg.arg1 + " cm");
        }
};
static Handler m1Handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
                derecha.setText(msg.arg2 + " cm");
                        }
                };
static Handler m2Handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
                frontal.setText(msg.obj + " cm");
        }
};
// Fragment notifica acerca de un problema no recuperable con la cámara.
@Override
public void onCameraError() {
        Toast.makeText(this, getString(R.string.toast_error_camera_preview),
                Toast.LENGTH_SHORT).show();
        finish();
        }
}
```

CameraFragment.java

```
package com.androidzeitgeist.mustache.fragment;
import java.io.IOException;
import java.util.List;
import android.app.Activity;
import android.app.Fragment;
import android.content.pm.ActivityInfo;
import android.hardware.Camera;
```

```
import android.hardware.Camera.CameraInfo;
import android.hardware.Camera.Size;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.View;
import android.view.ViewGroup;
import com.androidzeitgeist.mustache.listener.CameraFragmentListener;
import com.androidzeitgeist.mustache.view.CameraPreview;
// Fragment para la visualización previa de la cámara
//SurfaceHolder.Callback implementa una interfaz para recibir información
acerca de los cambios de la superficie
public class CameraFragment extends Fragment implements
SurfaceHolder.Callback {
public static final String TAG = "Mustache/CameraFragment";
private static final int PICTURE_SIZE_MAX_WIDTH = 1280;
private static final int PREVIEW_SIZE_MAX_WIDTH = 640;
    private int cameraId;
    private Camera camera;
    private SurfaceHolder surfaceHolder;
    private CameraFragmentListener listener;
@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    if (!(activity instanceof CameraFragmentListener)) {
      throw new IllegalArgumentException(
      "Activity has to implement CameraFragmentListener interface"
       );
    }
     listener = (CameraFragmentListener) activity;
```

```
}
//Creación de un view del fragment
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    CameraPreview previewView = new CameraPreview(getActivity());
 etRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
    previewView.getHolder().addCallback(this);
       return previewView;
   }
   private void setRequestedOrientation(int screenOrientationLandscape) {
                // TODO Auto-generated method stub
        }
// Fragment siendo reanudado
@Override
public void onResume() {
    super.onResume();
    try {
       camera = Camera.open(cameraId);
    } catch (Exception exception) {
  Log.e(TAG, "No se puede adrir la camara con el ID " + cameraId, exception);
  listener.onCameraError();
  return;
    }
}
//Fragmnet siendo pausado
@Override
public void onPause() {
    super.onPause();
    stopCameraPreview();
    camera.release();
}
```

```
//Comenzando la vista previa de la cámara
private synchronized void startCameraPreview() {
    determineDisplayOrientation();
    setupCamera();
    try {
        camera.setPreviewDisplay(surfaceHolder);
        camera.startPreview();
    } catch (IOException exception) {
      Log.e(TAG, "Can't start camera preview due to IOException", exception);
        listener.onCameraError();
    }
}
//Parando la vista previa de la cámara
private synchronized void stopCameraPreview() {
    try {
        camera.stopPreview();
    } catch (Exception exception) {
        Log.i(TAG, "Exception during stopping camera preview");
    }
}
//Determinar la orientación de la pantalla actual y girar la vista previa de la
cámara en consecuencia
public void determineDisplayOrientation() {
    CameraInfo cameraInfo = new CameraInfo();
    Camera.getCameraInfo(cameraId, cameraInfo);
        int rotation =
getActivity().getWindowManager().getDefaultDisplay().getRotation();
        int degrees  = 0;
        switch (rotation) {
            case Surface.ROTATION_0:
                degrees = 0;
                break;
```

```
            case Surface.ROTATION_90:
                degrees = 90;
                break;
            case Surface.ROTATION_180:
                degrees = 180;
                break;
            case Surface.ROTATION_270:
                degrees = 270;
                break;
        }
        int displayOrientation;
        if (cameraInfo.facing == Camera.CameraInfo.CAMERA_FACING_FRONT)
{
            displayOrientation = (cameraInfo.orientation + degrees) % 360;
            displayOrientation = (360 - displayOrientation) % 360;
        } else {
         displayOrientation = (cameraInfo.orientation - degrees + 360) % 360;
        }
        camera.setDisplayOrientation(displayOrientation);
    }
    //Configuración de los parámetros de la cámara
    public void setupCamera() {
        Camera.Parameters parameters = camera.getParameters();
        Size bestPreviewSize = determineBestPreviewSize(parameters);
        Size bestPictureSize = determineBestPictureSize(parameters);
    parameters.setPreviewSize(bestPreviewSize.width, bestPreviewSize.height);
    parameters.setPictureSize(bestPictureSize.width, bestPictureSize.height);
        camera.setParameters(parameters);
    }
private Size determineBestPreviewSize(Camera.Parameters parameters) {
    List<Size> sizes = parameters.getSupportedPreviewSizes();
    return determineBestSize(sizes, PREVIEW_SIZE_MAX_WIDTH);
```

