UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

FACULTAD DE INGENIERÍA.

CONSTRUCCIÓN DE UN ROBOT MÓVIL CONTROLADO INALÁMBRICAMENTE POR DISPOSITIVOS CON PLATAFORMA ANDROID

TESIS QUE PARA OBTENER EL TITULO DE:

INGENIERO MECATRÓNICO

PRESENTAN:

FRANCISCO ARMANDO LUGO MAYORGA

VÍCTOR MANUEL MENESES AGUILAR

DIRECTOR:

M.I. BILLY ARTURO FLORES MEDERO NAVARRO

ÍNDICE

Capítulo 3. Diseño del circuito electrónico para la obtención y transferencia de datos entre el microcontrolador y el dispositivo móvil.

- 3.1 Selección de componentes y funcionamiento.
- 3.2 Implementación del microcontrolador ATMEGA 328P-PU.
- 3.3 Programas desarrollados durante el proyecto para el ATMEGA 328P-PU.
  - 3.3.1 Para la adquisición de datos vía bluetooth.
  - 3.3.2 Para la movilidad de los motores usando PWM.
  - 3.3.3 Para la adquisición de temperatura.
  - 3.3.4 Para el cálculo del estado de carga de la batería.
  - 3.3.5 Para el tacómetro.
  - 3.3.6 Código final para el control del vehículo de exploración.
- 3.4 Implementación del dispositivo bluetooth BLUESMIRF GOLD.
- 3.5 Alimentación del circuito.
- 3.6 Esquema de la conexión del circuito electrónico.
- 3.7 Diseño del PCB del circuito electrónico.

Capítulo 4. Desarrollo de la aplicación para Android.

- 4.1 ¿Qué es y cómo funciona el SDK para Android?
- 4.2 Componentes de las aplicaciones de Android.
- 4.3 ¿Qué es y cómo funciona Eclipse?
- 4.4 Diseño de la pantalla para el usuario en el teléfono móvil.
  - 4.4.1 Identificación de las partes en la pantalla.
- 4.5 Aplicaciones desarrolladas durante el proyecto
  - 4.5.1 Obtención de datos de los acelerómetros del dispositivo móvil.
  - 4.5.2 Transferencia de datos vía bluetooth.
  - 4.5.3 Recepción de datos procedentes del ARDUINO.

Capítulo 5. Implementación de una cámara en el vehículo para observar la imagen en el dispositivo móvil.

5.1 Selección de cámara.

5.2 Transferencia de imagen al dispositivo móvil.

Capítulo 6. Trabajo a futuro.

6.1 Circuito electrónico reducido.

6.1.1 Selección de nuevos componentes.

6.1.2 Diseño de la placa del circuito impreso a doble cara y visualización con los componentes puestos.

6.2 Construcción del carro.

6.2.1 Diseño asistido por computadora del chasis.

6.2.2 Selección del método de fabricación.

6.3 Transferencia de video en tiempo real.

Capítulo 7. Resultados.

Conclusiones.

Bibliografía y mesografía.

Anexos

INTRODUCCIÓN

El desarrollo de la tecnología ha sido de suma importancia para el crecimiento de todas las sociedades del mundo. El ser humano debe, en gran parte, su desarrollo a que ha aprendido a vivir en grupos de individuos formando sociedades. Los inventos tecnológicos son un complemento a las actividades diarias de las personas; por lo que han sido un tema de investigación continua. La búsqueda de nuevas herramientas así como de inventos con mayor portabilidad y comodidad; ha tenido ocupados a investigadores e ingenieros alrededor del mundo. Cada día es más común realizar tareas, actividades y trabajos de oficina desde un invento tecnológico, como lo es el teléfono celular.

En los últimos años se ha intentado logrado que el celular le quite terreno a la computadora en tareas como revisar el correo electrónico, mantener conferencias con otras personas, mensajería instantánea, entretenimiento, elaboración de documentos y presentaciones por mencionar sólo algunas. De igual forma hay que tomar en cuenta un invento que revolucionó el campo de los celulares como lo es la plataforma Android, la cual en los últimos años ha crecido de manera notable y se ha convertido en uno de los sistemas operativos de vanguardia para los teléfonos móviles.

Por otro lado, en la continua búsqueda de conocimiento, el hombre se ha encontrado con ciertas barreras entre las que destacan: las de tipo físico (lugares con dimensiones poco accesibles) y las que atentan contra su integridad física. Para ello se ha visto en la necesidad de usar la tecnología en las tareas de exploración de terreno, monitoreo de actividades e investigación de fenómenos naturales y artificiales. En estas tareas se utilizan los vehículos de exploración a control remoto. Para este trabajo se definirá al vehículo de exploración como aquel que controlado a distancia, puede enviar y recibir información referente al terreno en el que se desenvuelve (imágenes, temperatura, etc.).

Tomando en cuenta los puntos anteriores, en este proyecto se busca interrelacionar a estos dos inventos (celular y vehículo de exploración) conectándolos de una forma creativa e innovadora. De igual forma se muestra un panorama general de lo que existe ya en el mercado y hacia donde van las compañías que se dedican a este tema. Se describe la aplicación deseada tomando en cuenta los parámetros críticos para su éxito y se hace una comparativa de las ventajas y desventajas que tendría el resultado final.

El alcance que se propone en este trabajo es el de crear una aplicación para el dispositivo con plataforma Android; así como los códigos necesarios para el microcontrolador de manera que se comunique inalámbricamente con el vehículo explorador. De esta manera se pueden enviar instrucciones desde el teléfono celular como: la velocidad y dirección de cada uno de los motores (utilizando PWM), la instrucción de pausa para ver el estado del vehículo y de manera similar se reciben datos del microcontrolador para desplegarlos en la pantalla del móvil como: la temperatura ambiente, temperatura de la etapa de potencia, velocidad a la que se desplaza, distancia total recorrida e imágenes del entorno en donde se encuentra el vehículo.

PLANTEAMIENTO DEL PROBLEMA

La tecnología se desarrolla de manera impresionante y cada vez surgen nuevos inventos que intenta facilitar la vida del hombre. Sin embargo, en la búsqueda de conocimiento, el hombre se ha encontrado con ciertas dificultades que un vehículo explorador controlado a distancia podría solucionar. Algunas de estas tareas son: la investigación de lugares poco accesibles para el tamaño de un humano, ambientes contaminados y dañinos para la salud, situaciones que pueden resultar en la muerte de un ser vivo o comprometer su integridad física. Junto a estas actividades es importante integrar la comodidad del control a distancia y que mejor que mediante un dispositivo como el teléfono celular, acompañado de una de las plataforma de vanguardia como lo es Android.

HIPÓTESIS

Es posible controlar el vehículo explorador usando un dispositivo con plataforma Android, que sea capaz de enviar instrucciones para los motores y recibir información del microcontrolador en el vehículo vía Bluetooth al mismo tiempo que recibe imágenes del entorno vía Wi-Fi.

Es posible tener una buena maniobrabilidad del vehículo con el control inalámbrico y poder seguir una ruta preestablecida.

OBJETIVOS

- Diseñar y construir un robot móvil para exploración controlado por un dispositivo con plataforma Android utilizando comunicación inalámbrica.
- Permitir al usuario conocer el estado del robot móvil en cuanto a estado de carga de batería, velocidad, distancia recorrida, temperatura de la etapa de potencia y temperatura ambiente.
- Implementar una cámara en el robot móvil y visualizar la imagen en el dispositivo controlador.

CAPÍTULO 1. MARCO TEÓRICO

1.1 Estado del arte: panorama general de lo que se ha hecho del tema.

El teléfono celular revolucionó la vida de las personas y de las comunicaciones. Una infinidad de tareas se realizan con este dispositivo. El primer antecedente respecto al teléfono móvil lo tiene la compañía Motorola en la década de los 80's. El ingeniero encargado de diseñar este dispositivo fue Rudy Krolopp (Fig 1-1).

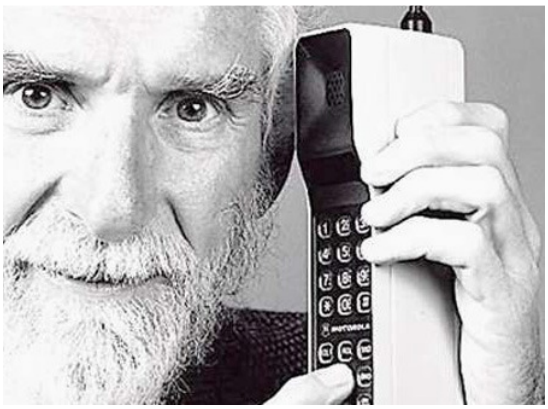


Fig 1-1 Rudy Krolopp y el primer celular, 1983

Los primeros teléfonos celulares eran sistemas embebidos (dispositivos que realizan una acción dependiendo únicamente de la entrada) que llamaban muy poco la atención de los programadores por su complejo software que era usado para programarlos y su difícil acceso al hardware, ya que los fabricantes guardaban celosamente los secretos de cada dispositivo. Con la aparición de dispositivos más complejos (como el GPS) se comenzaron a usar sistemas operativos estandarizados (sobre todo se basaban en Linux). La mayor parte de estos se programaban en C/C++.

Después surgieron plataformas como Symbian, aunque el código para realizar aplicaciones era aún demasiado complejo. Finalmente se crearon los MIDlets de Java que se podían ejecutar en una máquina virtual de Java (la J2ME) sobre los dispositivos móviles aunque seguían existiendo restricciones al acceso del hardware, al núcleo de la plataforma del dispositivo móvil, a aplicaciones específicas de la telefonía móvil y que solamente se podía ejecutar una aplicación a la vez. En la actualidad los dispositivos móviles han creado su propia rama en el camino de la tecnología: dentro de Android, cada aplicación genera su propia máquina virtual y cada una trabaja de manera aislada. Ahora son dispositivos sumamente complejos capaces de realizar tareas similares a los equipos de cómputo tradicionales.

En la actualidad existen muchos sistemas operativos para la telefonía celular. Este proyecto se centra en al sistema operativo Android y la relación que puede llegar a tener con los vehículos exploradores haciendo el uso de una aplicación específica para el control y envío de información.

1.1.1 ¿Qué es Android?

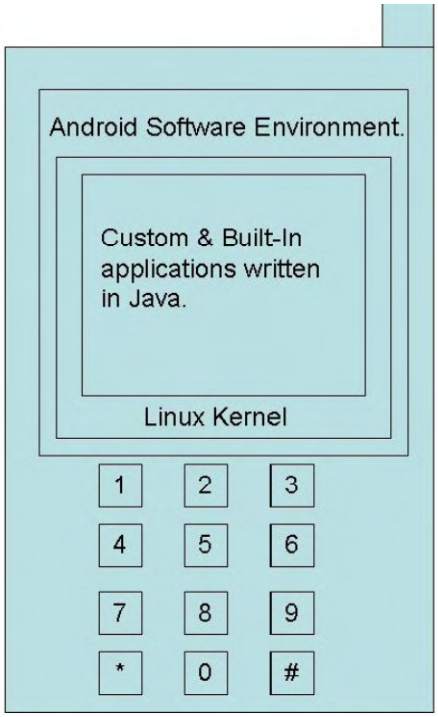


Fig 1-2 Elementos básicos que componen Android

Android es el sistema operativo con el crecimiento más rápido durante los últimos años originalmente creado para dispositivos móviles, tablets y que se ha desarrollado hasta abarcar a las netbooks y PCs con el Android-x86 (los equipos de cómputo si pueden operar con el sistema operativo de Android aunque evidentemente fue desarrollado específicamente para dispositivos móviles por lo que muchas funciones no pueden ser utilizadas debido al hardware; más que nada es de demostración). No es un lenguaje de programación; es un entorno de software donde podemos correr ciertas aplicaciones. La principal característica de este sistema operativo es que está basado en Linux (versión 2.6 del Kernel), es libre, gratuito, multiplataforma, ofrece agilidad y portabilidad para aprovechar las numerosas opciones de hardware de los futuros teléfonos equipados con Android.

Incluye la máquina virtual “Dalvik” optimizada para poder ejecutar código Java en dispositivos móviles, que es un lenguaje muy popular para los desarrolladores. Los controladores para componentes de hardware están programados en C/C++. Las aplicaciones de fábrica y las creadas por terceros tienen la misma prioridad dentro del sistema y se comportan de manera muy similar.

Se dice que es un sistema operativo libre: no se paga nada para programar en este sistema ni para incluirlo a un teléfono. Realmente es algo muy ambicioso ya que normalmente el software “libre” o las licencias “gratuitas” tienen un costo de propiedad en asistencia, en hardware o suelen incluir contratos anuales más impuestos. Gracias a que es software libre, cada vez se desarrolla más y más código ya que la comunidad de desarrolladores proporciona los elementos de los que carece, no se depende de Google para instalar actualizaciones y cada quien puede crear su propia versión.

Se comercializa bajo dos licencias: el núcleo Linux bajo la GPL (General Public License) como exige todo núcleo de sistema operativo que sea código abierto mientras que la plataforma Android sin el núcleo bajo la ASL (Apache Software License). No es el primer sistema operativo para móviles de código abierto pero si es el primero con un respaldo de un gigante como lo es Google. Incluye el WebKit que emplea igualmente Mac en sus dispositivos: es el motor de navegación líder en el mercado. De igual manera es un proyecto de software libre cuyo objetivo es asemejar la experiencia de navegación en el móvil como si fuera a través de una Pc.

Logo de Android diseñado por Ascender Corporation

Los componentes principales de Android se explican a continuación:

***Aplicaciones (Applications)*:** se incluyen correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Estas aplicaciones están escritas en lenguaje de programación Java.

Marco de trabajo de aplicaciones (Application Framework): los desarrolladores tiene libre acceso a las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes. Los componentes pueden ser reemplazados por el usuario. Cualquier aplicación puede publicar sus capacidades y se puede hacer uso de las mismas por otra aplicación.

Bibliotecas (Libraries): se incluye un conjunto de bibliotecas de C/C++ que utilizan varios componentes del sistema entre las que destacan: bibliotecas de medios, bibliotecas de gráficos, System C library (implementación biblioteca C estándar), 3D, etc.

Runtime de Android (Android Runtime): cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Un dispositivo puede correr múltiples máquinas virtuales de manera eficiente. La máquina virtual está basada en registros y corre clases que han sido compiladas por el compilador java y transformadas al formato .dex por una herramienta adicional denominada “dx”.

Núcleo Linux (Linux Kernel): Android depende de Linux para todos los servicios base como son: seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo sirve como una capa de abstracción entre el hardware y el resto de la pila del software.

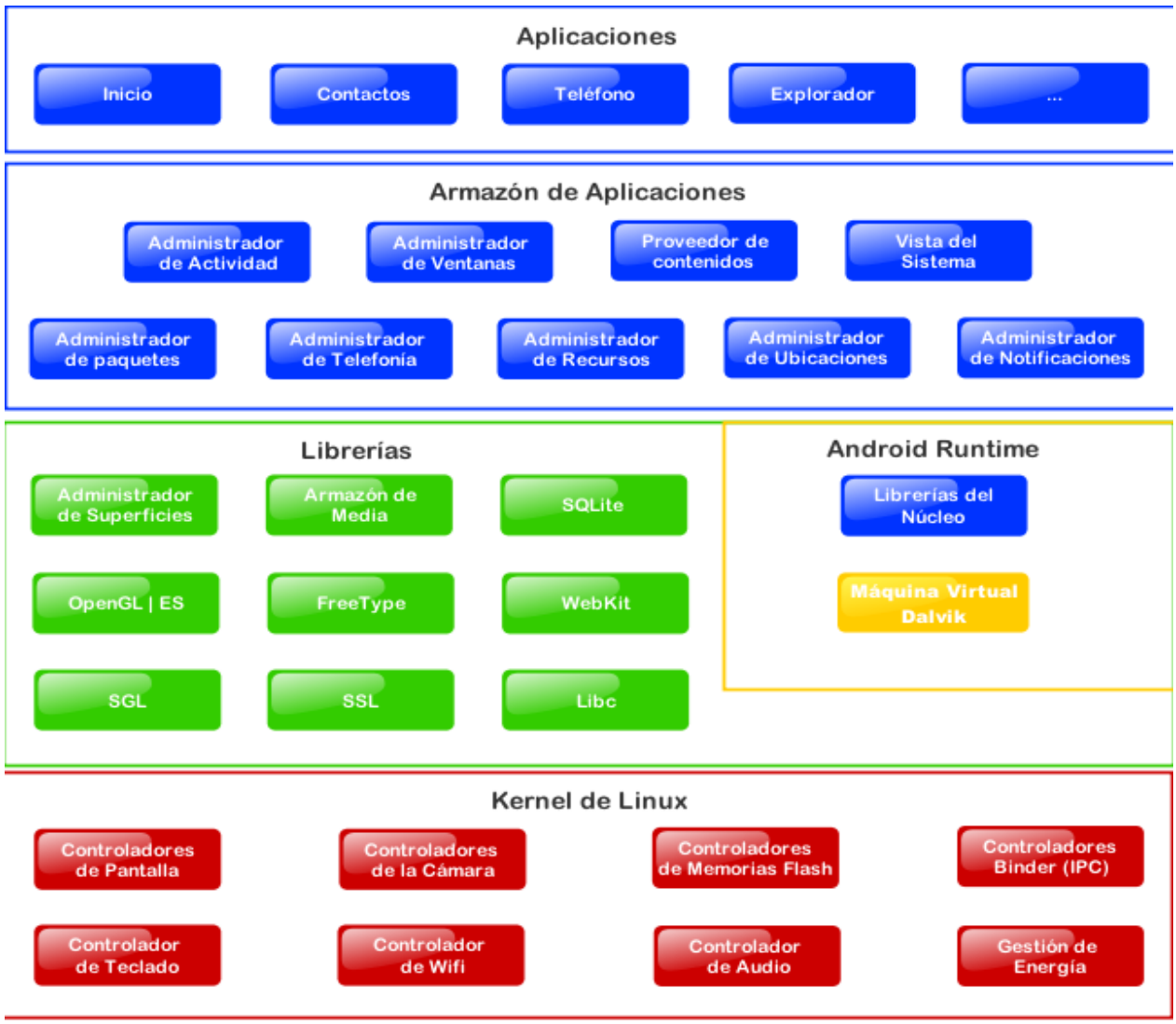


Diagrama de la arquitectura de Android

1.1.2 Evolución de Android.

En julio del 2005 el gigante Google adquirió Android Inc. con la intención de entrar en el mercado de la telefonía móvil. Google desarrolló una plataforma basada en el Kernel de Linux promocionando su flexibilidad y fácil actualización. Para finales del 2006, se habían registrado solicitudes de patentes en el área de la telefonía móvil por parte de Google.

Para noviembre del 2007 se había formado la Open Handset Alliance (Fig 1-2), una agrupación de compañías de telefonía, electrónica y desarrollo tecnológico. Esta agrupación permitió que se lanzara en el 2007 el primer producto Android. Para el 2008 se habían unido al grupo compañías como: Toshiba y Sony Ericsson.

Fig 1-3 Open Handset Alliance.

Con esto se puede decir que Android es la herramienta de Google para abarcar el mercado de la telefonía móvil y fomentar la necesidad de estar conectado constantemente a Internet. Algunos puntos a destacar por parte de Google es que siempre ha dado al usuario aplicaciones que funcionan de manera muy sencilla y sobre todo la facilidad de poderlas crear y distribuir libremente (es la plataforma preferida por los desarrolladores).

A continuación se muestra una tabla con las diferentes versiones de Android, sus sobrenombres y las fechas de lanzamiento al mercado:

Versión	Nombre	Fecha de lanzamiento
Beta		12 de noviembre del 2007
1.0		23 de septiembre del 2008
1.1		9 de febrero del 2009
1.5	Cupcake	30 de abril del 2009
1.6	Donut	15 de septiembre del 2009
2.0/2.1	Eclair	26 de octubre del 2009
2.2	Froyo	20 de mayo del 2010
2.3	Gingerbread	6 de diciembre del 2010
3.0/3.1	Honeycomb	22 de febrero del 2011
4.0	Ice Cream Sandwich	19 de octubre del 2011

1.1.3 Android en la actualidad.

Como se ha mencionado, los últimos avances que tiene Android se están desarrollando para tablets, netbooks y PCs. La versión 3.0, cuya clave es Honeycomb está especializada para brindar un mejor soporte para tablets. Entre las características principales destacan: un sistema multitarea mejorado, soporte para una gran variedad de periféricos, escritorio 3D, entre muchas otras.

Android es la plataforma con mayor crecimiento en el mercado en el último año. Su principal competidor es Apple, cuyo éxito se ha basado en el diseño novedoso y cautivador para los jóvenes. Para tener alguna idea de lo que se está hablando, Android pasará de ocupar el 25% del mercado en el 2010 a tener el 50% a finales del 2011.

Hasta octubre del 2011, se tenían 319,161 aplicaciones para Android y 459,589 para Apple. En diciembre del mismo año, Google anunció que existen más de 200 millones de dispositivos Android activados alrededor del mundo, así como así como más de 10 billones de descargas desde el Market. Tomando en cuenta que para el 2014 se espera que el número de ventas de teléfonos inteligentes este alrededor de los 500 millones de ejemplares; nos damos una idea de la importancia económica que tiene Android hoy en día.

Sin embargo cabe destacar que específicamente en el mercado europeo el 40% de los compradores de teléfonos inteligentes prefieren un Iphone mientras que sólo el 19% tiene en mente un Android (Fig 1-3). En Estados Unidos de América a fines del año 2011, el 46.9% de los smarthphones obtenidos fueron Android, mientras el 28.7% prefirió el iPhone y el 16.6% prefirieron el Blackberry (datos obtenidos del sitio oficial de la revista PC: http://www.pcmag.com).

Con estas cifras se pretende que este proyecto tenga el impacto deseado en el mercado americano. La continua ascendencia de Android es un punto a favor; podría traer consigo una mayor aceptación por parte del mercado y mucho más desarrollo a futuro. Actualmente con la versión 4.0 (Ice Cream Sandwich) Android competirá contra el Windows Phone Mango y el iOS 5.

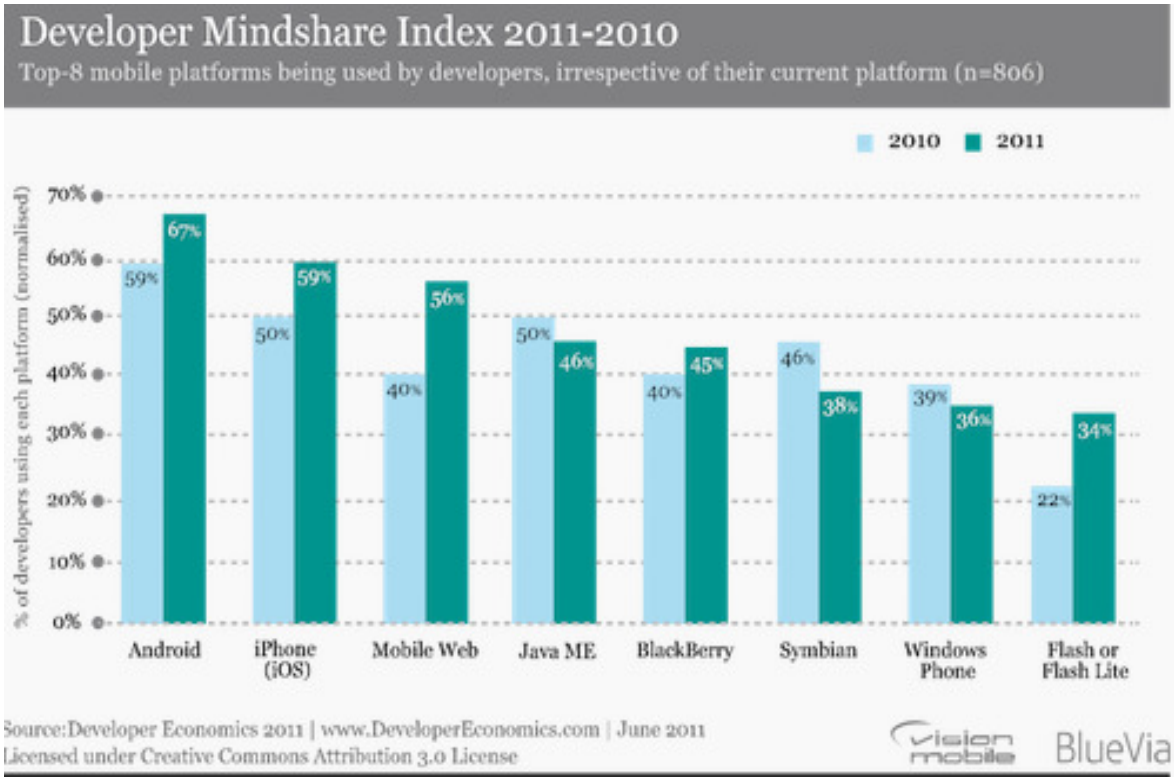


Fig 1-4 Gráfica que compara las preferencias de sistemas operativos móviles entre desarrolladores de aplicaciones.

Android e Intel han anunciado su alianza, por lo que no faltará mucho tiempo para ver Android ejecutándose plenamente en equipos de cómputo. También los equipos móviles se verán beneficiados ya que ahora contaran con núcleos Intel y serán más potentes.

Todo el entorno (SDK y el ADT) para programar la nueva versión de Android sufrió un gran cambio ya que cuenta con nuevas características más dinámicas, elegantes e interesantes. Está diseñado para funcionar tanto con tabletas como con teléfono celulares.

La interfaz de usuario es más simple y llena de animaciones. Cuenta con los botones virtuales de Home para ir a la pantalla de inicio, el botón para regresar a la vista anterior y el revolucionario botón multitarea para poder cambiar de aplicación rápidamente. Ahora no solo nos permite colocar iconos en el inicio, si no emplear cada aplicación como un Widget y ajustarlo al tamaño que queramos. Se puede colgar una llamada y mandar automáticamente un mensaje de texto personalizado seleccionando esta opción al momento de que entra una llamada.

También cuenta con un avanzado sistema de entrada de voz que nos deja dictarle al teléfono el texto que queramos por un tiempo indefinido usando el lenguaje que

queramos. Cuenta con varias nuevas características en el software de la cámara tanto para fotos como para video. Incluye muchas características y propiedades de la interfaz de usuario del Android 3.x y del 2.x así como varias de sus APIs. Otro detalle innovador que posé es la capacidad de desbloqueo por medio de reconocimiento facial.

Fig 1-5 Logo usado para el Ice Cream Sandwich

Intel también jugará un papel importante en esta versión de Android, ya que destinará un nuevo chip (Atom de núcleo Medfield) para los dispositivos Android ya que hace un par de meses Intel anunció su alianza con Google. Otra cosa que vale la pena destacar es que el asistente de voz personal del iPhone 4S (Siri) ya tiene rival (Iris) que es la versión alfa creada por los desarrolladores de Google. Por lo que se ve claramente que Android no se quedará atrás de su gran rival.

1.1.4 ¿Por qué Android?

- Android permite emplear cualquier herramienta de Hardware o cualquier aplicación de software el dispositivo simplemente pidiendo los permisos necesarios dentro del Android Manifest.xml de la aplicación creada.
- Soporta XML (eXtensible Markup Language) que es sencillo y estándar.
- Google incluyó la comunidad de desarrolladores http://code.google.com/android Aun así existen infinidad de foros, código e información que apoyan a desarrollar cualquier aplicación.
- El código fuente se escribe en Java (sin incluir el Layout ni el Manifest) y hay mucha información para aprender a usarlo ya que es un lenguaje muy común.
- No importa la marca del equipo; solo la versión de Android con que cuenta y el hardware con que se cuenta (depende de lo que requiera la aplicación).

- Es muy fácil subir las aplicaciones desarrolladas al Market (comparado con sus rivales) o se pueden distribuir por los medios que el desarrollador desee.
- Es código abierto y no hay que pagar por su desarrollo ni para obtener ningún software.
- Android está diseñado para sacar el máximo provecho al Hardware de sus dispositivos. Si se encuentra con alguna limitante por parte del Android, siempre se pueden esquivar.
- Para programar en Android no hay que considerar la marca del equipo, si no la versión de Android con la que cuenta y algunas limitantes de Hardware (no todos los equipos cuentan con las mismas características).
- ofrece un mayor rendimiento en el mismo hardware y con un coste menor de adquisición de software.
- Android no controla absolutamente el mercado de aplicaciones; los desarrolladores pueden ofrecer sus aplicaciones a través de diversos medios como el Market, páginas como Handango o en sus portales de Internet.
- Tanto las aplicaciones de usuario como las básicas de Android se escriben en Java y se compilan en código de Bytes (archivos dex), que se interpretan en tiempo de ejecución por medio de la máquina virtual Dalvik.

1.1.5 Vehículos de exploración y sus aplicaciones.

Tomando en cuenta que para este proyecto se define al vehículo de exploración como aquel que controlado a distancia, puede enviar información referente al terreno en el que se desenvuelve (imágenes, temperatura, etc.); existen varias aplicaciones de gran interés actual. Las aplicaciones más conocidas de vehículos de exploración se encuentran en el ámbito aeroespacial (para toma de fotografías en el espacio y reconocimiento de terreno en atmósferas desconocidas), militar (localización de objetivo, monitoreo, reconocimiento de terreno, robots bombas), de espionaje (toma de fotografías, video, audio), desmantelamiento de bombas y explosivos (control remoto y envió de imágenes) entre otros.

Como se mencionó en la introducción; este proyecto de centra en la exploración y reconocimiento de terreno poco accesible o peligroso para los humanos mediante imágenes y controlando el vehículo a distancia.

1.2 Comunicación inalámbrica entre el circuito electrónico y el dispositivo móvil.

La comunicación es un elemento crucial y de suma importancia para este proyecto. Para lo que se quiere obtener es necesario establecer una comunicación de tipo inalámbrica entre el dispositivo móvil y el microcontrolador del vehículo. Se ha seleccionado este tipo de comunicación por obvias razones; no son necesarios los cables, se puede tener un circuito electrónico que obtenga los datos y se encargue de la comunicación, la facilidad para la instalación en el vehículo y finalmente porque sería un desperdicio tener que conectar el celular a algún circuito mediante cables cuando su característica es la portabilidad.

1.2.1 Formas de comunicación inalámbrica.

Dentro de las diversas formas de comunicación inalámbrica, para efectos de este trabajo analizaremos tres de las más comunes: la radiofrecuencia, el bluetooth y el Wi-Fi. Esto se realiza con el propósito de hacer una buena elección para el objetivo final. Se mencionarán características así como ventajas y desventajas de cada una de estas formas de comunicación inalámbrica.

Radiofrecuencia: este término se utiliza para referirse al espectro de radiofrecuencia o RF, las comunicaciones inalámbricas por RF se pueden dividir en las que no cumplen ningún protocolo estándar (llamadas propietarias) y las que cumplen un protocolo estándar, y por otro lado en las frecuencias de trabajo (las actualmente llamadas <1GHz, y las de 2.4GHz). Las <1GHz van desde 300 a 900MHZ (según las normativas en cada zona) y las de 2.4GHz que están normalizadas en todo el mundo, que a la vez definen velocidad de transmisión o ancho de banda y campo de aplicación.

Algo que hay que considerar en la radiofrecuencia es que el rango de trabajo depende de la frecuencia, la potencia de salida, la sensibilidad de recepción, de las características de la antena y del entorno de trabajo. Dentro de los tipos de tecnologías que se emplean para la radiofrecuencia tenemos la banda estrecha y la banda ancha la cual aprovecha todo el ancho de banda disponible en vez de utilizar una portadora para concentrar toda la energía a su alrededor como se muestra en la Fig 1-5.

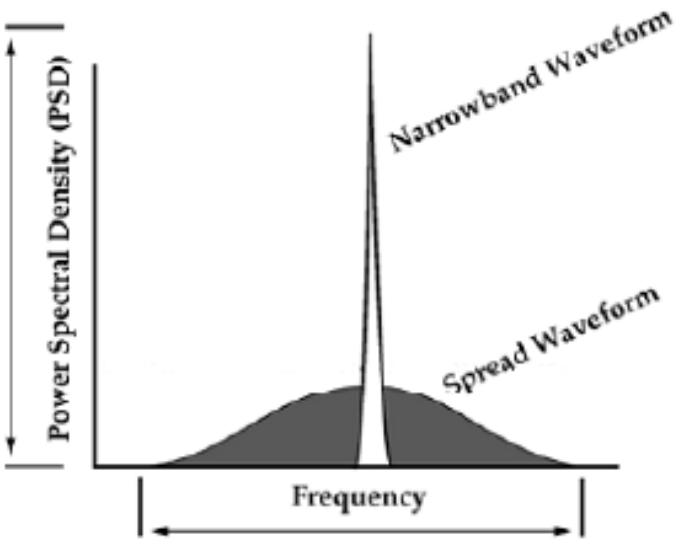


Fig 1-6 Banda estrecha y banda ancha

Bluetooth: se define como una tecnología inalámbrica cuyo propósito es el de reemplazar la conexión por cables en dispositivos portátiles como computadoras, mp3, celulares, entre otros, para conectarlos en una red local que es llamada Red de Área Personal o comúnmente PAN por sus siglas en inglés Personal Area Network.

Logo de Bluetooth

La palabra Bluetooth se originó alrededor del año 960 por el rey de Dinamarca Harald Blatand II Bluetooth. Sin embargo la tecnología fue inventada en 1994 por L. M. Ericsson, compañía sueca que desarrolla tecnología principalmente para los campos de la telefonía, telefonía móvil y comunicaciones multimedia. Más tarde en 1998 se formó el Grupo de Interés Especial (SIG); fundado inicialmente por Ericsson, Intel, IBM, Nokia y Toshiba. En la actualidad más de 1900 compañías se han unido al SIG.

Bluetooth opera en una banda no licenciada ISM (Industrial Scientific Medical) de 2.4-2.5GHz permitiendo la transmisión de voz y datos, de forma rápida y segura con un rango de hasta 10 metros con 1 mW o 100 metros si se usa un amplificador de 100 mW. Puede transferir datos de forma asimétrica a 721 Kbps y simétricamente a 432 Kbps.

Usa la llamada Frequency Hopping Spread Spectrum (FHSS), que divide la banda de frecuencia en un número de canales (2.402 – 2.480GHz, 79 canales). Para transmitir voz son necesarios tres canales de 64 Kbps, para transmitir vídeo es necesario comprimirlo en formato MPEG-4 y usar 340 Kbps para conseguir refrescar 15 veces por segundo una pantalla VGA de 320x240 puntos. Bluetooth minimiza la interferencia potencial al emplear saltos rápidos en frecuencia (1600 veces por segundo). Algunos de los elementos principales de una comunicación Bluetooth (Fig 1-6) son los siguientes:

- *Maestro (Master):* es el dispositivo Bluetooth que inicia la conexión, se encarga del control y de la temporización de los demás dispositivos. Todo esto se lleva a cabo en una red denominada Piconet.
- *Esclavo (Slave):* es cada uno de los dispositivos bajo las órdenes del Maestro. En una red Piconet se pueden tener hasta 7 esclavos.
- *Piconet:* es la red donde se lleva a cabo la conexión entre 1 maestro y hasta 7 esclavos.
- *Scatternet:* red formada por varias redes Piconet.

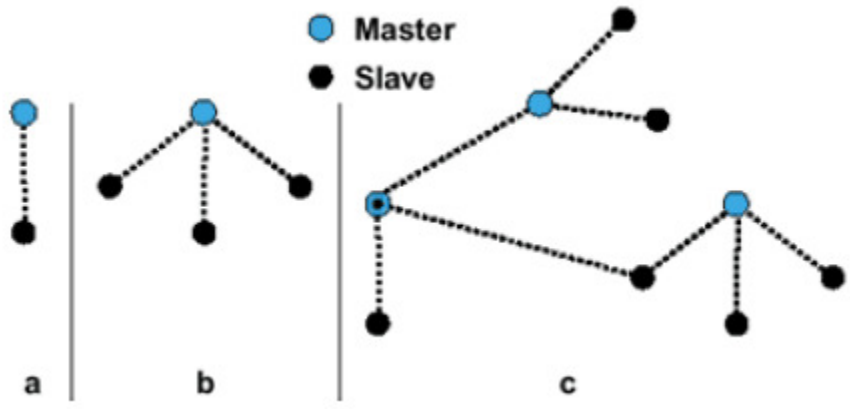


Fig 1-7 a) Piconet con un esclavo, b) Múltiples esclavos, c) Scatternet

A continuación se mencionan algunas de las ventajas que se tienen el utilizar una comunicación Bluetooth:

- Se puede tener una conexión de hasta 100 metros dependiendo la potencia del transmisor o bien utilizando repetidores. Esto le da la posibilidad de abarcar aplicaciones para telefonía celular y actuar como modem de computadoras.
- Bluetooth es capaz de de manejar simultáneamente la transferencia de datos y de voz.
- Facilita el descubrimiento y configuración de dispositivos lo que resulta en una accesibilidad sin la necesidad de un control explicito de direcciones de red, permisos y otros aspectos de las redes.

Wi-Fi: es una forma de conexión entre dispositivos electrónicos de manera inalámbrica. El Wi-Fi tuvo su origen cuando Nokia y Symbol Technologies crearon una asociación en 1999, la Alianza de Compatibilidad Ethernet Inalámbrica. Posteriormente en el 2003 pasó a ser la Wi-Fi Alliance, el objetivo de esta asociación es el de fomentar la tecnología inalámbrica y certificar la compatibilidad de equipos. Esto quiere decir que cada dispositivo y equipo que cuente con el sello de Wi-Fi garantiza la interoperabilidad entre ellos.

Logo de Wi-Fi

Para que se pueda realizar la conexión, se hace uso de un punto de acceso de red inalámbrica (WAP Wireless Access Point), que es un dispositivo que sirve para interconectar a otros dispositivos formando una red inalámbrica. El punto de acceso es el encargado de recibir, almacenar y enviar la información. Por lo general la distancia que cubre un solo punto de acceso es de 30 metros y puede soportar un pequeño grupo e usuarios.

La comunicación vía Wi-Fi utiliza ondas de radio similares a las que se utilizan en los walkie-talkies y celulares. Los dispositivos pueden enviar y transmitir información al convertir los 1's y 0's en ondas de radio y viceversa. Pero de igual forma presentan características que los diferencian de las demás ondas, algunas de ellas son las siguientes:

- Se transmite a frecuencias de 2.4 GHz o 5GHz. Estas frecuencias son más altas que las que se usan en celulares, televisiones y radios de comunicación, esto permite que se pueda transmitir una mayor cantidad de información.
- Wi-Fi puede transmitir en 3 bandas de frecuencia diferentes. Con esto es posible que se reduzca la interferencia cuando varios dispositivos se conectan a la misma red.

Dentro de los dispositivos Wi-Fi tenemos los dispositivos de distribución de red y los dispositivos terminales. Dentro de la primera clase además de los puntos de acceso se encuentran los routers, que son dispositivos compuestos diseñados para redes pequeñas. Se componen de un router (que es el encargado de interconectar las redes), un punto de acceso y de un switch que permite conectar equipos mediante cables.

Actualmente la tecnología Wi-Fi se enfrenta a problemas como la saturación del espectro radioeléctrico (causa interferencia y se pierde la conexión), redes que no cuentan con seguridad o redes abiertas (no protegen la información que circula por ellas) y finalmente la distancia que se cubre (variable dependiendo de los puntos de acceso).