```
}
private Size determineBestPictureSize(Camera.Parameters parameters) {
  List<Size> sizes = parameters.getSupportedPictureSizes();
    return determineBestSize(sizes, PICTURE_SIZE_MAX_WIDTH);
}
protected Size determineBestSize(List<Size> sizes, int widthThreshold) {
    Size bestSize = null;
    for (Size currentSize : sizes) {
          boolean isDesiredRatio = (currentSize.width / 4) == (currentSize.height /
          3);
          boolean isBetterSize = (bestSize == null || currentSize.width >
          bestSize.width);
      boolean isInBounds = currentSize.width <= PICTURE_SIZE_MAX_WIDTH;
         if (isDesiredRatio && isInBounds && isBetterSize) {
           bestSize = currentSize;
        }
    }
    if (bestSize == null) {
        listener.onCameraError();
        return sizes.get(0);
    }
   return bestSize;
}
//Cámara de pre visualización Surface creada
@Override
public void surfaceCreated(SurfaceHolder holder) {
    this.surfaceHolder = holder;
    startCameraPreview();
}
```

```
//Cámara de pre visualización Surface cambiada
@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width, int
height) {
}
//Superficie de la cámara de pre visualización destruido
@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    }
}
```

CameraFragmentListener.java

```
package com.androidzeitgeist.mustache.listener;
import com.androidzeitgeist.mustache.fragment.CameraFragment;
//Interfaz Listener que tiene que ser implementado por las actividades utilizando
instancias
public interface CameraFragmentListener {
//Un error en la cámara no recuperable ha sucedido
    public void onCameraError();
}
```

CameraPreview.java

```
package com.androidzeitgeist.mustache.view;
import android.content.Context;
import android.util.AttributeSet;
import android.view.SurfaceView;

//Muestra una vista previa de la cámara (cuadrada).
public class CameraPreview extends SurfaceView {
    private static final double ASPECT_RATIO = 3.0 / 4.0;
    public CameraPreview(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
```

```
    }
    public CameraPreview(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
    public CameraPreview(Context context) {
        super(context);
    }
// Medir la vista y su contenido para determinar el ancho y la altura medida
    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)
{
        int height = MeasureSpec.getSize(heightMeasureSpec);
        int width = MeasureSpec.getSize(widthMeasureSpec);
        setMeasuredDimension(width, height);
    }
}
```

activity_camera.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
     <fragment
         android:id="@+id/camera_fragment"
android:name="com.androidzeitgeist.mustache.fragment.CameraFragment"
         android:layout_width="1200dp"
         android:layout_height="match_parent"
         android:layout_alignParentLeft="true"
         android:layout_alignParentTop="true" />

     <ToggleButton
         android:id="@+id/toggleButtonLED"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:onClick="blinkLED"
    android:textColor="@color/BLANCO"
    android:textOff="Desactivar Sonido"
    android:textOn="Activar Sonido"
    android:textSize="25sp" />

<ImageView
    android:id="@+id/parqueo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:src="@drawable/parqueo" />

<TextView
    android:id="@+id/textView_distancia2_string"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_marginRight="146dp"
    android:onClick="camara"
    android:text="@string/distancia2"
    android:textColor="@color/CELESTE"
    android:textSize="35sp"
    android:textStyle="bold" />
```

```
<TextView
    android:id="@+id/textView_frontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_marginRight="101dp"
    android:layout_toLeftOf="@+id/textView_derecha"
    android:text="c "
    android:textColor="@color/CELESTE"
    android:textSize="35sp"
    android:textStyle="bold" />

<TextView
    android:id="@+id/textView_distancia_string"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_marginLeft="23dp"
    android:text="@string/distancia"
    android:textColor="@color/CELESTE"
    android:textSize="35sp"
    android:textStyle="bold" />

<TextView
    android:id="@+id/textView_izquierda"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginLeft="21dp"
    android:layout_toRightOf="@+id/textView_distancia_string"
    android:text="a "
```

```
        android:textColor="@color/CELESTE"
        android:textSize="35sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textView_derecha"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView_distancia2_string"
        android:layout_alignParentBottom="true"
        android:layout_marginLeft="56dp"
        android:text="b "
        android:textColor="@color/CELESTE"
        android:textSize="35sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textView_distancia3_string"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginRight="29dp"
        android:layout_toLeftOf="@+id/textView_frontal"
        android:text="@string/distancia3"
        android:textColor="@color/CELESTE"
        android:textSize="35sp"
        android:textStyle="bold" />

</RelativeLayout>
```

AndroidManifiest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.androidzeitgeist.mustache"
    android:versionCode="1"
    android:versionName="1.0" >
    <!-- Permite utilizar el accesorio USB -->
    <uses-feature android:name="android.hardware.usb.accessory" />
    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="20" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:allowClearUserData="false" >
        <activity
            android:name=".activity.CameraActivity"
            android:label="@string/app_name">
                    <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.LAUNCHER" />
                 </intent-filter>
<!-- Permite que la aplicación sea notificada de un accesorio USB conectado --
>
```

```
        <intent-filter>
<action
android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
/>
        </intent-filter>