Algunas de las ventajas que se tiene al utilizar una conexión vía Wi-Fi son que al ser una red inalámbrica se tiene una comodidad mayor que utilizando una cableada, cualquier persona dentro de un espacio definido se puede conectar, permiten el acceso a varias computadoras sin tener que gastar en infraestructura, al ser parte de la Wi-Fi Alliance te garantiza la compatibilidad entre los dispositivos sin importar el lugar del mundo donde uno se encuentre.

Dentro de las desventajas principales tenemos que la velocidad es menor a la de una conexión por cables debido a las interferencias que se puedan dar en el ambiente. Como antes se mencionó, la seguridad es una desventaja a considerar ya que en una red abierta es posible obtener la información que circula y finalmente aún está en desarrollo la compatibilidad entre Wi-Fi y otras conexiones inalámbricas como Bluetooth.

1.2.2 Ventajas y desventajas de tener una comunicación inalámbrica.

En el punto anterior se mencionaron tres formas de comunicación inalámbrica y algunas diferencias entre ellos. Ahora de manera general se enlistarán las principales ventajas y desventajas de usar una conexión inalámbrica para el proyecto.

Ventajas:

1. La comodidad de no tener que utilizar cables, lo que hace que el proyecto pueda ser manejado a una distancia que un cable no permitiría (movilidad).
2. Se pueden conectar varios dispositivos sin la necesidad de estar en un lugar físico específico.
3. Es más sencilla la instalación y configuración de una red inalámbrica.
4. No requieren de tanto mantenimiento como lo sería un sistema cableado.
5. Compartir información con varios los usuarios de la red o con un cierto grupo de personas es una tarea fácil.
6. Las ondas pueden atravesar barreras físicas de cierto espesor.

Desventajas:

1. Alcance limitado, es decir, que existen zonas en donde a señal no llega al usuario. El alcance principalmente depende de la potencia del punto de acceso, potencia del accesorio Wi-Fi con el que nos conectamos, que la señal no esté obstaculizada y las interferencias que el medio ambiente puede imponer.
2. La velocidad de transmisión es limitada dependiendo de cómo se comporte y se comparta la frecuencia. La velocidad máxima de transmisión es de 11Mbps aunque lo normal está entre 1,5 y 5Mbps para la 802.11b. Son valores más que suficientes para las necesidades del hogar y para las ofertas de todos los proveedores de Internet, sin embargo, las tecnologías cableadas (Ethernet en este caso) son potencialmente más veloces, con hasta 100Mbps, 1Gbps y más.
   El estándar 'IEEE 802.11' es el que define el uso de los niveles inferiores de la arquitectura OSI (capas física y de enlace de datos). Las ramificaciones entes mencionadas 802.11b y 802.11g definen la tecnología de redes de área local y redes de área metropolitana. 802.11b tiene una velocidad máxima de 11 Mbps funcionando en la banda de 2.4 GHz sin embargo la velocidad real con la que funciona debido a la codificación del protocolo CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) está entre 5.9 Mbps y 7.1 Mbps.

3. La seguridad para algunas conexiones inalámbricas es una desventaja. Es necesario encriptar la información para que no pueda ser robada.
4. El consumo de batería en caso de necesitarla es mayor que al utilizar cables.

1.2.3 Selección del tipo de comunicación inalámbrica para el proyecto.

Para el proyecto se decidió utilizar dos conexiones inalámbricas. La primera es la conexión vía Bluetooth, la cual se encargará de comunicarse con el microcontrolador enviando datos de los acelerómetros y recibiendo retroalimentación del micro. La segunda es la conexión Wi-Fi, la que se encargará de la transmisión y recepción de imágenes.

Se decidió de esta manera debido a que la velocidad de transmisión por Wi-Fi es mucho más rápida que por Bluetooth y en la parte de las imágenes se necesita que así sea. La parte de los datos que se envían de los acelerómetros no ocupa tal tamaño por lo que la conexión Bluetooth cumple perfectamente con lo que se desea, de igual forma para los datos que el microcontrolador procesa y envía al celular. Finalmente para facilitar el trabajo se piensa crear una red local (encriptada) entre dos teléfonos móviles, a manera que uno sea el router y el otro reciba las imágenes que este envía, y utilizar la conexión

Wi-Fi para que la imagen que se observe en el receptor sea lo más apegada al tiempo real. En otras palabras se aprovechará la velocidad de la conexión para que el retraso sea mínimo. A continuación se hace una tabla comparativa tanto de la conexión Bluetooth como de la conexión Wi-Fi.

	Bluetooth	**Wi-Fi**
Año de desarrollo	1994	1991
Autoridad	Bluetooth SIG	IEEE, WECA
Facilidad de uso	Fácil implementación. Es posible conectar hasta 7 dispositivos (7 esclavos), fácil para cambiar de dispositivo.	Implementación compleja y requiere de configuración en hardware y software.
Requerimientos de hardware	Adaptadores Bluetooth en los dispositivos que se quieran conectar.	Adaptadores inalámbricos en los dispositivos que se quieran conectar, routers o puntos de acceso.
Dispositivos principales	Teléfonos celulares, periféricos de la computadora (impresoras, mouse, teclados).	Computadoras, laptops, servidores.
Usos principales	Transmitir audio (como ejemplo del celular a los audífonos o estéreo).	Transmitir señales de la red (Internet) hacia los dispositivos.
Diferencias principales	Se usa para reemplazar cables entre dispositivos. La conexión se realiza uno a uno entre los dispositivos	Se utiliza para proveer acceso inalámbrico a una red local. Es posible conectarse con varios dispositivos mientras se envían datos.
Rango de cobertura	10 m pero puede aumentar usando repetidores por lo que en la literatura aparece como 10-100 metros.	100 metros
Ancho de banda/ Velocidad de transmisión	1 Mbps	11 Mbps con 802.11b y hasta 54 Mbps con 802.11g
Seguridad de transmisión	Más seguro al cubrir distancias más cortas y tener dos filtros de contraseñas.	La seguridad es vulnerable si se encuentra en una red abierta, al acceder a un poco de información es fácil acceder a la demás. Necesita encriptamiento.
Desventajas	Distancia de cobertura, velocidad.	Interferencia causada por cosas ajenas como estructuras y lugares cubiertos. Costo. Seguridad

Ventajas	Seguridad, fácil implementación.	Distancia de cobertura, velocidad, fácil interacción con dispositivos.
Consumo de batería	Bajo	Alto
Frecuencia de trabajo	2.4 GHz	2.4 GHz

1.3 Compañías que se dedican al tema últimos avances.

Como es de notar, el desarrollo de aplicaciones para celulares con plataforma Android sigue avanzando. No es de sorprenderse en las similitudes que se han generado entre Apple (Iphone) y Google (Android), como lo es el control total del dispositivo por voz, la interacción entre el dispositivo y el usuario y finalmente el diseño totalmente táctil.

La innovación de nuevos proyectos como el que se propone en este trabajo no se ha hecho esperar, la interacción entre teléfonos móviles con otras áreas de conocimiento, en este caso la electrónica, es uno de los campos con más impacto actualmente. Sin embargo se mencionará a continuación los productos más parecidos a este proyecto junto con sus características, diferencias y similitudes.

1.3.1 Proyectos que se han desarrollado.

a) BeeWi:

La compañía *BeeWi Simply Wireless* se dedica a ofrecer productos con la última tecnología en Wi-Fi y Bluetooth en Europa. Esta compañía tiene un vehículo a escala, copia de un Mini Cooper, controlado por smarthphones (incluye la plataforma Android). La transferencia de datos se lleva a cabo mediante Bluetooth, lo que limita su espacio de trabajo entre 10-15 metros.

Tiene dos formas de controlarlo, la primera es usando la pantalla táctil (Fig 1-7) de manera que mediante presionar la pantalla se indica la dirección y sentido de giro de las llantas. Por su parte; la segunda forma es usando los acelerómetros del teléfono móvil para indicar sentido de giro y dirección (Fig 1-8).

Fig 1-8 BeeWi junto a la interfaz para el control mediante la pantalla táctil.

Fig 1-9 Interfaz para el control usando los acelerómetros (Motion Control)

Este vehículo utiliza tres baterías AA, es compatible con versiones iguales o superiores a Android 2.1 y hace uso de servomotores para la dirección. Es necesario descargar la aplicación para el dispositivo móvil desde el Android/Market, no tiene ningún costo (Fig 1-9).

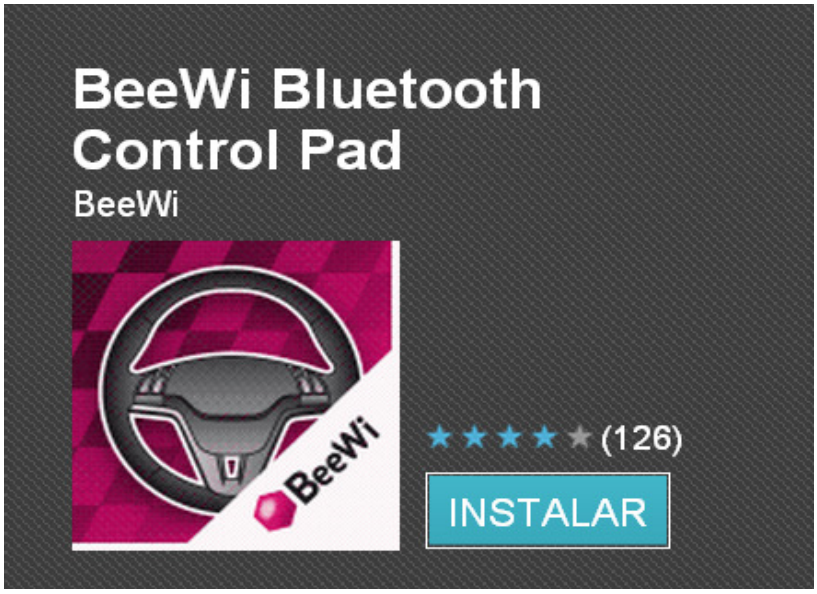


Fig 1-10 Aplicación para BeeWi en Android/Market.

b) Blue-Drone:

Es el nuevo lanzamiento de la compañía de mismo nombre. Es bastante similar al BeeWi a diferencia de que no es un Mini Cooper. Tiene las mismas características que el producto anterior, las dos formas de controlarlo (táctil y con los acelerómetros), la misma duración de las baterías (3 pilas AA para 3 horas), el mismo rango de control (10-15 metros) y de igual forma la transmisión de datos es vía Bluetooth.

Si se hace la comparación con el BeeWi, se notará que no existen diferencias además del modelo del vehículo, es un producto igual pero con diferente marca. Incluso se utilizan los mismos motores lo que hace que el control del vehículo sea completamente idéntico al del BeeWi. En las siguientes imágenes (Fig 1-10) se muestra el producto así como las acciones que promociona.

Fig 1-11 Blue-Drone junto con las actividades que promociona para su lanzamiento.

c) Ar.Drone:

La compañía Parrot, fundada en 1994, se dedica a la creación y desarrollo de dispositivos controlados por teléfonos celulares. Es una compañía grande, incluso cuenta con las siguientes certificaciones en calidad: ISO 9001, ISO TS 16949 e ISO 14001. El proyecto a analizar es el Ar.Drone. Es una cuadricóptero controlado por teléfonos móvil incluyendo plataforma Android. Sin duda alguna es el invento con más tecnología y desarrollo actualmente (Fig 1-11).

Los controles y la transmisión de video se llevan a cabo por conexión WI-FI, tiene control táctil y de igual forma utilizando los acelerómetros. Para la seguridad del dispositivo se cuenta con una pantalla de emergencia en el celular y sensores de contacto para bloquear las hélices. Cuenta con 2 cámaras, horizontal y vertical con una cobertura de 93º y 64º respectivamente.

Tiene sensores de distancia como el altímetro (6 metros), sensores de posición como los acelerómetros y el girómetro, finalmente cabe mencionar que tiene piloto automático y estabilizador automático también. Sin duda es el producto más completo de nuestros días, la batería es de litio y tiene una duración de 12 minutos en el aire, tiempo razonable tomando en cuenta todo lo que contiene.

Fig 1-12 Ar.Drone de Parrot.

El Ar.Drone está construido de fibra de carbón y plástico de alta resistencia PA66. Tiene tecnología MEMS (Micro-electro-mechanical-systems). Los motores que utiliza son de tipo brushless (4 motores), finalmente es importante conocer que la resolución es VGA (640x480 pixeles).

1.3.2 Comparación con nuestro proyecto.

	COMPARACIÓN ENTRE LOS PROYECTOS EXISTENTES Y EL PROYECTO A DESARROLAR			
	BeeWi	Blue Drone	Ar.Drone	Proyecto a desarrollar
Función principal	Controlar un coche mediante el celular Android. Se controla la dirección y sentido de giro de los motores con los acelerómetros del celular. De igual forma se puede utilizar la pantalla tactil.	Controlar un coche mediante el celular Android. Se controla la dirección y sentido de giro de los motores con los acelerómetros del celular. De igual forma se puede utilizar la pantalla tactil.	Controlar un cuadricóptero utilizando un celular Android. Se controla la velocidad, dirección y altura del vehículo. Se tiene transmisión de video tanto horizontal como vertical. En la pantalla se despliegan datos que envían los sensores en el cuadricóptero. Puede ser un vehículo áereo de exploración, espía o de diversión.	Controlar un vehículo utilizando un celular android. Se controla la velocidad, dirección y sentido de giro. Se tiene transmisión gráfica de lo que a cámara en el vehículo ve. En la pantalla del controlador se ven las imágenes y datos sobre el estado del teléfono (sensores internos y externos). Puede ser un vehículo terrestre de exploración, espía o de diversión.
Tipo de conexión para envío de datos	Bluetooth	Bluetooth	Wi-FI	Bluetooth (microcontrolador), Wi-FI (imágenes).
Tipo de motores	Motores de CD y servos para la dirección.	Motores de CD y servos para la dirección.	Brushless	Motorreductores de CD.
Dispositivo de control	Celular con plataforma superior a Android 2.1	Celular con plataforma superior a Android 2.1	Celular con plataforma superior a Android 2.1. IOS.	Celular con plataforma superior a Android 2.1
Sensores internos	NO	NO	SI. MEMS.	SI. Sensor de temperatura de la etapa de potencia.
Sensores externos	NO	NO	SI. Altímetro ultrasónico. Girómetros. Sensores de distancia. Sensores de choque. Estabilizadores.	SI. Sensor de temperatura ambiente. Tacómetro. Sistema de iluminación.
Sesores de control en el dispositivo	Acelerómetros y pantalla tactil.	Acelerómetros y pantalla tactil.	Acelerómetros, pantalla táctil, giroscopios.	Acelerómetros y pantalla tactil.
Control de velocidad	NO	NO	SI. Dependiendo de la posición de los sensores en el controlador.	SI. Dependiendo de la posición de los acelerómetros se controla la velocidad usando PWM.
Retroalimentación entre el microcontrolador y el dispositivo controlador	NO	NO	SI. Transmisión de video e información de los sensores externos.	SI. Transmisión de imágenes e información de los sensores internos y externos.
Baterías	3 pilas AA	3 pilas AA	Litio	Batería recargable de 9.6 v
Cámara integrada	NO	NO	SI. Cámara horizontal y vertical	SI. Cámara de un celular que funje como router.
Interfaz gráfica	SI. Aplicación gratis.	SI. Aplicación gratis.	SI. Aplicación gratis.	SI. Aplicación gratis.

CAPÍTULO 2. ESPECIFICACIONES DEL PROYECTO

Para que quede claro el alcance de este proyecto hay que definir algunas cosas. Primero hay que establecer que es lo que se espera que el proyecto cumpla, es decir, las necesidades que tenemos como clientes o usuarios. Después hay que mencionar que requerimientos técnicos se necesitan para lograr que las necesidades del usuario se satisfagan. Finalmente hay que especificar que materiales se utilizarán para poder terminar de establecer limitantes, ventajas, desventajas y condiciones de operación del vehículo de exploración.

2.1 Necesidades del usuario.

Si se analiza el proyecto desde el punto de vista de usuario, hay necesidades que son indispensables para el éxito de este proyecto, hay otras que no son críticas y algunas que lo pueden mejorar o son de confort. Se enlistarán las necesidades que el proyecto debe cumplir y se establecerá una jerarquización para cada una de ellas.

NECESIDADES CRÍTICAS:

- Que el vehículo avance hacia adelante, atrás, derecha e izquierda.
- Que al controlarlo se puedan hacer movimientos suaves, es decir, que no haga giros bruscos.
- Que el control se haga por conexión Bluetooth.
- Que el microcontrolador provea de información al usuario sobre el estado del vehículo (estado de carga de la batería, temperatura de la etapa de potencia).
- Que las imágenes que tome la cámara del vehículo sean enviadas con éxito al controlador.
- El dispositivo que se coloque en el vehículo para capturar las imágenes debe tener la capacidad de realizar una conexión tether. Tethering es el proceso mediante el cual el dispositivo móvil funge como un modem o router, permitiendo que otros dispositivos se conecten a él y tengan acceso a los datos suministrados por la red.
- Que el envío de imágenes sea por conexión Wi-Fi.
- Que el celular controlador indique si los dispositivos están conectados.
- Que al perder la conexión el vehículo se detenga.

OTRAS NECESIDADES:

- Que el microcontrolador obtenga datos como la temperatura ambiente, la velocidad a la que se mueve y la distancia recorrida.
- Que los motores del vehículo tengan un buen torque o buena reducción para evitar que se atasque.
- Que el vehículo este construido con materiales adecuados para evitar rupturas o desgaste prematuro.

NECESIDADES DE CONFORT:

- Que la aplicación para controlar el vehículo sea en plataforma Android, ya que al ser software libre se puede distribuir y modificar sin problemas.
- Que el vehículo tenga dispositivos de iluminación por si se encuentra en un medio oscuro.
- Que la el diseño exterior del vehículo sea estético y agradable a la vista.
- Que el control se haga con los acelerómetros del celular y se maneje como si fuera un volante.
- Que al ver el estado del vehículo se detenga el envío de datos de los acelerómetros para evitar accidentes.
- Que cualquier dispositivo con plataforma Android superior a la versión 2.1 pueda manejar el vehículo.

2.2 Requerimientos técnicos.

REQUERIMIENTOS TÉCNICOS	
NECESIDAD	*REQUERIMIENTO*
Movilidad hacia adelante, atrás, derecha e izquierda.	Motores de corriente directa CD, microcontrolador.
Sensibilidad al controlarlo.	Aplicación Android.
Control inalámbrico.	Conectar el microcontrolador con el dispositivo controlador por Bluetooth.
Información sobre el estado del vehículo.	Microcontrolador.
Temperatura etapa de potencia y ambiente.	Sensor de temperatura.
Imágenes que tome el vehículo.	Cámara en el coche.

Envío de imágenes por Wi-Fi.	Conectar la cámara y el dispositivo controlador por Wi-Fi.
Que se indique si la conexión se estableció.	Aplicación Android.
Detener el vehículo al perder conexión.	Microcontrolador.
Velocidad y distancia recorrida.	Sensores IR y fototransistores.
Buen torque.	Motoreductores de CD con reducción 1:48.
Materiales adecuados.	Diseño del vehículo, CAM.
Llantas resistentes.	Llantas
Aplicación controladora en plataforma Android.	Celular Android para controlar el vehículo.
Dispositivo de iluminación.	LEDS
Diseño exterior estético.	CAD
Control con acelerómetros.	Aplicación Android y celular con acelerómetros.
Detener el vehículo para ver el estado del teléfono.	Aplicación Android.
Cualquier dispositivo Android puede ser controlador.	Aplicación Android.

2.3 Materiales.

MATERIALES		
Requerimiento	*Elemento*	*Características*
Motores de corriente directa	Motoreductores	Motores de corriente directa con reducción de 1:48 para evitar frenarse con piedras y todo tipo de pisos. Consumo de corriente sin carga 80 mA y atrancado 600mA.
Módulo Bluetooth	Bluetooth Modem - BlueSMiRF Gold	Tiene un alcance de 106 metros. El voltaje de alimentación está entre 3-6 V. Consumo de energía bajo 25mA. Funciona en ambientes Wi-Fi, 802.11g y Zigbee. Cuenta con una conexión encriptada. Antena superficial. Frecuencia de operación de 2.4-2.5 GHz.

Microcontrolador	ATMEGA 328P-PU	Microcontrolador de 8 bits con 32K Bytes en sistema y memoria flash. 2Kb en RAM. 23 pines para propósitos generales. 32 registros de trabajo. Tiene una arquitectura RISC. Convertidor Analógico Digital de 10 bits de resolución. Funciona de -40 a 85º C. Tiene memoria EEPROM de 1 Kb.
Celulares con plataforma Android	Sony Ericsson Xperia X10 mini	Plataforma Android 2.1. Cámara de 5 MP con grabador de video. Controles con pantalla táctil. A-GPS integrado. Conexiones inalámbricas Wi-Fi, HSPA y Bluetooth 2.1 con A2DP. Procesador Qualcomm de 600 MHz.
	Samsung i5500 Galaxy 5 GT-I5500L	Plataforma Android 2.2. Controles con pantalla táctil. Cámara de 2 MP con grabador de video. GPS integrado. Conexión Wi-Fi 802.11 b/g. Bluetooth 2.1. Microprocesador de 600 MHz.
Sensor de temperatura	LM35DZ	Es un circuito integrado de precisión para temperatura en grados Celsius. Es un sensor linear que aumenta 10mV por cada grado centígrado. Su rango es de -55 a 150º. Su voltaje de operación es de 4 a 30V.
Puente H para los motores de CD	L293d	Controlador para 4 medios puentes H. Alimentación de entradas lógicas por separado. Protección para descargas electrostáticas. Corriente de salida de 1 A por canal, 2 A en total.

CAPÍTULO 3. DISEÑO DEL CIRCUITO ELECTRÓNICO PARA LA OBTENCIÓN Y TRANSFERENCIA DE DATOS ENTRE EL MICROCONTROLADOR Y EL DISPOSITIVO MÓVIL.

Una de las partes críticas del proyecto, es el funcionamiento y la interacción de los componentes electrónicos (circuito electrónico). Como se ha mencionado; el dispositivo controlador será el encargado de enviar los datos provenientes de los sensores del celular (acelerómetros) al microcontrolador, quien a su vez los procesará e indicara que acciones son las que se deben de llevar a cabo.

La transferencia de datos (envío y recepción) se hará mediante una conexión Bluetooth. En este capítulo se mencionan puntos como la selección de componentes, la implementación de la electrónica, el diseño del PCB y los códigos del microcontrolador que funcionaron como pruebas hasta obtener el resultado final.

3.1 Selección de componentes y funcionamiento.

El primer paso para diseñar el circuito electrónico es la selección de los componentes que se utilizarán. Para ello se hizo uso de las tablas previamente establecidas en este trabajo y de la tabla de materiales. Pero en este punto se analiza a detalle cada uno de los componentes contenido en la tarjeta electrónica.

Controlador de los motores (puente H): L293d (Fig 3-1).

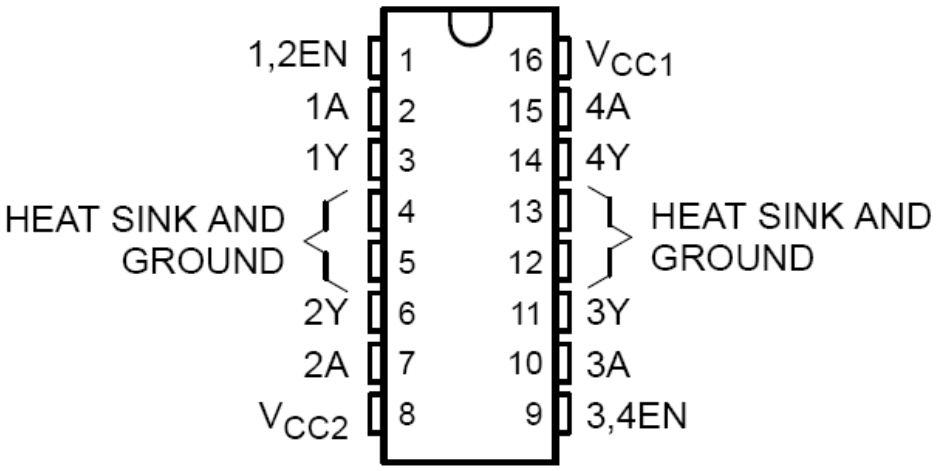


Fig 3-1 Esquema de conexiones del L293d.

Como se observa, el controlador es para dos motores de corriente directa. El voltaje que es para el motor (pin 8) es el mismo que entrega la pila recargable que se usará. Los demás pines necesarios para el funcionamiento del motor se controlaran con el microcontrolador.

Regulador de voltaje (5V)): KA-78R05 (Fig 3-2).

Fig 3-2 Esquema de conexiones del KA-78R05.

Para la alimentación del circuito electrónico así como del microcontrolador, es necesario regular el voltaje a 5V, ya que con este valor funcionan los circuitos digitales. Para ello se hace uso del KA78R05; este regulador a diferencia de otros similares como el 7805 tiene una mejor estabilidad en la salida, es decir, no tiene una variación significativa con valores cercanos a los 5 V. De igual forma tiene protección contra picos de voltaje, de corriente, se apaga si la temperatura sube demasiado y tiene un pin para deshabilitarlo.

Sensor de temperatura: LM35DZ (Fig 3-3).

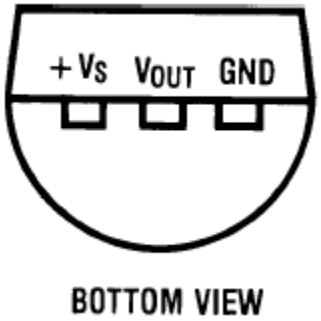


Fig 3-3 Esquema de conexiones del LM35DZ.

Para la tarea de sensar la temperatura ambiente y de la etapa de potencia se utilizará sensores LM35DZ. Son sensores de precisión cuyo voltaje de salida es proporcional a la escala Celsius. Es un sensor que no requiere de calibración externa, funciona de -55 a 150ºC y tiene una impedancia baja a la salida.

Diodo emisor IR y fototransistor: IR383_HT y PT1302BC2 (Fig 3-4).

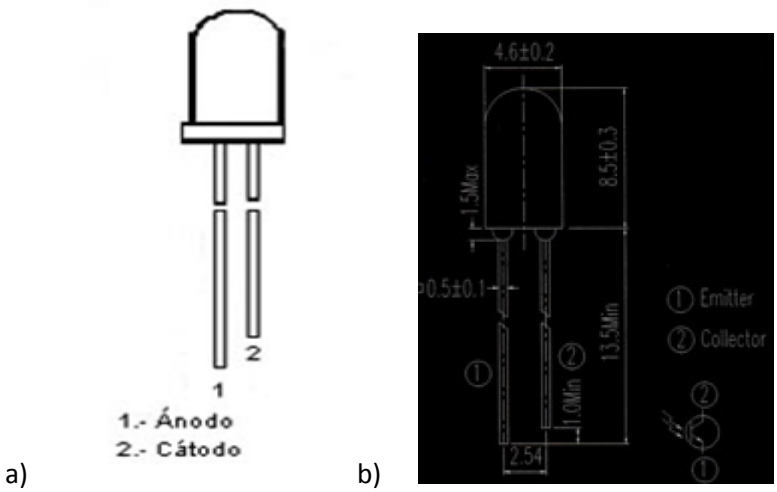


Fig 3-4 a) IR383_HT (Diodo emisor IR) b) PT1302BC2 (fototransistor)

Este diodo IR y el fototransistor con los elementos que se utilizan para sensar la velocidad, revoluciones y distancia recorrida de los motores del proyecto. Se hace uso de las llantas como encoders, los cuales generan un tren de pulsos de cierto periodo y frecuencia al interrumpir la incidencia de la luz infrarroja en el fototransistor. Se utilizo el IR383_HT porque el haz de luz infrarroja que emite es de 12º de apertura y el elemento posee un filtro de luz ambiente. Con este filtro se minimiza la interferencia o ruido causado por luz externa.

El microcontrolador se encrga de hacer los cálculos para el periodo y con las medidas de las llantas de la velocidad y distancia. Se decidió utilizar estos elementos por la forma de la llanta y porque al hacer pruebas con otros sensores como el CNY70 y el QRD1114 se presentaban lecturas incorrectas. Esto se puede deber a la superficie de refracción (color y textura). Otra cosa a considerar es que al ser de incidencia; se pueden colocar de una manera sencilla alrededor de la llanta del vehículo simulando una herradura.

Amplificador Operacional: LM324 (Fig 3-5).

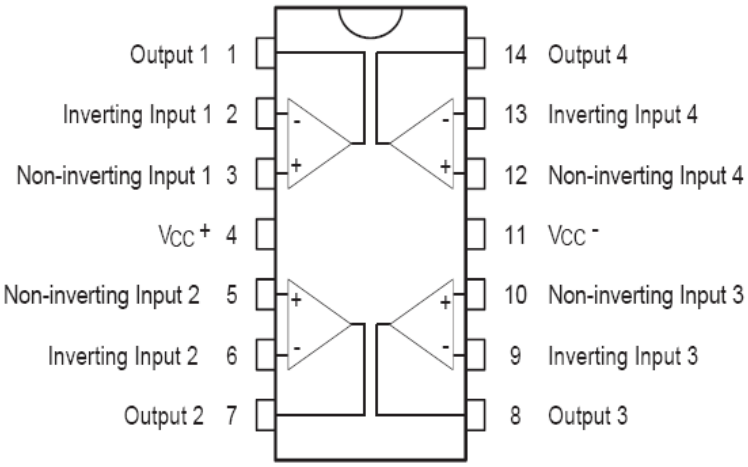


Fig 3-5 Diagrama de conexiones LM324.

Uno de las principales características del amplificador operacional es la de eliminar efectos de la carga o acoplar impedancias, es decir, te permite conectar un dispositivo con gran impedancia a uno con baja impedancia y viceversa. En este proyecto, las salidas de los sensores de temperatura y de la pila (para obtener el estado de carga después del divisor de voltaje) se conectan a un seguidor de voltaje (Fig 3-6) cuya entrada es igual a la salida pero obtiene las ventajas antes mencionadas. Con ello se evita ruido e interferencia debido a la corriente al momento de procesar los datos en el microcontrolador.

El amplificador operacional se alimentará con la pila recargable para asegurar que la salida del seguidor brinde los 5 V necesarios. Se utiliza el LM324 porque se necesitan tres seguidores y este dispositivo contiene cuatro; se minimiza el espacio para la elaboración de la tarjeta final. De igual forma porque te permite que el voltaje de saturación negativo sea igual a 0 V.

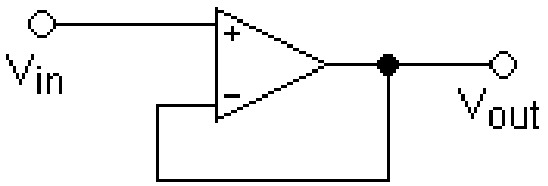


Fig 3-6 Seguidor de voltaje Vin=Vout.

3.2 Implementación del microcontrolador ATMEGA 328P-PU.

Como se mencionó previamente, el microcontrolador que se utilizará es el ATMEGA 328P-PU, para su implementación es necesario conocer su diagrama de conexiones (Fig 3-7). Con el diagrama de conexiones nos damos de los pines con los que se cuentan, de cuales necesita el dispositivo para funcionar y de cuales podemos utilizar.

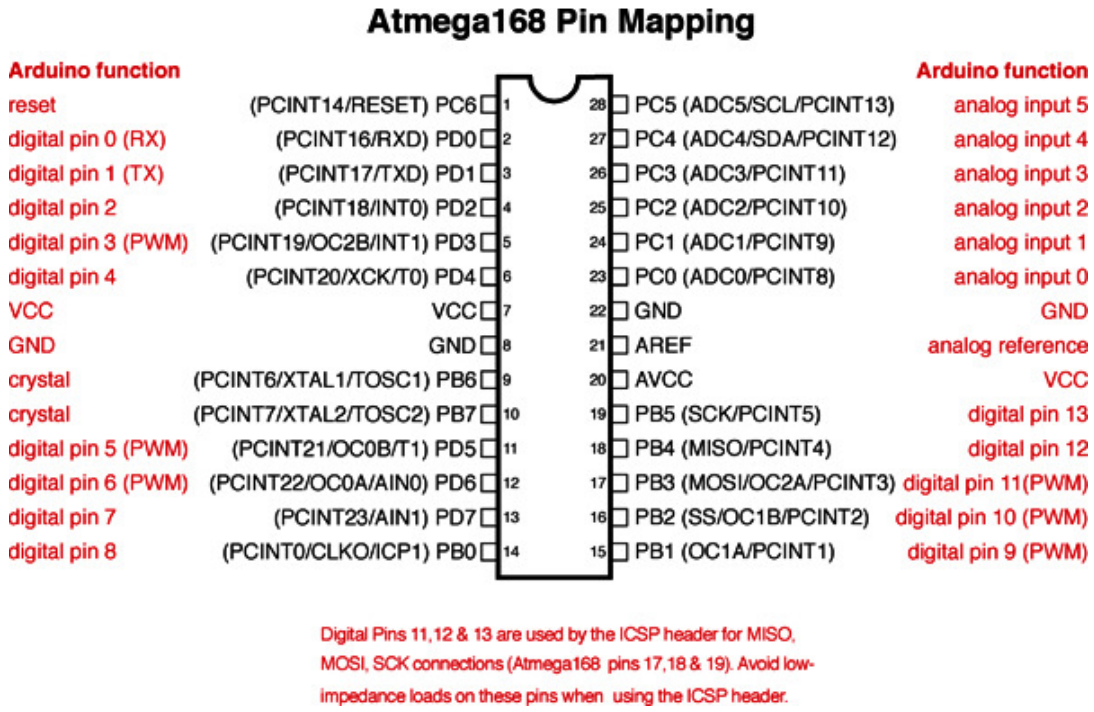


Fig 3-7 Diagrama del ATMEGA 168, similar al ATMEGA 328

Por lo general este dispositivo se asocia con lo que es la tarjeta Arduino Uno o bien la anterior Arduino Duemilanove, sin embargo para este proyecto no es necesario contar con ellos puesto que el microcontrolador no será reprogramado constantemente. Los únicos casos donde se tendrá que reprogramar es, si se decide aumentar el número de sensores, de motores o demás fuentes de información para procesar. De igual forma el espacio geométrico es un elemento a considerar, si se utilizará el microcontrolador con la tarjeta de Arduino el espacio aumenta considerablemente. Por ello se decidió utilizar el puro microcontrolador con los elementos necesarios para su funcionamiento (cristal de 16 MHz., reset, alimentación externa, referencias analógicas).

Se utiliza el cristal de 16 MHz, externo debido a que el reloj interno es poco preciso y no alcanza a trabajar de buena manera con el BLUESMIRF. Si se utiliza el reloj interno como resultado se obtiene una comunicación nula, no se logran leer los datos que envía el BLUESMIRF y el microcontrolador no puede procesarlos. El mejor resultado se obtuvo con el cristal externo de 16 MHz y una velocidad de 57600 bps en el microcontrolador.

Con estas consideraciones es posible diseñar una tarjeta para circuito impreso (PCB) en donde se contenga a los elementos electrónicos como son el puente H, amplificador operacional, microcontrolador, parte de alimentación, parte de regulación de voltaje y lo conectores necesarios para los sensores externos del vehículo.

3.3 Programas desarrollados durante el proyecto para el ATMEGA 328P-PU.

Es necesario mencionar los programas que se desarrollaron para poder ir probando las diferentes etapas del proyecto. Para ello dividimos el proyecto en 5 etapas que lo relacionan con el microcontrolador, etapas críticas para el buen funcionamiento del proyecto. Los códigos se encuentran comentados línea por línea en la parte de anexos de este trabajo.

1. Adquisición de datos provenientes del dispositivo móvil por Bluetooth e imprimirlos en la pantalla.
2. Movilidad y control de los motores usando los datos provenientes del dispositivo móvil por Bluetooth (PWM).
3. Adquisición y cálculo de la temperatura con los datos analógicos del sensor LM35DZ.
4. Adquisición y cálculo del estado de la batería con los datos analógicos provenientes del divisor de voltaje.
5. Adquisición y cálculo de la velocidad, revoluciones y distancia recorrida por el vehículo utilizando interrupciones.

Una vez con estas pruebas de pudo llegar al código final; el cual será programado en el microcontrolador del prototipo funcional. De igual manera este código de encuentra en la parte de anexos y de explica de mejor manera la parte dedicada a los resultados del proyecto.

3.4 Implementación del dispositivo Bluetooth BLUESMIRF GOLD.

El módulo que se utiliza para la recepción de datos provenientes del dispositivo móvil y su envío al microcontrolador es el BLUESMIRF GOLD (Fig 3-8). En el capítulo 2 se mencionaron sus características principales. En este punto se describirá la forma en que interactúa con el microcontrolador.

Fig 3-8 BLUESMIRF GOLD

En la página web del microcontrolador ATMEGA 328P-PU se describe la forma en que debe conectarse el módulo. Un aspecto a resaltar es que el pin TX (transmisor) del módulo se conecta el RX (receptor) del microcontrolador y pos u parte el TX del microcontrolador se conecta al RX del módulo Bluetooth. Mientras que los pines de alimentación se manejan de manera normal.

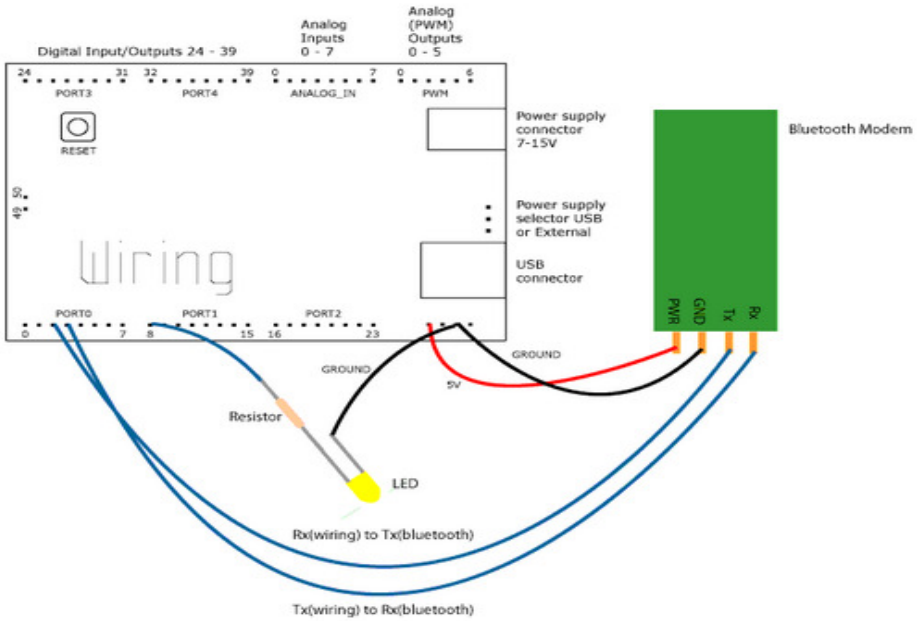


Diagrama para interactuar con el ARDUINO.

Este dispositivo se conectará al ATMEGA 328P-PU el cual procesará los datos que reciba el módulo Bluetooth. La velocidad a la que funcionó mejor el microcontrolador junto con el módulo Bluetooth fue la de 57600 bps. Si se usaban velocidades inferiores se perdían algunos datos y se representaba en una respuesta lenta; como estamos manejando velocidades con los acelerómetros del controlador Android, es necesario que no se pierdan datos para que la respuesta sea lo más cercana al tiempo real.