<!-- Archivo de recursos XML externos declara la identificación de información
sobre el accesorio que desee detectar  -->
<meta-data
android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
                android:resource="@xml/accessory_filter" />
      </activity>
   </application>
</manifest>
```

accessory_filter.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
<usb-accessory manufacturer="UNIVERSIDAD_TÉCNICA_DE_AMBATO"
model="ANDROID_ARDUINO" version="1.0" />
</resources>
```

strings.xml

```
<resources>
    <string name="app_name">Parking</string>
   <string name="toast_error_camera_preview">No se puede inicializar la
    camara previa.</string>
   <string name="distancia">L:</string>
   <string name="distancia2">R:</string>
   <string name="distancia3">F:</string>

</resources>
```

colors.xml

```
<resources>
    <color name="camera_preview_background">#000000</color>
    <color name="BLANCO">#FFFFFF</color>
    <color name="NARANJA">#F28220</color>
    <color name="MORADO">#C733C9</color>
    <color name="PISTACHO">#C0D787</color>
    <color name="NEGRO">#000000</color>
    <color name="ROJO">#DE0814</color>
    <color name="ROSA">#E13157</color>
    <color name="SALMON">#FA8071</color>
    <color name="VERDE">#7EF40A</color>
    <color name="AMARILLO">#FCFA36</color>
    <color name="CELESTE">#0DDBD9</color>
    <color name="COBALTO">#0C23B1</color>
</resources>
```

ANEXO 2

- **CÓDIGO DE PROGRAMACIÓN EN ARDUINO**

```
// manejan vectores o matrices de datos
#include "variant.h"
// cabecera estandar E/S
#include <stdio.h>
// comunicación entre Arduino y dispositivos Android, tanto móviles como tablets
#include <adk.h>
// Configuración del Sensor A
// Trigger es el que se encarga de enviar la señal por el emisor cada cierto intervalo de tiempo
int PIN_TRIG_A = 50;
// Eco es la señal de retorno que ingresa al receptor
int PIN_ECO_A = 51;
// variable para almacenar solo números positivos
int TIME_A = 0;

// Configuración del Sensor B
int PIN_TRIG_B = 42;
int PIN_ECO_B = 43;
int TIME_B = 0;

// Configuración del Sensor C
int PIN_TRIG_C = 40;
int PIN_ECO_C= 41;
int TIME_C = 0;

// Configuración del Sensor D
int PIN_TRIG_D = 36;
int PIN_ECO_D = 37;
int TIME_D = 0;
```

```
// Configuración del Sensor E
int PIN_TRIG_E = 32;
int PIN_ECO_E= 33;
int TIME_E = 0;

// Configuración del Sensor F
int PIN_TRIG_F = 28;
int PIN_ECO_F = 29;
int TIME_F = 0;

// Declaración de variables
int SENSOR_A = 0;
int SENSOR_B = 0;
int SENSOR_C = 0;
int SENSOR_D = 0;
int SENSOR_E = 0;
int SENSOR_F = 0;
int SENSORA = 0;
int SENSORB = 0;
int SENSORC = 0;
int SENSORD = 0;
int SENSORE = 0;
int SENSORF = 0;
int FRONTAL = 0;
int IZQUIERDA;
int DERECHA;
int F;
int L;
int R;
int AUX = 25;
int SONIDO = 24;
int PAUSE;
```

```
// Descriptor de accesorios. Es la forma como Arduino identifica a Android.
// la aplicación instalada en Android
char applicationName[] = "Parking";
// Nombre del Accesorio (Necesita ser el mismo definido en la aplicación para
Android)
char accessoryName[] = "ANDROID_ARDUINO";
// Nombre de la Compañia (Necesita ser el mismo definido en la aplicación para
Android)
char companyName[] = "UNIVERSIDAD_TÉCNICA_DE_AMBATO";
// versión (Necesita ser el mismo definido en la aplicación para Android)
char versionNumber[] = "1.0";
// Número del serial
char serialNumber[] = "1";
// Link de ayuda para aplicaciones en Android
char url[] = "http://www.aidilab.it";
// Controlador de host USB para soportar controladores de cliente USB.
USBHost Usb;
//inicializa el accesorio
ADK adk(&Usb, companyName, accessoryName, applicationName,
versionNumber, url, serialNumber);

void setup()
{
   // Declaración de cada pin como entrada o salida
   pinMode(PIN_TRIG_A, OUTPUT);
   pinMode(PIN_ECO_A, INPUT);
   pinMode(PIN_TRIG_B, OUTPUT);
   pinMode(PIN_ECO_B, INPUT);
   pinMode(PIN_TRIG_C, OUTPUT);
   pinMode(PIN_ECO_C, INPUT);
   pinMode(PIN_TRIG_D, OUTPUT);
   pinMode(PIN_ECO_D, INPUT);
```

```
    pinMode(PIN_TRIG_E, OUTPUT);
    pinMode(PIN_ECO_E, INPUT);
    pinMode(PIN_TRIG_F, OUTPUT);
    pinMode(PIN_ECO_F, INPUT);
    pinMode(SONIDO, OUTPUT);
    Serial.begin(115200);
    pinMode(AUX, OUTPUT);
    // Habilita todas las interrupciones (Pines)
    cpu_irq_enable();
    //Texto enviado a Android
    printf("\r\nADK demo start\r\n");
    delay(10);
}

#define RCVSIZE 128

void loop()
{
  //pin en estado bajo
  digitalWrite(PIN_TRIG_A, LOW);
  delayMicroseconds(2);
  //pin en estado alto
  digitalWrite(PIN_TRIG_A, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG_A, LOW);
  //lee un pulso ya sea en alto o en bajo (espera a que el pin sea HIGH
  //empieza a cronometrar, espera a que el pin sea LOW y detiene la medida de
  tiempo)
  TIME_A = pulseIn(PIN_ECO_A, HIGH);
  //cálculo de la distancia en cm (velo.sonido*tiempo)/2,
  sonido=345m/s=0.0345cm/micros
  SENSORA = ((TIME_A/2)/29);
```

```
SENSOR_A = (0.1*SENSORA + SENSORA);

digitalWrite(PIN_TRIG_B, LOW);
delayMicroseconds(2);
digitalWrite(PIN_TRIG_B, HIGH);
delayMicroseconds(10);
digitalWrite(PIN_TRIG_B, LOW);
TIME_B = pulseIn(PIN_ECO_B, HIGH);
SENSORB = ((TIME_B/2)/29);
SENSOR_B = (0.1*SENSORB+SENSORB);

digitalWrite(PIN_TRIG_C, LOW);
delayMicroseconds(2);
digitalWrite(PIN_TRIG_C, HIGH);
delayMicroseconds(10);
digitalWrite(PIN_TRIG_C, LOW);
TIME_C = pulseIn(PIN_ECO_C, HIGH);
SENSORC = ((TIME_C/2)/29);
SENSOR_C = (0.1*SENSORC+SENSORC);

digitalWrite(PIN_TRIG_D, LOW);
delayMicroseconds(2);
digitalWrite(PIN_TRIG_D, HIGH);
delayMicroseconds(10);
digitalWrite(PIN_TRIG_D, LOW);
TIME_D = pulseIn(PIN_ECO_D, HIGH);
SENSORD = ((TIME_D/2)/29);
SENSOR_D = (0.1*SENSORD+SENSORD);

digitalWrite(PIN_TRIG_E, LOW);
delayMicroseconds(2);
digitalWrite(PIN_TRIG_E, HIGH);
```

```
delayMicroseconds(10);
digitalWrite(PIN_TRIG_E, LOW);
TIME_E = pulseIn(PIN_ECO_E, HIGH);
SENSORE = ((TIME_E/2)/29);