Otro dato a considerar es que la aplicación Android tendrá un indicador de conexión, es decir, mediante la aplicación se indicará cuando la conexión este establecida correctamente, el módulo indica mediante un led (que cambia de rojo/apagado a verde/encendido), por su parte el celular indicará de igual forma que los dispositivos están enlazados.

3.5 Alimentación del circuito.

Dentro de la etapa de alimentación del circuito, tenemos dos partes. La primera es la alimentación para el microcontrolador, el amplificador y los sensores. La segunda es para el puente H l293d en su Vcc2 (motores del vehículo) y para obtener el estado de carga de la batería. Se decidió usar fuentes independientes para evitar ruido y que el consumo de batería fuera exagerado.

Alimentación para el microcontrolador y sensores:

El ATMEGA 328P-PU se alimenta con 5V al igual que el puente H y los sensores (temperatura e infrarrojos). Se utilizará una pila cuadrada de 9V y un regulador KA78R05 para que a la salida se tengan 5V. Se utilizan condensadores en los pines de entrada y salida para estabilidad y filtro de ruido. La forma de conectarlo se muestra en la Fig 3-9 mediante el apoyo de Proteus.

Por su parte el amplificador operacional, como se mencionó anteriormente, funge con la tarea de un seguidor de voltaje par acoplar las impedancias. Como el amplificador operacional sufre pérdidas por la alimentación de los transistores internos, el voltaje de saturación positivo debe ser superior a los 5V por lo que se utiliza el mismo de la pila de 9V.

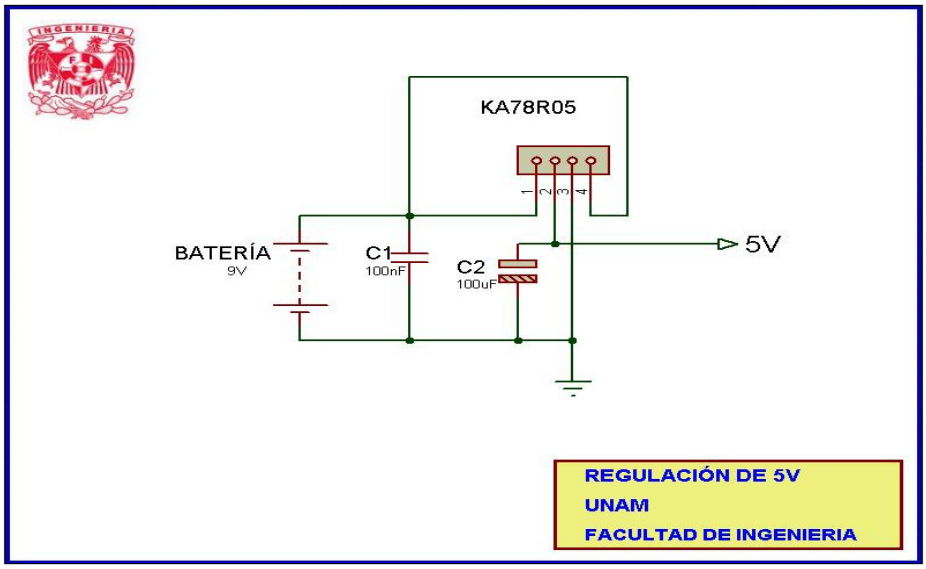


Fig 3-9 Regulación a 5V para alimentación del circuito.

Alimentación para los motores y el amplificador operacional:

El voltaje para la alimentación de los motores es de la pila recargable para juguete de radiocontrol (9.6V). Este voltaje se conecta al pin 8 del puente H l293d de manera directa y a su vez entra en un divisor de voltaje para que la salida del mismo pueda ser procesada por el microcontrolador. El divisor de voltaje se emplea para que el voltaje máximo de la pila no rebase los 5V a la hora de procesarlo por el microcontrolador. La forma de conectarlo se muestra en la Fig 3-10 mediante el apoyo de Proteus.

Un dato importante a considerar es que al caracterizar la pila de 9.6V se obtuvieron datos fuera de lo común. El voltaje máximo que la pila entrega es de 11.3V mientras que el mínimo para que el proyecto siga operando normalmente es 8.5V. Los cálculos tanto del divisor de voltaje como de la función para el microcontrolador se hicieron tomando en cuenta estos valores.

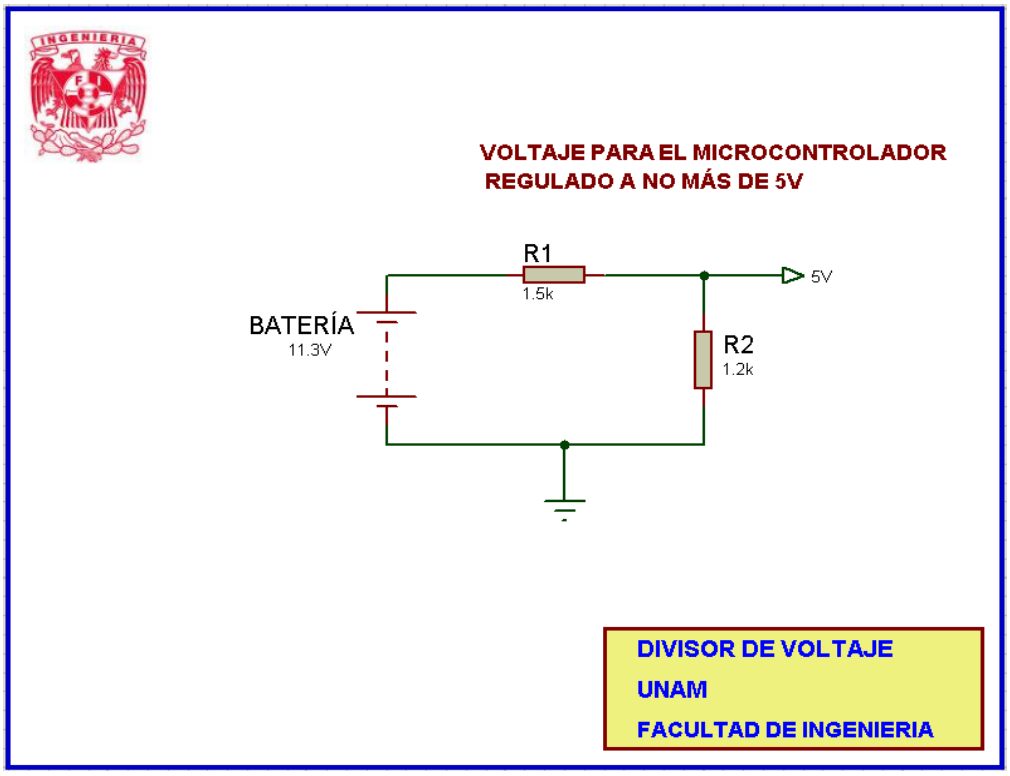
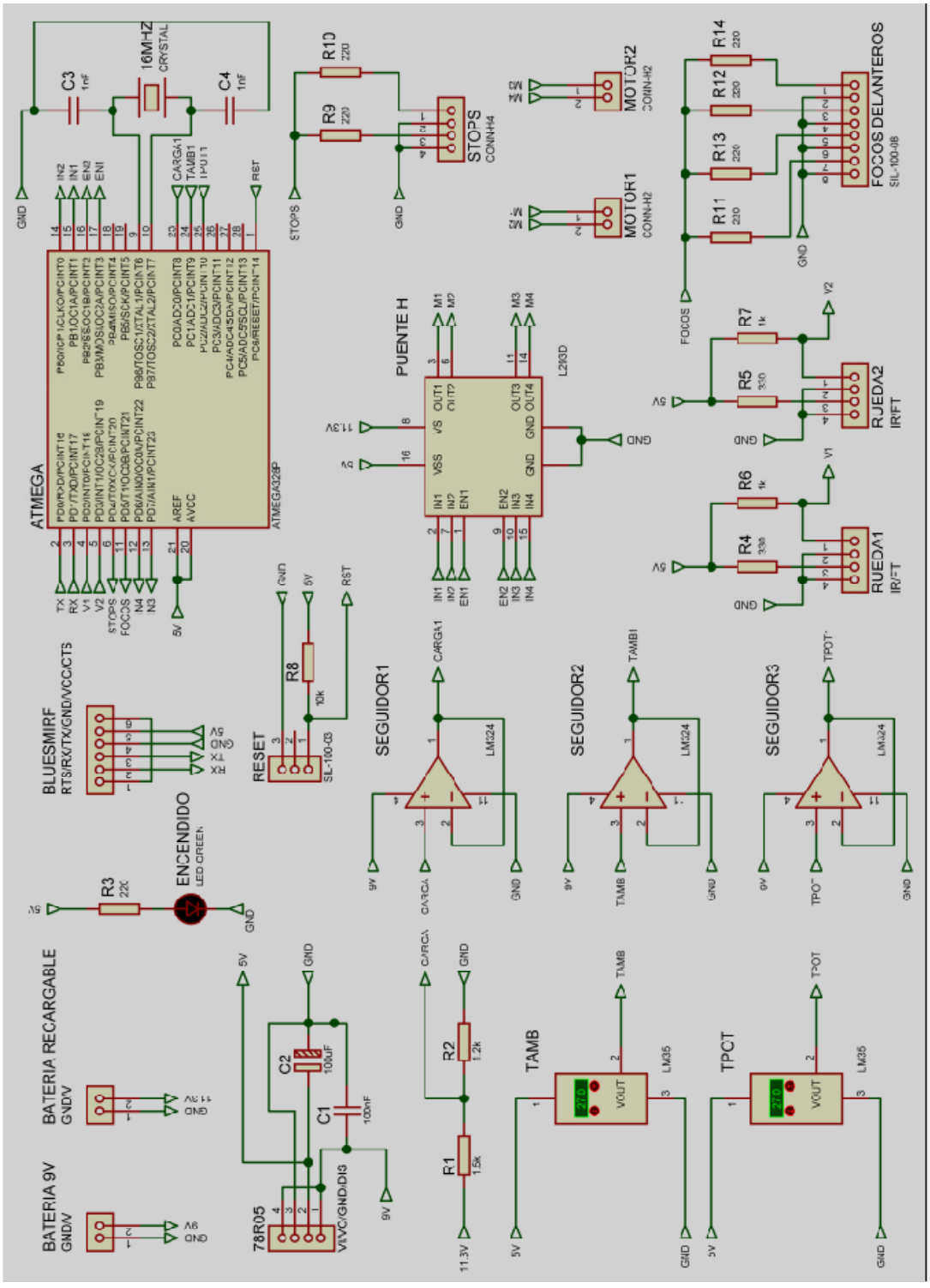


Fig 3-10 Divisor de voltaje para evitar que la salida sea mayor a 5V.

3.6 Esquema de la conexión del circuito electrónico.

En el siguiente esquema se muestra la conexión general del circuito electrónico para el vehículo explorador de este proyecto. De igual forma se muestran la configuración de las entradas, salidas del microcontrolador y demás dispositivos electrónicos.



3.7 Diseño del PCB del circuito electrónico.

Tomando en cuenta el esquema mostrado en el punto anterior, se diseñará un circuito impreso (PCB Printed Circuit Board). Se tomó la decisión de hacerlo de esta manera para optimizar espacio. Después de hacer varias pruebas en el software ARES de Labcenter Electronics (diseño de circuitos impresos), se llegó a la conclusión de que la mejor manera de optimizar espacio era hacer el circuito a doble cara y con elementos de montaje superficial. Sin embargo en el caso de nuestro prototipo del vehículo de exploración; por practicidad se realizó el circuito con las pistas en una sola cara y de manera modular, obteniendo dos módulos. El primero de ellos (Fig 3-11 y Fig 3-12) contiene los conectores para la batería de 9V, la batería recargable y la entrada para los sensores infrarrojos de cada llanta. De igual manera contiene la regulación de voltaje con el 78R05, el divisor de voltaje para medir el estado de carga de la batería recargable, un led para indicar que el circuito está alimentado, la entrada de los leds para los focos delanteros y los traseros y finalmente contiene a manera de salida los voltajes y sensores necesarios para el otro módulo.

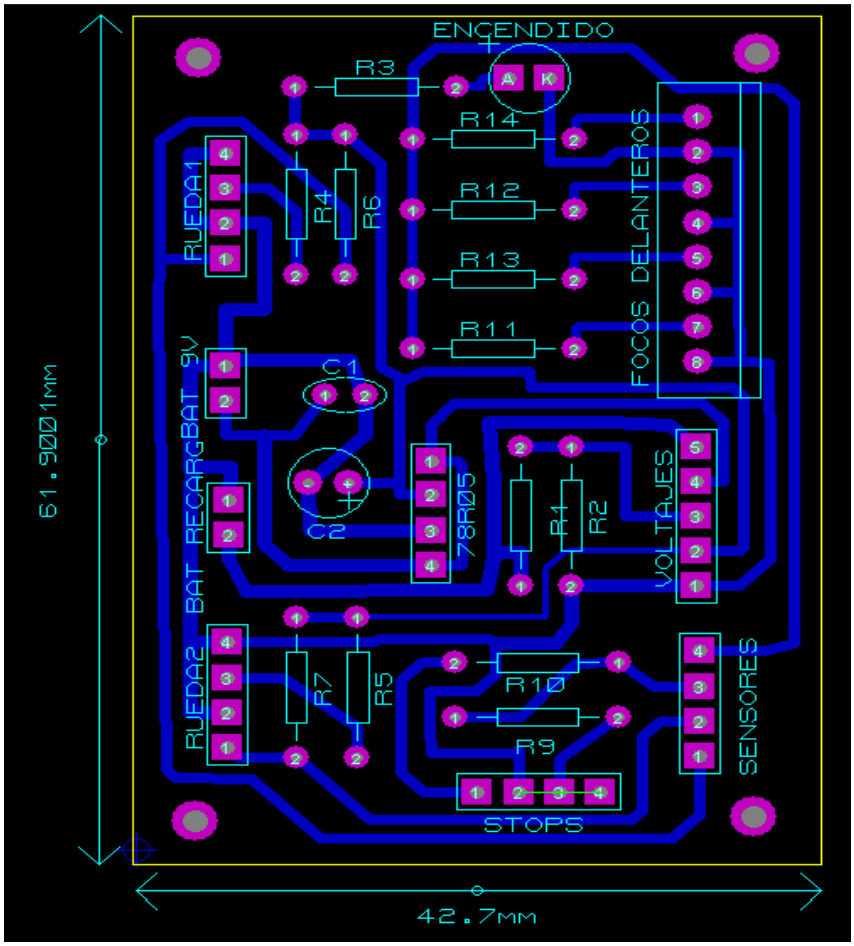


Fig 3-11 Diseño del primer módulo del circuito en una sola cara, con medidas.

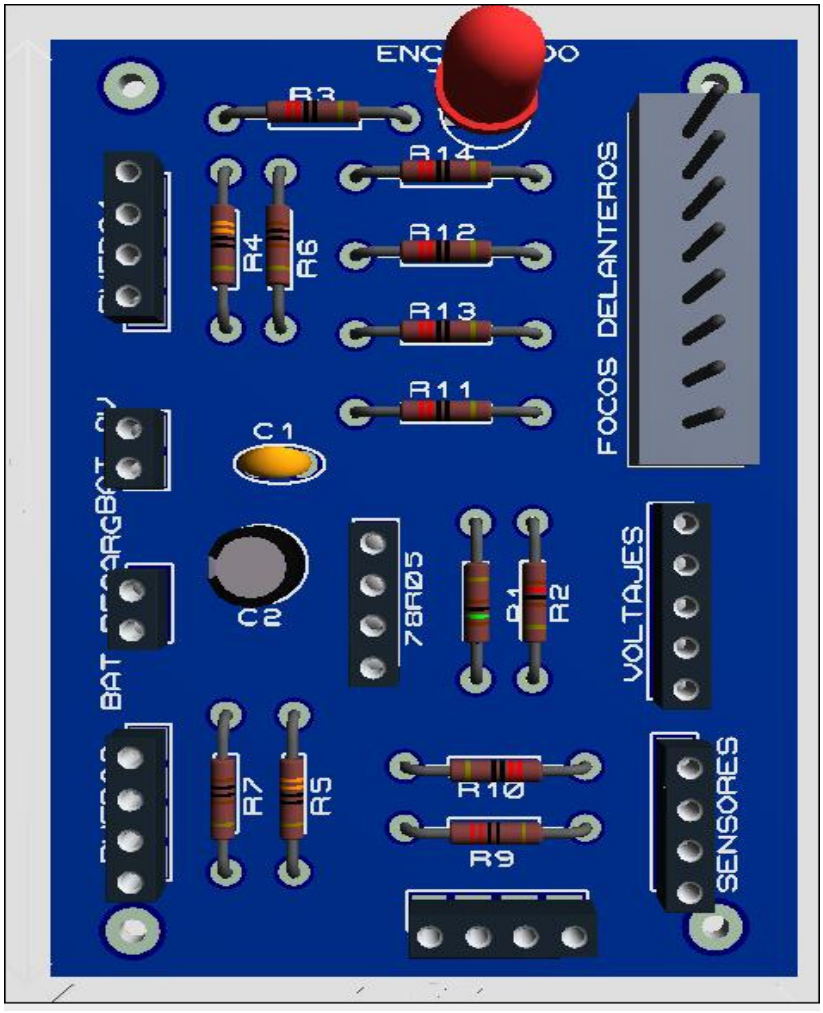


Fig 3-12 Visualización del primer módulo del circuito.

En el segundo módulo se tienen al microcontrolador, al puente h, al amplificador operacional, a los sensores de temperatura, el módulo bluetooth y elementos para su buen funcionamiento. Los conectores que se tienen son para los datos provenientes de la tarjeta del módulo 1 (sensores y voltajes) y para la conexión con los motores de corriente directa salientes del puente h (Fig 3-13 y 3-14). Se hizo de acuerdo al tamaño de la primera para que al conectarlas se tuviera un circuito final de un tamaño reducido y con los elementos bien posicionados.

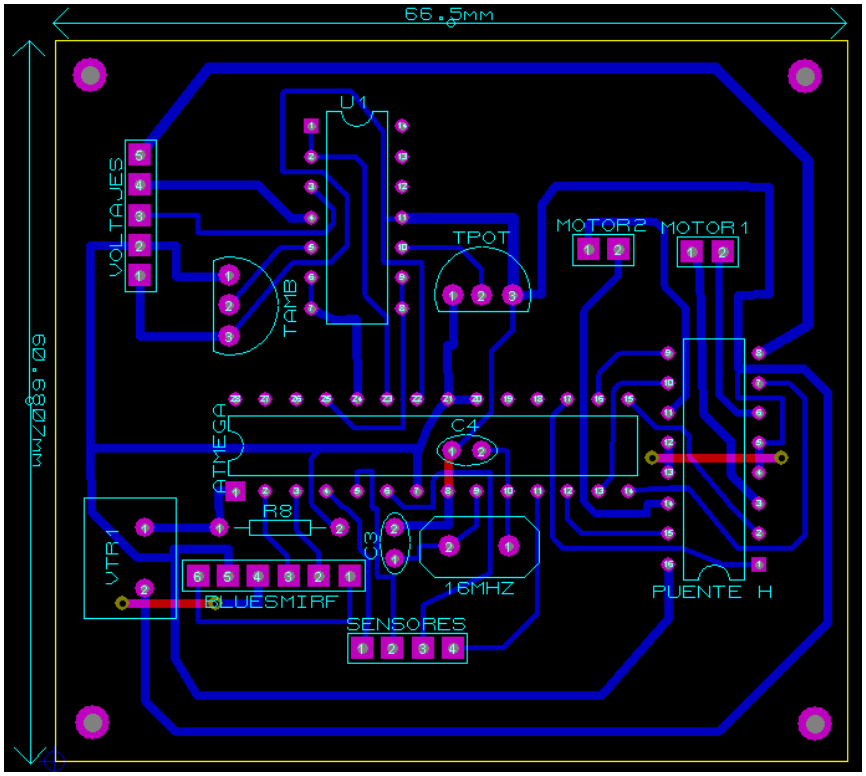


Fig 3-13 Diseño del segundo módulo del circuito, con medidas.