SENSOR_E = (0.1*SENSORE+SENSORE);

digitalWrite(PIN_TRIG_F, LOW);
delayMicroseconds(2);
digitalWrite(PIN_TRIG_F, HIGH);
delayMicroseconds(10);
digitalWrite(PIN_TRIG_F, LOW);
TIME_F = pulseIn(PIN_ECO_F, HIGH);
SENSORF = ((TIME_F/2)/29);
SENSOR_F = (0.1*SENSORF+SENSORF);

// Declaración de una matriz
  uint8_t bufRead[RCVSIZE];
  uint32_t nbread = 0;
  uint8_t bufWrite[8];

// Comienza a realizar la tarea de enviar los datos
  Usb.Task();
  if (adk.isReady()) {
      // leer datos de la matriz leerbuffer
      adk.read(&nbread, RCVSIZE, bufRead);
      if (nbread > 0) {
        // compara los daros recibidos
        if (bufRead[0] == 1)
          digitalWrite(AUX, HIGH);
        else
          digitalWrite(AUX, LOW);
      }
```

```
if (SENSORA < SENSORB) {
  FRONTAL=SENSORA;
 } else
  FRONTAL = SENSORB;
 if (SENSORC < SENSORD) {
  IZQUIERDA=SENSORC;
 } else
  IZQUIERDA = SENSORD;
 if (SENSORE < SENSORF) {
  DERECHA=SENSORE;
 } else
 if (IZQUIERDA < DERECHA) {
  PAUSE=IZQUIERDA;
 } else
  PAUSE = DERECHA;

 if (FRONTAL >0 && FRONTAL <40){
  F = 0;
 }if (FRONTAL >40 && FRONTAL <50){
  F = 40;
 }if (FRONTAL >50 && FRONTAL <60){
  F = 50;
 }if (FRONTAL >60 && FRONTAL <70){
  F = 60;
 }if (FRONTAL >70 && FRONTAL <80){
  F = 70;
 }if (FRONTAL >80 && FRONTAL <90){
  F = 80;
 }if (FRONTAL >90 && FRONTAL <100){
  F = 90;
 }if (FRONTAL >100 && FRONTAL <110){
  F = 100;
```

```
}if (FRONTAL >110 && FRONTAL <120){
  F = 110;
}if (FRONTAL >120 && FRONTAL <130){
  F = 120;
}if (FRONTAL >130 && FRONTAL <140){
  F = 130;
}if (FRONTAL >140 && FRONTAL <150){
  F = 140;
}if (FRONTAL >150 && FRONTAL <160){
  F = 150;
}if (FRONTAL >160 && FRONTAL <170){
  F = 160;
}if (FRONTAL >170 && FRONTAL <180){
  F = 170;
}if (FRONTAL >180 && FRONTAL <190){
  F = 180;
}if (FRONTAL >190 && FRONTAL <200){
  F = 190;
 }if (FRONTAL >200){
  F = 200;

}if (IZQUIERDA >0 && IZQUIERDA <40){
  L = 0;
}if (IZQUIERDA >40 && IZQUIERDA <50){
  L = 40;
}if (IZQUIERDA >50 && IZQUIERDA <60){
  L = 50;
}if (IZQUIERDA >60 && IZQUIERDA <70){
  L = 60;
}if (IZQUIERDA >70 && IZQUIERDA <80){
  L = 70;
}if (IZQUIERDA >80 && IZQUIERDA <90){
```

```
  L = 80;
}if (IZQUIERDA >90 && IZQUIERDA <100){
  L = 90;
}if (IZQUIERDA >100 && IZQUIERDA <110){
  L = 100;
}if (IZQUIERDA >110 && IZQUIERDA <120){
  L = 110;
}if (IZQUIERDA >120 && IZQUIERDA <130){
  L = 120;
}if (IZQUIERDA >130 && IZQUIERDA <140){
  L = 130;
}if (IZQUIERDA >140 && IZQUIERDA <150){
  L = 140;
}if (IZQUIERDA >150 && IZQUIERDA <160){
  L = 150;
}if (IZQUIERDA >160 && IZQUIERDA <170){
  L = 160;
}if (IZQUIERDA >170 && IZQUIERDA <180){
  L = 170;
}if (IZQUIERDA >180 && IZQUIERDA <190){
  L = 180;
}if (IZQUIERDA >190 && IZQUIERDA <200){
  L = 190;
}if (IZQUIERDA >200){
  L = 200;

}if (DERECHA >0 && DERECHA <40){
  R = 0;
}if (DERECHA >40 && DERECHA <50){
  R = 40;
}if (DERECHA >50 && DERECHA <60){
  R = 50;
```

```
}if (DERECHA >60 && DERECHA <70){
  R = 60;
}if (DERECHA >70 && DERECHA <80){
  R = 70;
}if (DERECHA >80 && DERECHA <90){
  R = 80;
}if (DERECHA >90 && DERECHA <100){
  R = 90;
}if (DERECHA >100 && DERECHA <110){
  R = 100;
}if (DERECHA >110 && DERECHA <120){
  R = 110;
}if (DERECHA >120 && DERECHA <130){
  R = 120;
}if (DERECHA >130 && DERECHA <140){
  R = 130;
}if (DERECHA >140 && DERECHA <150){
  R = 140;
}if (DERECHA >150 && DERECHA <160){
  R = 150;
}if (DERECHA >160 && DERECHA <170){
  R = 160;
}if (DERECHA >170 && DERECHA <180){
  R = 170;
}if (DERECHA >180 && DERECHA <190){
  R = 180;
}if (DERECHA >190 && DERECHA <200){
  R = 190;
}if (DERECHA >200){
  R = 200;
}
```

```
if (PAUSE <= 40)
        {
          digitalWrite(SONIDO, HIGH);
        }
         else if(PAUSE > 40 && PAUSE <= 50)
        {
           digitalWrite(SONIDO, HIGH);
           delay (30);
           digitalWrite(SONIDO, LOW);
           delay (30);
         }
          else if (PAUSE > 50 && PAUSE <= 60)
         {
           digitalWrite(SONIDO, HIGH);
           delay (40);
           digitalWrite(SONIDO, LOW);
           delay (40);
         }
          else if (PAUSE > 60 && PAUSE <= 70)
         {
           digitalWrite(SONIDO, HIGH);
           delay (50);
           digitalWrite(SONIDO, LOW);
           delay (50);
         }
         else if (PAUSE > 70 && PAUSE <=80)
         {
           digitalWrite(SONIDO, HIGH);
           delay (60);
           digitalWrite(SONIDO, LOW);
           delay (60);
         }
```

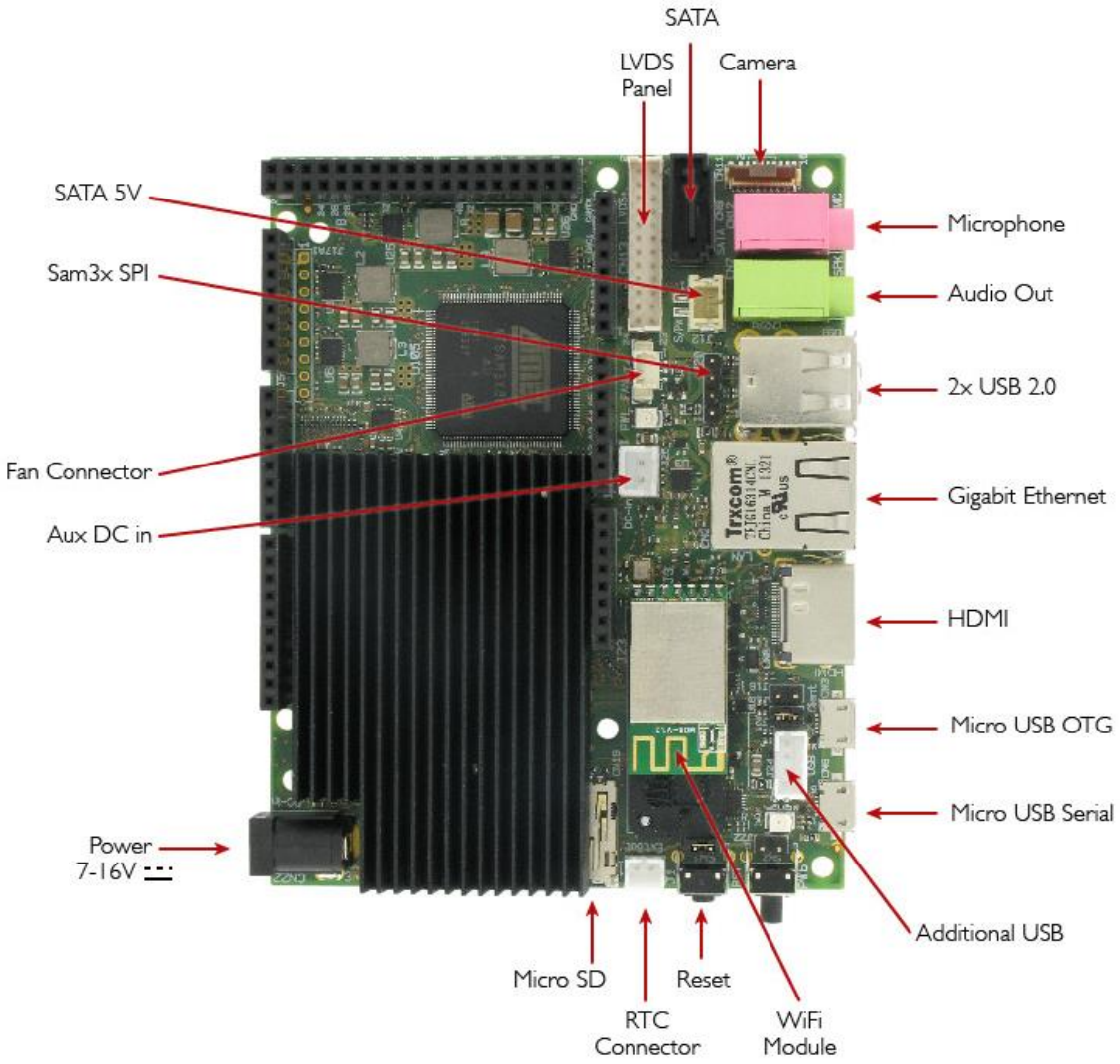
```
else if (PAUSE > 80 && PAUSE <= 90)
{
  digitalWrite(SONIDO, HIGH);
  delay (70);
  digitalWrite(SONIDO, LOW);
  delay (70);
}
else if (PAUSE > 90 && PAUSE <= 100)
{
  digitalWrite(SONIDO, HIGH);
  delay (80);
  digitalWrite(SONIDO, LOW);
  delay (80);
}
else if (PAUSE > 100 && PAUSE <= 120)
{
  digitalWrite(SONIDO, HIGH);
  delay (90);
  digitalWrite(SONIDO, LOW);
  delay (90);
}
else if (PAUSE > 120 && PAUSE <= 140)
{
  digitalWrite(SONIDO, HIGH);
  delay (100);
  digitalWrite(SONIDO, LOW);
  delay (100);
}
else if (PAUSE > 140 && PAUSE <=160)
{
  digitalWrite(SONIDO, HIGH);
  delay (110);
```

```
  digitalWrite(SONIDO, LOW);
  delay (110);
 }
 else if (PAUSE > 160 && PAUSE <=180)
 {
  digitalWrite(SONIDO, HIGH);
  delay (120);
  digitalWrite(SONIDO, LOW);
  delay (120);
 }
 else if (PAUSE > 180)
 {
  digitalWrite(SONIDO, HIGH);
  delay (500);
  digitalWrite(SONIDO, LOW);
  delay (500);
 }
 else
 {
  digitalWrite(SONIDO, LOW);
 }

// guarda el dato en un espacio de memoria
bufWrite[1] = (L);
bufWrite[2] = (R);
bufWrite[3] = (F);

// escribe la distancia en Android
adk.write(sizeof(bufWrite),(uint8_t *)bufWrite);
adk.write(sizeof(bufWrite),(uint8_t *)bufWrite);
adk.write(sizeof(bufWrite),(uint8_t *)bufWrite);
```

```
    } else {
      digitalWrite(AUX, LOW);
      digitalWrite(SONIDO, LOW);
    }

    delay(1);
}
```