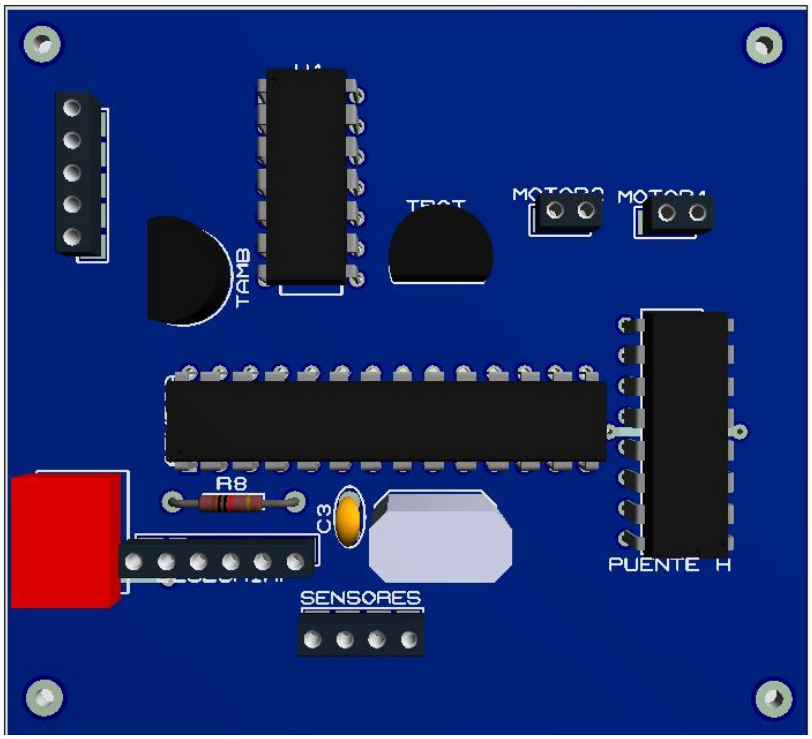


Fig 3-14 Visualización del segundo módulo del circuito.

Para la generación del código de las máquinas de control numérico, se utilizó el programa CopperCAM. Este software te permite visualizar el circuito impreso y a su vez modificar las dimensiones de los barrenos, ancho de las pistas, realizar cortes entre muchas otras funciones. A continuación se muestran algunas vista del circuito electrónico de este proyecto en CopperCAM (Fig 3-15 y 3-16).

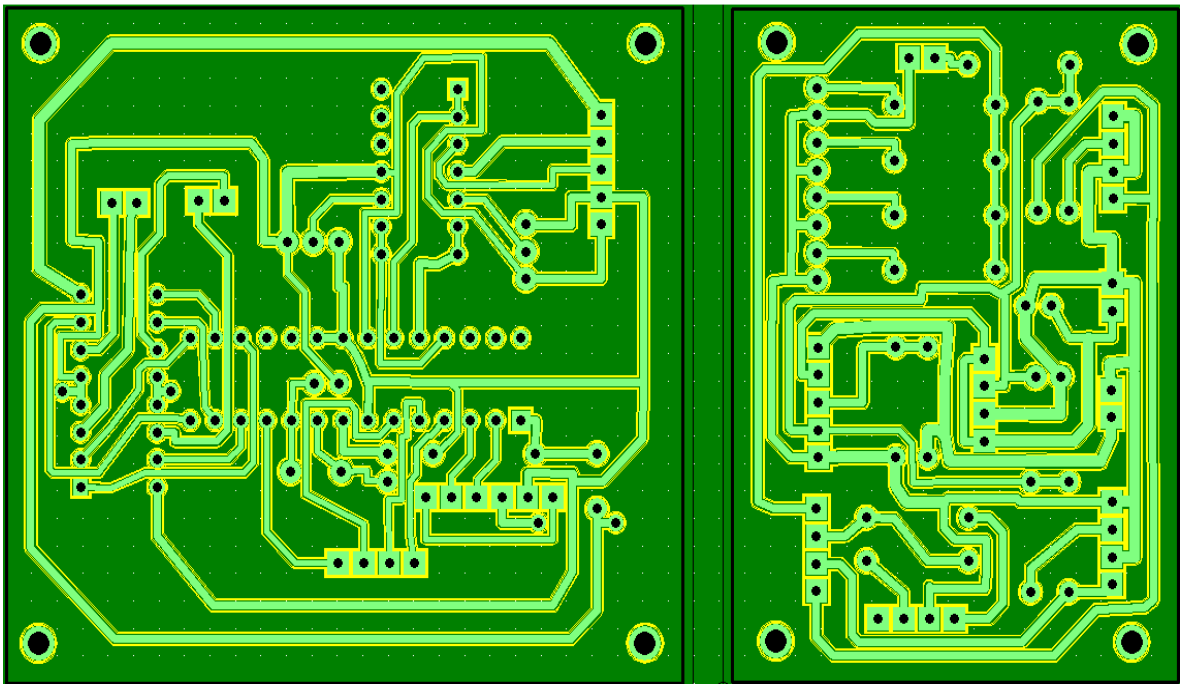


Fig 3-15 Visualización del circuito electrónico usando CopperCAM.

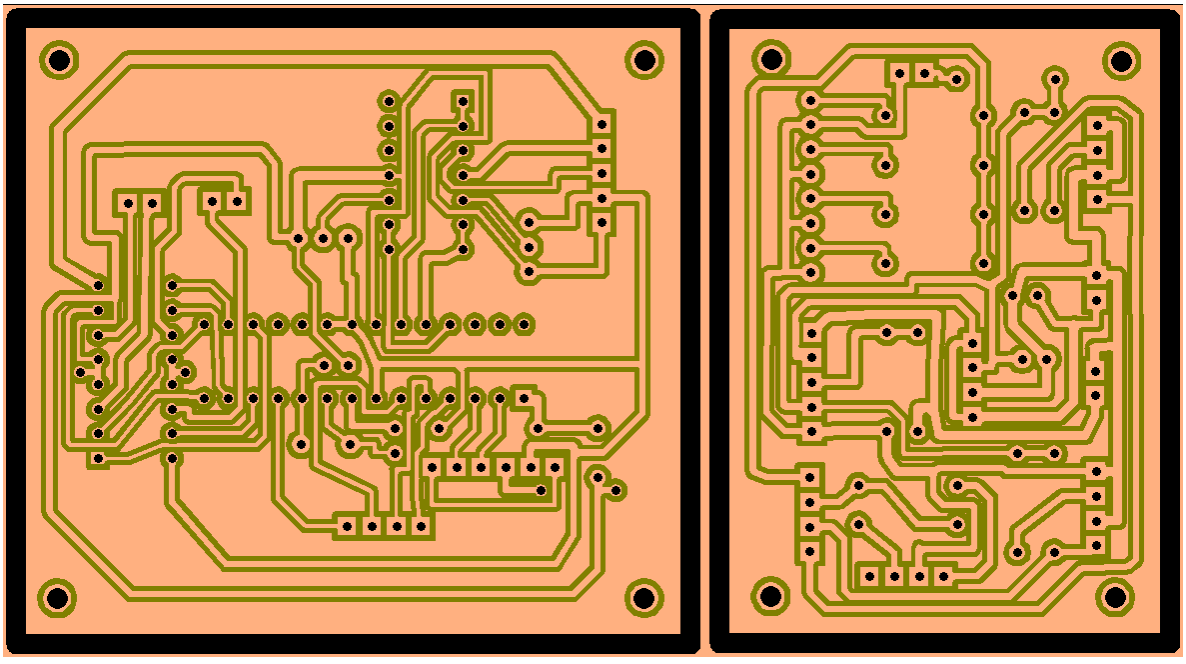


Fig 3-16 Visualización de la placa de cobre usando CopperCAM.

3.8 Pruebas en el circuito electrónico.

Una vez que se maquinó el circuito electrónico y se colocaron todos los elementos necesarios, se verificó que funcionara de acuerdo a lo esperado. Se decidió hacer una tabla con algunas características del circuito electrónico para facilitar su análisis. Se muestran algunas fotografías del circuito electrónico (Fig 3-17 y 3-18).

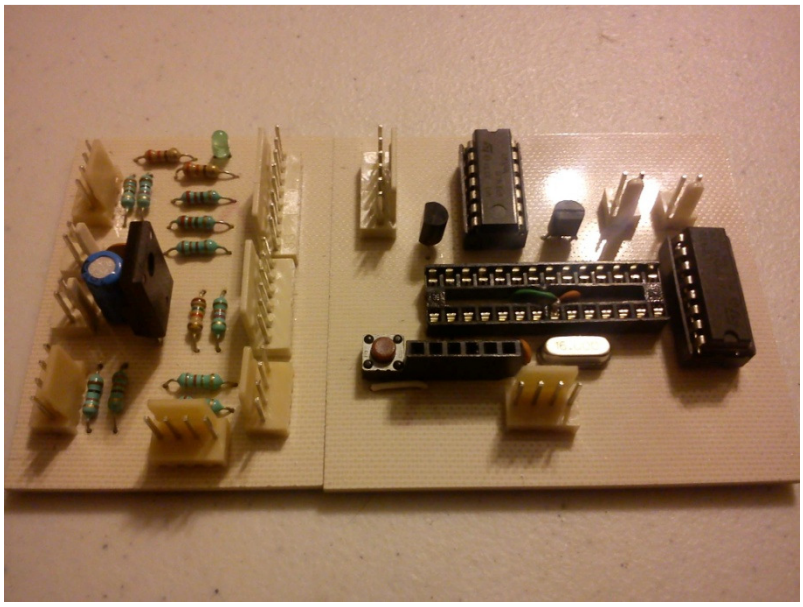


Fig 3-17 Circuito electrónico con los componentes colocados.

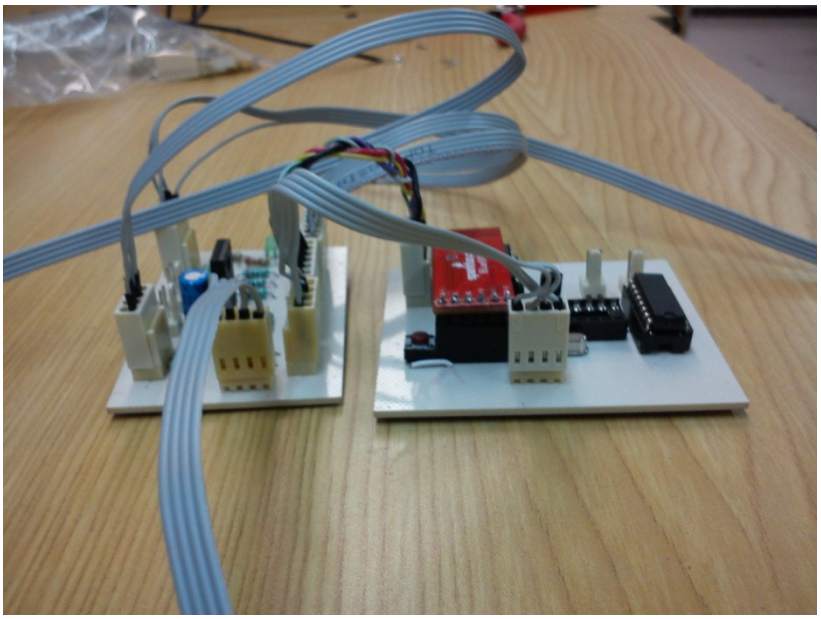


Fig 3-18 Circuito electrónico con los conectores puestos.

Características del circuito electrónico	
Descripción	***Resultado***
Tamaño del circuito	10.5 cm X 6 cm
Corriente de salida para los motores	1.2 A
Tipos de conectores y usos	Molex para asegurar la fijación del circuito. Cabe mencionar que se utilizo cable de diferente calibre dependiendo de la corriente que llevaba.
Voltaje a la salida del regulador de voltaje 78R05	4.99
Voltaje a la salida del divisor de voltaje	4.8 V
Voltaje máximo de las baterías	11.3 V la batería recargable y 9V la pila cuadrada
Voltaje a la salida de los sensores de temperatura	variación de 10 mV por grado centígrado
Voltaje a la salida del seguidor de voltaje	Con un voltaje de saturación positivo de 7.2 V. la carga entra 4.65V y sale 4.65, en la tamb entra .234 y sale .237 y en la tpot entra .238 y sale .241
Duración de las baterías	1.5 horas

Capítulo 4. DESARROLLO DE LA APLICACIÓN PARA ANDROID

4.1 ¿Qué es y cómo funciona el SDK para Android?

El SDK (Software Development Kit) nos proporciona las herramientas y las API (Application Programming Interface) necesarias para poder crear y desarrollar aplicaciones que se puedan ejecutar en dispositivos con la tecnología de Android. Puede usarse en los siguientes sistemas operativos:

- Windows XP (32-bit), Vista (32- o 64-bit), o Windows 7 (32- o 64-bit).
- Mac OS X 10.5.8 o superior (únicamente x86).
- Linux (probado en Ubuntu Linux y Lucid Lynx).

Para poder ejecutar el SDK dentro de cualquier sistema, se requiere del JDK (Java Development Kit). Estos son los componentes que están disponibles desde el sitio oficial de Android y se pueden descargar con la herramienta de Android SDK and AVD Manager como lo muestra la Fig 4-1.

Tipo de componente	**Tamaño aproximado**	**Importancia**
SDK Tools	35 MB	Forzosa
SDK Platform-tools	6 MB	Forzosa
Android platform (para cada versión)	150 MB	Por lo menos se requiere de una
SDK Add-on (para cada versión)	100 MB	Opcional
USB Driver para Windows	10 MB	Opcional (únicamente para usuarios con Windows)
Ejemplos (Para cada versión de Android)	10 MB	Opcional
Documentación fuera de línea.	250 MB	Opcional

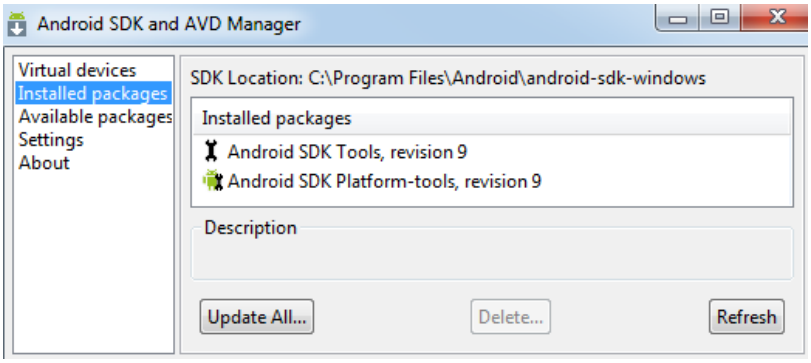


Fig 4-1 Herramienta para descargar los archivos necesarios

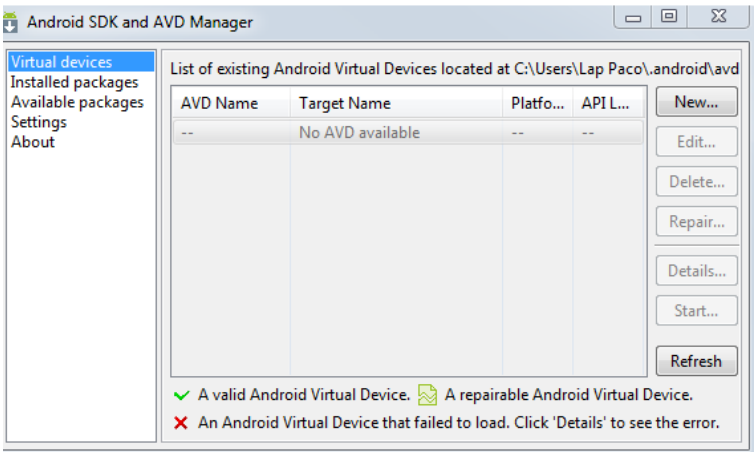


Fig 4-2 Herramienta para crear, editar y eliminar emuladores de Android.

Cuando este paquete se descarga también se incluyen el emulador, el AVD (Android Virtual Device) Manager y el DDMS (Dalvik Debug Monitor Server). El emulador simula un dispositivo móvil con Android que podemos manipular con la computadora. Nos permite probar las aplicaciones que vayamos creando de manera más rápida y sin la necesidad de conectar un dispositivo móvil. La limitante más importante al probar las aplicaciones es que el emulador no puede acceder a sensores como acelerómetros, sensores de iluminación, de temperatura, de proximidad, etc., o a características del hardware como antena Wi-Fi, Bluetooth, GPS, etc. aunque existen plug-ins que se pueden instalar para mandar al emulador valores aleatorios de sensores y ver su comportamiento final.

El AVD Manager (Fig 4-2) nos permite crear distintos tipos de dispositivos Android, modificar la configuración del emulador para poder modelarlo definiendo el hardware y el software que necesitamos y eliminar los perfiles que hayamos creado y ya no sean necesarios.

El DDMS es una herramienta de depuración que brinda distintos servicios como captura de pantalla, información acerca de los Threads (hilos de programación) que se corren simultáneamente, el Logcat, etc. El Logcat es una herramienta muy útil de Android, ya que nos permite ir siguiendo el código que se ejecuta tanto en el dispositivo como en el emulador. Podemos involucrar valores de variables y es fácil de manipular ya que cada línea lleva una etiqueta que se puede filtrar para desplegar solo las etiquetas de ese proceso. También nos despliega información relacionada al Hardware que se esté utilizando y, en caso de existir errores, nos los desplegará.

4.2 Componentes de las aplicaciones en Android

Como ya se había mencionado, las aplicaciones de Android se escriben en el lenguaje de programación Java. Este código se compila junto con otros datos y archivos de recursos en un "paquete Android" que tiene un sufijo .apk. Este archivo se utiliza para poder instalar las aplicaciones en el dispositivo. Las aplicaciones de Android se basan en los archivo de código Java. Existen diferentes maneras en que el sistema puede ejecutar la aplicación y esto depende de cómo se extiendan o empleen las clases del código. Una sola aplicación puede contener varias clases y estas pueden interactuar entre sí o actuar de manera independiente.

Existen cuatro diferentes tipos de clases principales. Cada una tiene un diferente propósito y tienen distintos ciclos de vida para determinar cómo es creada y destruida:

-*Activity*: representa una sola presentación de pantalla con su interfaz de usuario. Se pueden usar varios tipos Activity para una sola acción de una aplicación aunque trabajen independientemente.

-*Service*: es un componente que corre dentro del sistema sin interfaz de usuario por un periodo de tiempo largo.

-*Content Provider*: Nos permite manipular cierta información de aplicaciones (datos de contactos, texto de notas, etc.). Se puede modificar la información solo si se tiene permiso del content provider.

-*Broadcast Receiver*: es un componente que responde a sucesos determinados dentro del sistema.

Las aplicaciones de Android pueden iniciar otras aplicaciones que estén dentro del dispositivo. Durante el desarrollo de aplicaciones es muy importante hacer los ajustes necesarios e incorporar el código necesario para que funcione adecuadamente y para

evitar que deje de funcionar inesperadamente por alguna entrada de datos o por no contar con el hardware necesario para implementar la aplicación.

Algunos de estos aspectos que deben ser indicados son:

- Los componentes que la aplicación usará.
- Permisos especiales del hardware y software para realizar tareas específicas.
- Nivel de la API (Application Programming Interface) mínimo para poder ser ejecutada.