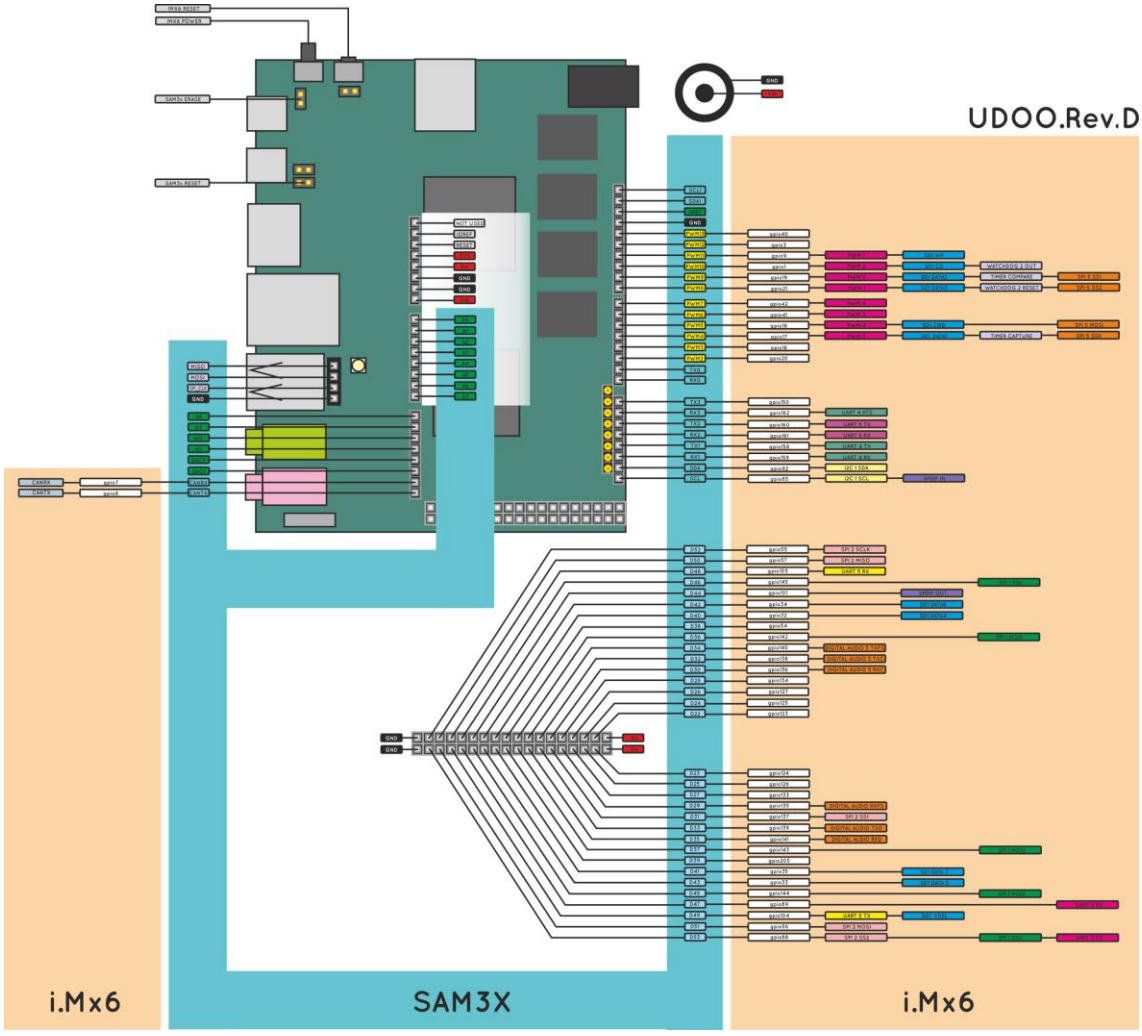
ANEXO 3

- **UBICACIÓN DE LOS PUERTOS EN LA PLACA UDOO**



ANEXO 3

- **DISTRIBUCIÓN DE LOS PINES DEL ARDUINO DUE EN LA PLACA UDOO**



ANEXO 4

- **DIMENSIONES DE LOS AUTOMÓVILES**

CHEVROLET			
Modelo	**Altura (m)**	**Largo (m)**	**Ancho (m)**
Spark	1,522	3,64	1,597
Aveo 3p	1,505	3,92	1,68
Aveo 4p	1,517	4,399	1,735
Aveo 5p	1,515	4,039	1,735
Cruze 4p	1,477	4,597	1,788
Cruze 5p	1,477	4,51	1,797
Cruze SW	1,521	4,681	1,797
Malibu	1,465	4,865	1,855
Volt	1,439	4,498	1,787
Camaro	1,36	4,837	1,917
Corvette	1,246	4,459	1,927
Trax	1,674	4,248	1,776
Orlando	1,633	4,652	1,836
Captiva	1,756	4,673	1,868
	1,5076429	**4,4298571**	**1,7925**

HYUNDAI			
Modelo	**Altura (m)**	**Largo (m)**	**Ancho (m)**
i10	1,5	3,665	1,66
i20	1,49	3,995	1,71
i30	1,47	4,3	1,78
i30cw	1,5	4,485	1,78
Elantra	1,445	4,55	1,775
i40 Sedán	1,47	4,74	1,815
i40 Familiar	1,47	4,77	1,815
ix20	1,6	4,1	1,765
ix35	1,66	4,41	1,82
Veloster	1,399	4,22	1,79
Santa Fe	1,69	4,66	1,88
Sonata	1,475	4,8	1,832
Orlando	1,633	4,652	1,836
	1,5232308	**4,4113077**	**1,7890769**