Como se había dicho anteriormente, la aplicación contiene un archivo denominado "Manifest.xml" el cual, al ser externo al código principal, se puede acceder fácil y rápidamente al momento de instalarla en el dispositivo y con ello el usuario sabrá que funciones del equipo empleará la aplicación. También se declaran todos los componentes o archivos que se utilizaran. Con los permisos adecuados se puede utilizar prácticamente cualquier función del hardware del teléfono. Las aplicaciones desarrolladas para Android siguen el siguiente ciclo de vida:

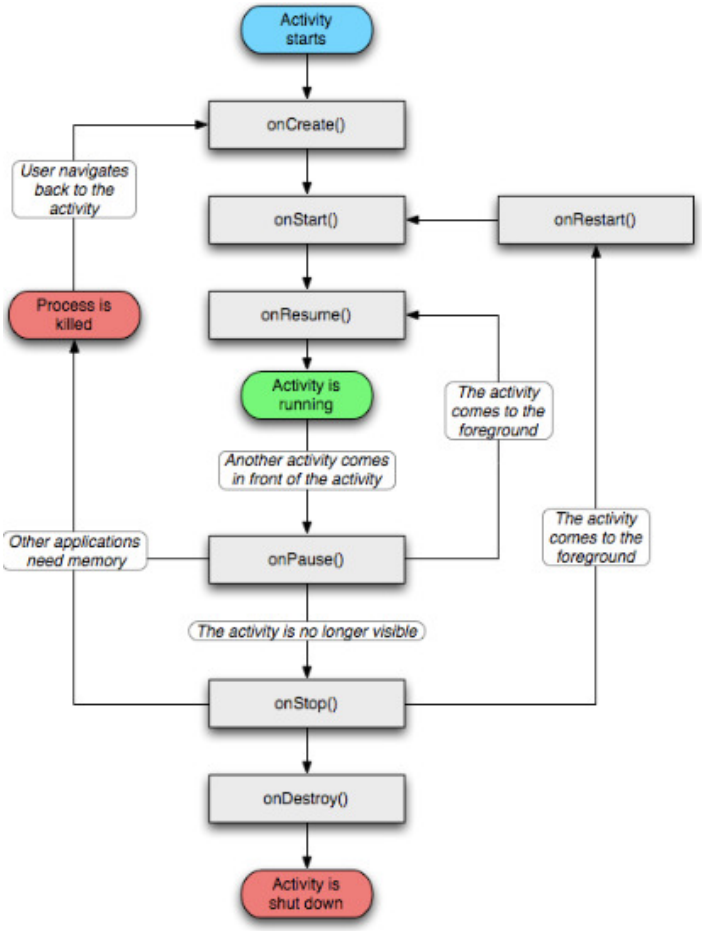


Fig 4-3 Ciclo de vida de las aplicaciones Android

- OnCreate: es invocado cuando la actividad es creada. Normalmente aquí se establece toda la interfaz de usuario y en general inicia la actividad.

- onStart: Se invoca justo antes de que la actividad sea visible en la pantalla. Si la actividad aparece en primer plano en la pantalla se pasa al método onResume, si no al método onStop.

- onResume: En este punto la actividad interactúa con el usuario. Se pueden recibir eventos de entrada y la pantalla mostrará alguna interfaz. onResume también se invoca cuando se regresa a la misma actividad después de haber pasado a otra.

- onPause: se invoca al momento de que se pasará a una nueva actividad. En este punto la actividad dejará de aparecer en la pantalla y es bueno liberar ciertos recursos. No se garantiza que al volver a la actividad está corra de nuevo bajo el método onResume ya que si el dispositivo requiere de la memoria usada por la aplicación, este la finalizará para usar los recursos requeridos.

- onDestroy: El último método invocado justo antes de que el proceso sea destruido. Normalmente se llega a este punto usando el método finish(). Como se mencionó antes, este método puede ser llamado por Android ante la necesidad de recursos. En este lugar no se deben poner las instrucciones para guardar datos ya que puede no ser invocado.

El SDK incluye la herramienta de compresión de archivos AAPT (Android Asset Packaging Tool) la cual toma los archivos de recursos como el main.xml (los componentes de la pantalla) y los compila en un archivo llamado R.java para que pueda ser usado por el código escrito en Java de la aplicación.

La carpeta res es la que debe contener archivos de recursos de Android como:

- Drawables: imágenes como mapas de bits e iconos.
- Layout: elementos de la interfaz de usuario.

El archivo main.xml dentro de la subcarpeta Layout, nos proporciona una vista grafica desde donde se pueden insertar amistosamente elementos que necesitaremos en la interfaz de usuario (con la ayuda del IDE u otro programa como el DroidDraw) y una vista de código desde donde se pueden modificar, quitar o agregar estas herramientas. Este archivo se convertirá en el archivo R.java para poder ser accedido desde el código escrito en Java.

Otra carpeta es la de Values, que contiene la localización de valores de cadenas como el nombre de la aplicación u otras empleadas por esta.

El archivo Manifest representa información de implementación del proyecto. Aquí se establecen las actividades que se usaran, los servicios, permisos de los que se tiene que disponer, y ciertas características de la aplicación como si será debuggable directamente con el dispositivo (se tiene que activar esta opción también desde el dispositivo), la versión de Android para la cual está diseñada y la versión mínima que la puede ejecutar.

El archivo R.java se une con los demás códigos escritos en Java y se crea un solo archivo de tipo class el cual a su vez es convertido en un archivo tipo dex (Dalvik Executable). Este archivo junto con otras librerías, imágenes y otros recursos se empaquetan en un archivo apk (Application Package) con la información y restricciones otorgadas por el archivo Manifest como se muestra a continuación:

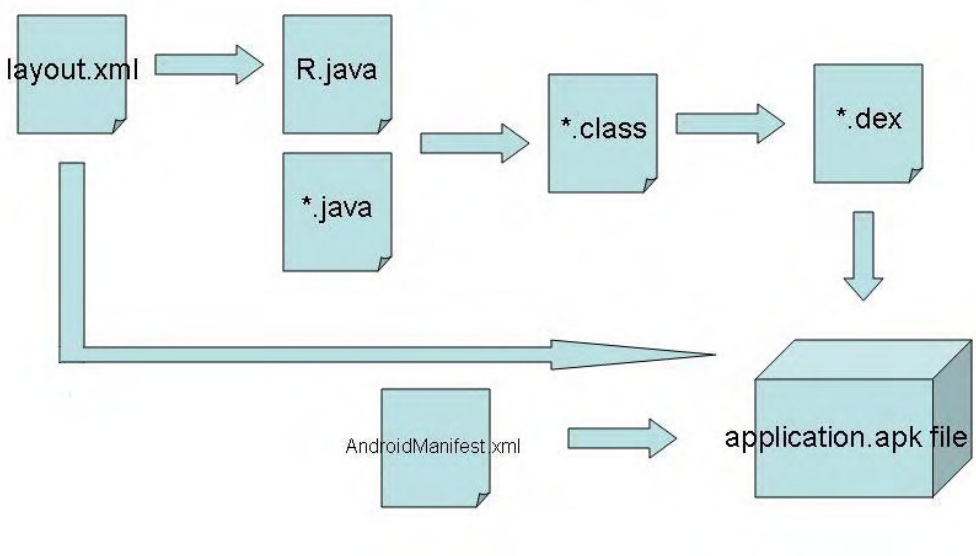


Fig 4-4 Componentes para crear una aplicación de Android

4.3 ¿Qué es y cómo funciona Eclipse?

Para desarrollar aplicaciones se puede o no usar un entorno de desarrollo integrado (IDE por sus siglas en inglés) usando una interfaz de línea de comando (CLI). Es muy complicado, pero no se requiere de la instalación de otros programas que pueden llegar a ser muy pesados.

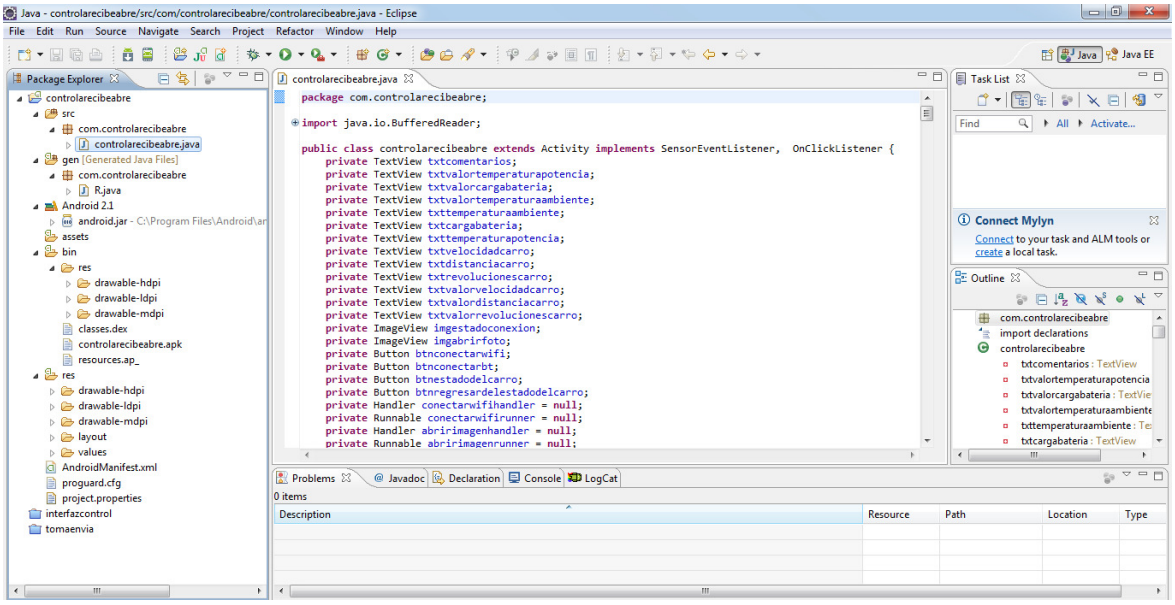
El uso de un IDE en desarrollo de aplicaciones facilita su desarrollo ya que se tiene completo control de lo que se programa, se tienen muchas herramientas que nos ayudan y varias interfaces para que podamos observar varios aspectos que pueden ser muy importantes. Se pueden usar diversos IDE como el JBuilder o el NetBeans, pero la OHA recomienda el Eclipse y actualmente también el IntelliJ IDEA.

Se optó por usar Eclipse ya que es un entorno completo y avanzado de compilación y depuración, fácil de usar, no requiere instalación y es bastante común en el desarrollo de aplicaciones Java y Android (todos los libros de Android consultados lo usan). Al igual que Android, Eclipse también se caracteriza por ser software libre. La OHA lanzó un plug-in especial para Eclipse que permite usar todas las herramientas del SDK como emuladores, simulador y el compilador desde su entorno, lo que hace que también se gane tiempo ya que el código se puede compilar cada vez que se guarda el código fuente o se puede ejecutar el código en el emulador con tan solo oprimir un botón.

Eclipse está disponible para Mac, Linux y Windows. Se requiere del JDK, el SDK para Android, el Eclipse para Java y el plug-in ADT (Android Developer Tools). Dentro de las ventajas que ofrece Eclipse, si tenemos errores o cosas donde el código se pueda detener de forma imprevista, lo marca dentro del código, o bien, tiene una ventana especial para mostrar todos los errores y advertencias.

Otra ventaja es que si los elementos que estamos usando dentro del código incluyen información del tipo Javadoc nos la puede mostrar de manera sencilla, únicamente se tiene que posicionar el cursor del ratón sobre el elemento del cual tengamos duda y nos mostrará automáticamente la información disponible. Esta función se puede activar o desactivar presionando las teclas control + barra espaciadora.

La perspectiva DDMS (Dalvik Debug Monitor Service) proporciona una especie de panel de control de un dispositivo Android en ejecución o emulador. El logcat es una vista dentro de esta perspectiva que nos permite ver el registro del sistema y los distintos mensajes entre el emulador y/o el dispositivo. Crea la posibilidad de escribir mensajes dentro del código fuente para poder verlos dentro del logcat mientras el código está en, así como para obtener valores de ciertas variables que se pueden considerar críticas dentro de la aplicación.



Entorno de Eclipse

4.4 Diseño de la pantalla para el usuario en el teléfono móvil.

Para tener una aplicación más interactiva se dispone de varios diseños de pantalla amigables que irán guiando al usuario para tener una mejor experiencia a la hora de manejar el vehículo como se muestra a continuación.

4.4.1 Identificación de las partes en la pantalla.

Para el dispositivo que controla al vehículo, abre las imágenes y despliega el estado del carro tenemos lo siguiente: Contamos con texto que nos irá guiando para establecer las conexiones necesarias para utilizar la aplicación de manera completa, un ícono que estará demostrando cierta expresión y estará en color rojo siempre que se encuentre desconectada la conexión Bluetooth con el vehículo y botones para poder ir avanzando en los pasos para realizar conexiones exitosas.

En la Fig 4-3 se muestra la pantalla inicial de la aplicación junto con sus respectivos componentes. Al oprimir el botón “conectar con el carro” se establecerá la conexión Bluetooth del vehículo siempre y cuando la antena se encuentre disponible y dentro del rango de operación del dispositivo móvil.

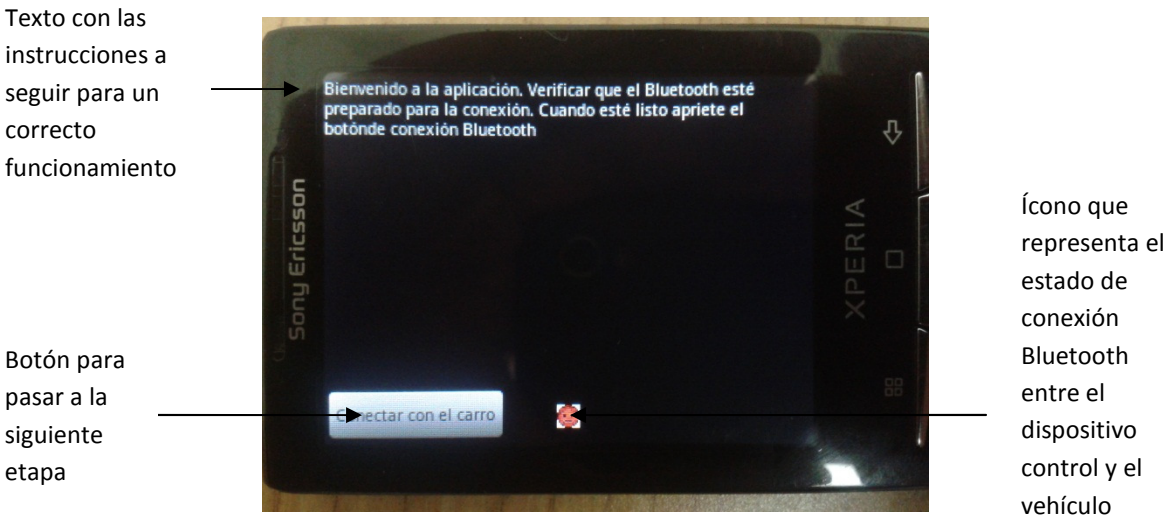


Fig 4-5 Pantalla inicial

Si se establece la conexión Bluetooth de manera exitosa, pasaremos a una pantalla intermedia para poder realizar la conexión entre los dispositivos Android que previamente deben de estar conectados vía Wi-Fi usando el dispositivo que toma imágenes como servidor y el dispositivo de control como cliente usando la opción de anclaje de internet o Tethering. Se abre la aplicación para tomar imágenes que se encuentra en el dispositivo situado en el vehículo y se oprime el botón “conectar y enviar”. Cuando hayamos hecho todo lo anterior, apretamos el botón que aparece en la pantalla del teléfono de control “Conectar con la cámara” como lo muestra la figura 4.4.

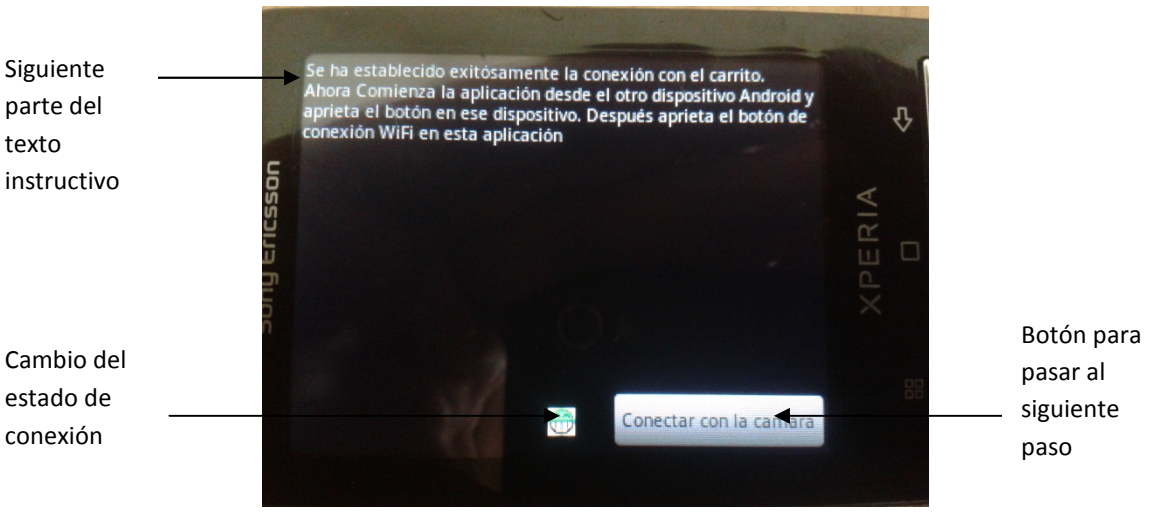


Fig 4-6 Pantalla después de establecer la comunicación Bluetooth con el vehículo

Una vez establecida la comunicación Wi-Fi con el otro dispositivo, automáticamente comienza la secuencia de tomar y enviar fotos por parte del otro dispositivo, mientras que son guardadas y abiertas de igual manera por el dispositivo de control (ver figura 4.5)

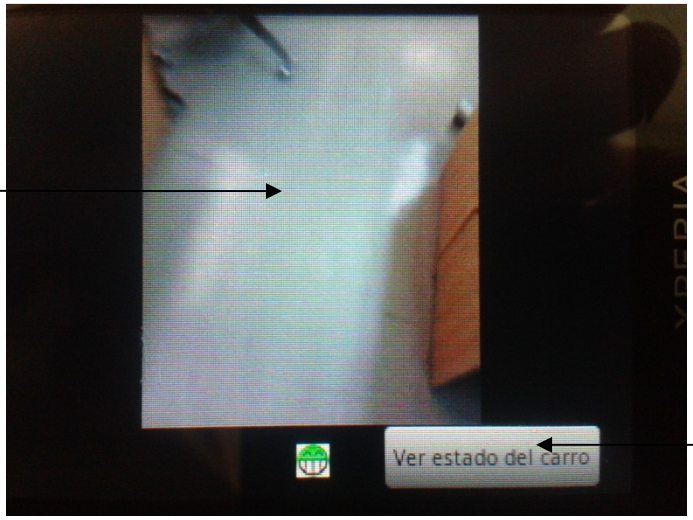


Fig 4-7 Pantalla después de realizar la conexión Wi-Fi con el dispositivo Android

En cualquier momento de esta etapa se puede oprimir el botón “ver estado del carro” que Nos desplegará información acerca de las condiciones en las que se encuentra el vehículo, entre otros aspectos (Fig 4-6). Al momento de presionarlo, el vehículo se detendrá automáticamente para poder visualizar con atención estos datos, o bien, puede servir como un botón de paro del vehículo. Oprimiendo el botón “regresar” podemos volver a la vista anterior donde se abren las imágenes recibidas del otro dispositivo.