NISSAN			
Modelo	**Altura (m)**	**Largo (m)**	**Ancho (m)**
Pixo	1,47	3,565	1,6
Micra	1,52	3,825	1,665
Note	1,53	4,1	1,695
Juke	1,57	4,135	1,765
Leaf	1,55	4,445	1,77
Qashqai	1,59	4,379	1,8
Qashqai+2	1,645	4,541	1,78
X-Trail	1,7	4,635	1,79
Pathfinder	1,858	4,813	1,848
Murano	1,72	4,86	1,885
370Z	1,315	4,25	1,845
GT-R 2013	1,37	4,67	1,895
	1,5698333	**4,3515**	**1,7781667**

KIA			
Modelo	**Altura (m)**	**Largo (m)**	**Ancho (m)**
Picanto	1,48	3,595	1,595
Rio	1,455	4,045	1,72
Pro_cee'd	1,43	4,31	1,78
cee'd	1,47	4,31	1,78
cee'd Sportswagon	1,485	4,505	1,78
Venga	1,6	4,068	1,765
Soul	1,66	4,105	1,785
Optima	1,455	4,845	1,83
Sportage	1,63	4,44	1,855
Sorento	1,755	4,685	1,885
Carens	1,61	4,525	1,805
Carnival	1,805	4,81	1,985
	1,5695833	**4,3535833**	**1,7970833**

BMW			
Modelo	Altura (m)	Largo (m)	Ancho (m)
Serie 1	1,421	4,324	1,765
X1	1,545	4,454	1,798
Serie 1 Coupé	1,423	4,36	1,748
Serie 2 Coupé	1,418	4,432	1,774
Serie 2 Active Tourer	1,555	4,342	1,8
M3 Coupé	1,424	4,615	1,804
Serie 3 Berlina	1,429	4,624	1,811
Serie 3 Touring	1,429	4,624	1,811
Serie 3 Gran Turismo	1,508	4,824	1,828
i3	1,578	3,999	1,775
i8	1,298	4,689	1,942
X3	1,678	4,657	1,881
Serie 4 Coupé	1,362	4,638	1,825
M5 Berlina	1,467	4,91	1,891
Serie 5 Berlina	1,464	4,899	1,86
Serie 5 Touring	1,462	4,907	1,86
Serie 5 Turismo	1,559	4,998	1,901
X5	1,762	4,886	1,938
Serie 6 Coupé	1,369	4,894	1,894
M6 Coupé	1,374	4,898	1,899
X6	1,69	4,877	1,983
Serie 7	1,479	5,072	1,902
Z4	1,291	4,239	1,79
	1,5447727	**4,871**	**1,9309091**

FORD			
Modelo	Altura (m)	Largo (m)	Ancho (m)
Ka	1,505	3,62	1,658
Fiesta	1,495	3,969	1,722
Fusion	1,543	4,013	1,724
Focus	1,484	4,358	1,823
Focus Sedán	1,484	4,534	1,823
Focus Sport Break	1,505	4,556	1,823
Kuga	1,702	4,524	1,838
C-MAX	1,626	4,38	1,828
Grand C-MAX	1,684	4,52	1,828
S-MAX	1,66	4,801	1,884
Mondeo	1,5	4,784	1,886
Mondeo 4p	1,5	4,85	1,866
Galaxy	1,709	4,819	1,884
B-MAX	1,604	4,077	1,751
EcoSport	1,665	4,235	1,765
	1,5777333	**4,4026667**	**1,8068667**

TOYOTA			
Modelo	Altura (m)	Largo (m)	Ancho (m)
iQ	1,5	2,985	1,68
AYGO	1,465	3,415	1,615
Yaris	1,51	3,885	1,695
GT86	1,285	4,24	1,775
Auris	1,46	4,56	1,76
Land Cruiser 3p	1,83	4,335	1,885
Verso	1,62	4,46	1,79
Prius	1,49	4,48	1,745
Auris Touring Sports	1,46	4,56	1,76
RAV4	1,66	4,57	1,845
Prius+	1,575	4,615	1,775
Avensis Sedán	1,48	4,71	1,81
Land Cruiser 5p	1,845	4,78	1,885
Avensis Cross Sport	1,48	4,78	1,81
	1,5471429	**4,3125**	**1,7735714**

SUZUKI			
Modelo	Altura (m)	Largo (m)	Ancho (m)
Alto	1,48	3,595	1,595
Splash	1,455	4,045	1,72
Swift	1,43	4,31	1,78
Jimny	1,47	4,31	1,78
SX4	1,485	4,505	1,78
SX4 S-Cross	1,6	4,068	1,765
Grand Vitara	1,66	4,105	1,785
Kizashi	1,455	4,845	1,83
	1,504375	**4,222875**	**1,754375**

MAZDA			
Modelo	Altura (m)	Largo (m)	Ancho (m)
2	1,475	3,92	1,695
3	1,45	4,46	1,795
5	1,615	4,585	1,75
6	1,45	4,865	1,84
6 Wagon	1,48	4,8	1,84
MX-5	1,245	4,02	1,72
CX-5	1,67	4,555	1,84
3 SportSedan	1,47	4,58	1,755
	1,481875	**4,473125**	**1,779375**

Promedio de la distancia del ancho de los vehículos: 1,80021379 metros