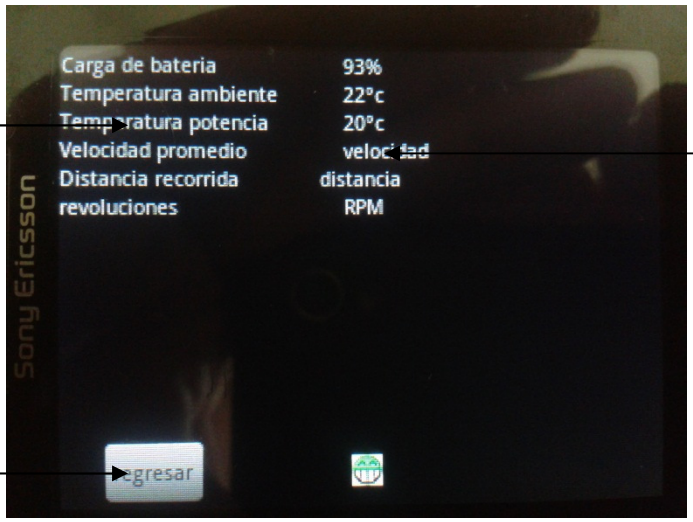


Fig 4-8 Pantalla donde se despliegan los datos del estado del vehículo

Para el dispositivo que estará en el vehículo tomando y enviando las imágenes tenemos lo siguiente:

Al igual que en la aplicación anterior, se tiene una pantalla inicial con instrucciones iniciales que indican que ambos equipos tienen que estar conectados en una red local Wi-Fi (Fig 4-7). De ser así, se oprime el botón “conectar y enviar” para tomar la primera foto, guardarla y prepararla para ser enviada al otro dispositivo.

Fig 4-9 Pantalla inicial de la aplicación

Una vez iniciado el programa, se puede detener la secuencia de fotos en cualquier momento presionando el botón “salir” (fig. 4-8).

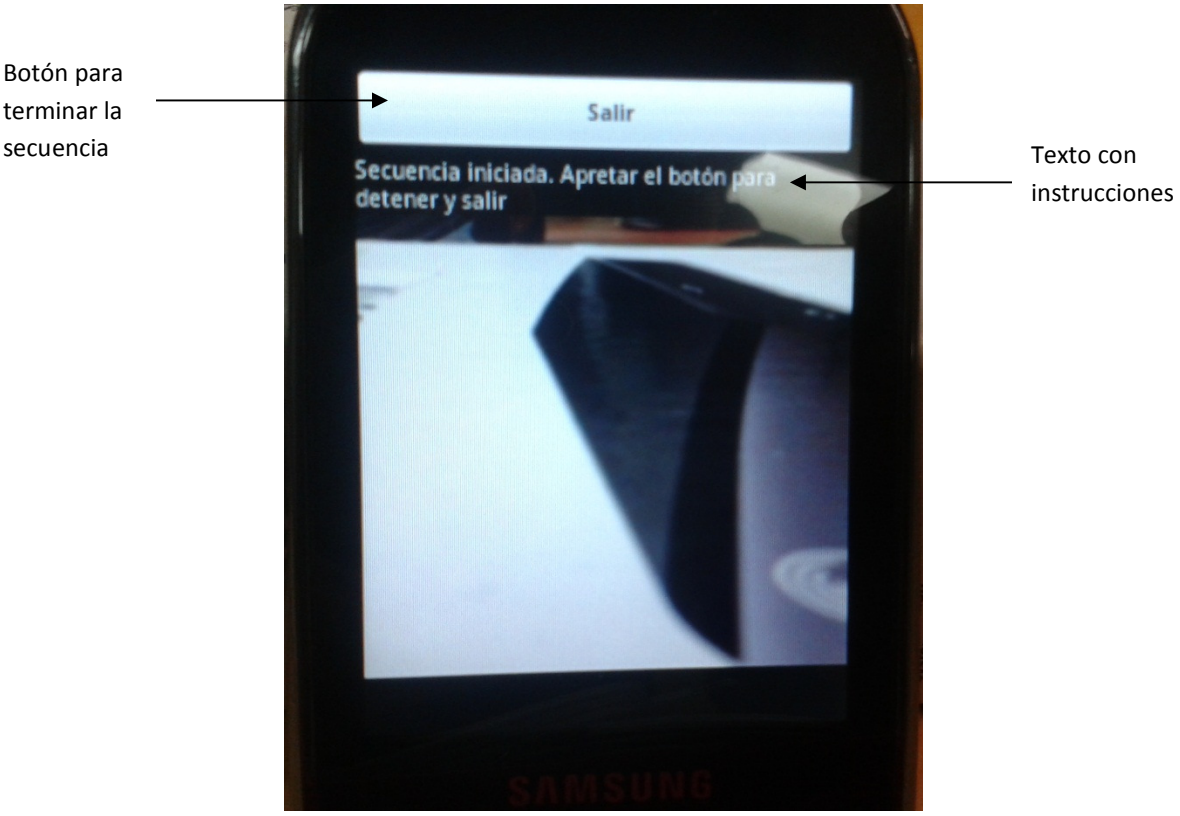


Fig 4-10 Pantalla después de establecer la conexión Wi-Fi con el otro dispositivo

4.5 Aplicaciones desarrolladas durante el proyecto

Los códigos comentados se encuentran en la parte de anexos.

Para el desarrollo del vehículo, se crearon dos aplicaciones principales: una para el teléfono que controla al vehículo y a la vez recibe imágenes, y otra para el dispositivo que está en el carrito y envía las imágenes.

Estas aplicaciones están divididas en sus tres componentes principales para su desarrollo:

- El Layout el cual está integrado por los elementos de la interfaz (main.xml por default).
- El Manifest, que contiene los permisos, propiedades y recursos que utilizará la aplicación (manifest.xml por default).
- La clase principal que controla todos los elementos de la aplicación desarrollada en Java que, para nuestro, caso en ambas se crea extendiendo

la clase Activity para poder emplear recursos de la interfaz (se nombra a partir del identificador de la clase, seguido por un .java).

Tanto el Layout como el Manifest se escriben en XML.

4.5.1 Obtención de datos de los acelerómetros del dispositivo móvil.

Para poder emplear los datos recogidos por el sensor del acelerómetro se tuvo que implementar la librería SensorEventListener que es la que controla todos los sensores del dispositivo, luego se le especifica que el deseado es el del acelerómetro. Se le indica la velocidad entre lecturas. La librería SensorEventListener dispone de dos métodos: onAccuracyChanged (para nuestro propósito no fue empleado) y onSensorChanged que, al percibir alguna variación en la lectura de los acelerómetros, es ejecutado. La precisión con la que cuentan los acelerómetros de los dispositivos actuales hace que prácticamente siempre se esté ejecutando esta parte del código, por lo que es importante determinar de forma adecuada la velocidad entre lecturas que tendrá. Los métodos son del tipo void por lo que no permite regresar valores de variables para utilizarlos en otros métodos por lo que es importante utilizar synchronized(this) para que los valores de variables globales puedan registrar los valores deseados dentro de estos métodos.

Se cuentan con tres acelerómetros dentro del dispositivo (Fig 4-9) pero solo se utilizaron dos (uno para saber el giro que debe hacer el carro: “Y” y otro para la velocidad que tendrá el carro: “Z”, ya sea para adelante o para atrás). Estos datos se truncan, arreglan y se convierten a valores en bytes. En base a los valores negativos o positivos de los acelerómetros, también se especifica si se posiciona el celular a la izquierda, derecha, adelante o atrás. Se crea un arreglo del tipo byte de cinco valores para poder mandar la información de los acelerómetros que contiene:

- La dirección del acelerómetro en “Y” (Si la disposición del teléfono es hacia la izquierda o hacia la derecha).
- La intensidad del giro del acelerómetro en “Y”.
- La dirección del acelerómetro en “Z” (Si la disposición del teléfono es hacia la adelante o hacia la atrás).
- La intensidad del giro del acelerómetro en “Z”.
- Un punto y coma (“;”) para poder dividir estos datos en cadenas que serán enviadas al microcontrolador.

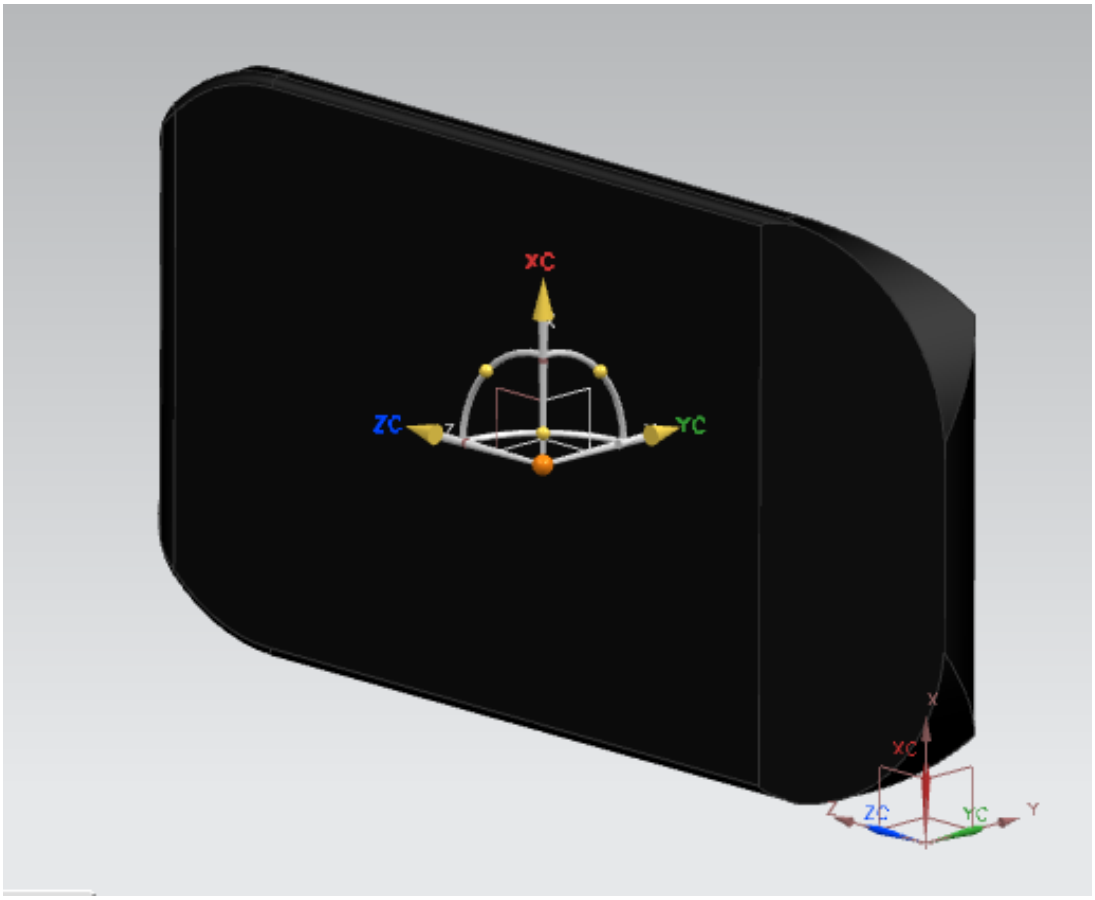


Fig 4-11 Con la pantalla de frente, este es el sistema cordenado de los acelerómetros de nuestro dispositivo Android.

4.5.2 Transferencia de datos vía Bluetooth.

Para establecer la conexión Bluetooth entre el Bluesmirff y el dispositivo Android, se estableció que el Bluesmirff funcione como esclavo (la conexión se realizaría por medio de otro dispositivo), se modificó la velocidad de transferencia de datos ya que con otros valores no se trasmitían de manera adecuada y se dejaron la clave MAC y contraseña de conexión como las establecidas de fábrica.

Al realizar por primera vez la conexión, se tienen que vincular el Bluesmirff con el celular Android para poder ingresar la contraseña de acceso del Bluesmirff. Esto se hace a través del menú de configuración, conexiones inalámbricas, configuración de Bluetooth y se selecciona el Bluesmirff (es el mismo procedimiento que cuando se vinculan dos celulares o con una computadora).

En el código de Android, se establece que la antena Bluetooth que se utilizará es la que viene establecida en el equipo (si es que se cuenta con la capacidad para realizar una conexión Bluetooth, si no tiene la antena, la aplicación se sale a la pantalla principal mostrando que no se cuenta con ella). Si la antena de Bluetooth se encuentra apagada, la aplicación automáticamente la prenderá y esperará cinco segundos para garantizar que la antena se prenda correctamente y que la aplicación no falle debido a que intentará realizar la conexión con la antena apagada.

Se establece en el hilo de este programa dedicado a la transmisión de datos por Bluetooth, la velocidad a la que se enviará la información (un dato en determinada cantidad de microsegundos). Se rastrea la dirección MAC del Bluesmirff para conectarlo, se crea el socket de conexión, se establece la conexión y finalmente se mandan los datos creados en el apartado de los acelerómetros.

4.5.3 Recepción de datos procedentes del ARDUINO.

Para la recepción de información procedente del Atmega, la conexión ya quedó establecida antes de mandar los datos del acelerómetro por lo que ya no se explicará ese punto. De igual manera se establece una velocidad a la cual se leerá la entrada de datos por la antena Bluetooth, se lee la información en la cadena de datos que llega hasta encontrar "/" que es el indicador de que ha finalizado la línea para proceder a una nueva lectura, esta línea es dividida y, dependiendo de la posición de la información, se mandarán los datos a sus respectivas posiciones:

- El primer valor es el del porcentaje de carga de la batería para las llantas de tracción.
- El segundo, para conocer la medida del sensor de temperatura ambiental (°c).
- El tercero, para determinar el valor del sensor de temperatura de la etapa de potencia (°c).
- El cuarto nos proporciona la velocidad promedio que ha registrado el vehículo (cms/s).
- El quinto nos da la distancia recorrida del vehículo (cms).
- El sexto valor da las revoluciones por minuto de las llantas.

Capítulo 5. IMPLEMENTACIÓN DE UNA CÁMARA EN EL VEHÍCULO PARA OBSERVAR LA IMAGEN EN EL DISPOSITIVO MÓVIL.

5.1 Selección de cámara.

Originalmente se pensó utilizar una cámara de video de las utilizadas como medida de seguridad para vigilar algún lugar, que tuviera configuración IP (Internet Protocol) para no tener que intervenir el hardware del equipo, que fuera cómodo el manejo del vehículo (sin un cable que limite la movilidad). Estuvimos buscando la más adecuada, que a su vez fuera económica y se encontraron de distintos tipos:

Fig 5-1 Distintos tipos de camaras IP que encontramos

Las desventajas contra las cuales nos hubiéramos enfrentado seleccionando este tipo de cámara eran:

- Aunque los costos eran diversos, todas exedian el precio que teniamos contemplado.
- Se tendría que utilizar una aplicación que se descargaría desde el Market u otro lugar para poder moniterear el equipo y ,al no ser nuestra aplicación, tendríamos un acceso sumamente restringido al momento de querer manipular las acciones del programa o bien, se hubiera tenido que crear un driver especial para la camara.
- Por el tamaño de la cámara, aún la más pequeña, incrementaría las dimensiones del vehículo de manera considerable.
- La cámara debía de estar conectada a internet mediante un router y el celular (Fig 5-1) a una red Wi-Fi de acceso a internet, o bien, conectar el celular mediante la red de datos de la compañía telefónica, lo que implicaría un gasto cada vez que se usara y, si no se disponía de cobertura de red, sería imposible establecer la conexión.

Fig 5-2 Modelo empleado como celular de control (Sony Ericsson Mini- Xperia con Android v2.1)

Se optó por utilizar la cámara de otro dispositivo Android (Fig 5-2) para contrarestar las desventajas de la camara IP:

- Prácticamente tienen el mismo costo un teléfono tipo smartphone usado que el de la cámara IP.
- Con los conocimientos adquiridos acerca de la programación de Android, ahora podíamos programar una aplicación que nos dejara utilizar las librerías de los drivers de la cámara del teléfono, lo cúal es mucho más sencillo que crear un nuevo driver. También disponemos del completo control de la aplicación.
- El tamaño del teléfono utilizado es mucho menor al de cualquier camara IP.
- Si se dispone con un teléfono con versión de Android 2.2 o superior, se puede emplear la opción de tethering o compartir la red 3G de un dispositivo a otro vía Wi- Fi y con ello logramos que los dos teléfonos queden conectados en una red privada evitando que las compañias telefónicas nos cobren por la transferencia de datos y conectandonos en cualquier lugar aunque no haya otras redes de comunicación del tipo Wi- Fi o 3G disponibles.

Fig 5-3 Teléfono Android seleccionado para tomar y enviar las imágenes (Samsung Galaxy GT-I5500L con Android v2.2)

5.2 Transferencia de imagen al dispositivo móvil.

Antes que nada, el celular que enviará la imagen y que estará dentro del vehículo debe de tener una versión de Android de 2.2 o superior; también se pudiera usar uno con versión 2.1, pero debe de tener acceso al Root del sistema (superpermisos de usuario) y tener instalada alguna aplicación para establecer la conexión vía Tethering (anclaje de internet). Se tiene que conectar el dispositivo de control al celular que tomará las imágenes. El resto de la conexión la establece la aplicación.

En el dispositivo que envía la imagen, se creó un display de previsualización junto con un botón en la pantalla del celular que, al momento de ser apretado, activa una bandera que le indica al programa que inicie la secuencia de tomar fotos la cual, captura la foto y la guarda en una dirección de memoria (test1.jpeg) y se dispone a mandarla vía Wi- Fi al otro dispositivo.

Mientras se envía la primer foto, se toma la segunda foto y la va guardando para que, al momento en que se haya enviado la primer foto, se envíe después la segunda foto (test2.jpeg) y no tener demoras esperando a que el dispositivo guarde la foto. Una vez que se toma y guarda la segunda foto, se retorna a la dirección de test1.jpeg para volver a iniciar la secuencia hasta que el usuario la detiene.

Para la conexión Wi- Fi con el otro dispositivo, este teléfono utiliza un número de puerto para poder enviar la imagen. Cuando el otro dispositivo solicita la conexión y es aceptada, inmediatamente comienza a mandar la primera foto tomada y hasta que se termina de enviar se manda la segunda foto.

Por parte del teléfono que controla al vehículo, el dispositivo solicita la conexión al teléfono que envía la imagen y una vez que se acepta, se espera al archivo que se recibirá, y cuando llega lo almacena en la ruta “recibido1.jpeg” inmediatamente se cambia la dirección de almacenamiento a “recibido2.jpeg” para poder ir guardando la segunda imagen en lo que se despliega la primer imagen en la pantalla.

CAPÍTULO 6. TRABAJO A FUTURO

En este capítulo se mencionan algunas cosas del proyecto que se dejan abiertas para trabajar posteriormente. Las mejoras a futuro que se observaron pueden ser importantes; se dan en la parte electrónica, la construcción del vehículo de acuerdo al diseño CAD y en la transferencia de video en lugar de imágenes al dispositivo controlador. El trabajo a futuro es importante porque resalta la importancia del proyecto y su adaptabilidad a nuevos entornos así como nuevas tareas.

6.1 Circuito electrónico reducido

Dentro de la realización de este proyecto de tesis, se mencionó en el capítulo 3 que la mejor opción de realizar el circuito electrónico era hacerlo a doble cara y con elementos de montaje superficial. Esto se propone como trabajo a futuro para optimizar el espacio y de esta manera que el vehículo sea de un tamaño menor para otro tipo de tareas que así lo requieran.

En los puntos posteriores se profundiza un poco en esta mejora que se propone, comenzando con la selección de nuevos componentes hasta llegar a una visualización de cómo se espera que quede el circuito. Cabe mencionar que la funcionalidad no se ve afectada ya que los componentes realizan las mismas tareas que el circuito que se utilizó para el prototipo, pero se notará una mejora en cuanto al tamaño y estética.

6.1.1 Selección de nuevos componentes.

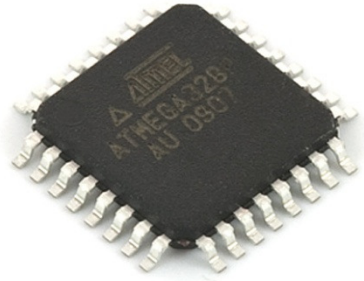
Los elementos que se proponen son los mismos que se tienen en el circuito utilizado, pero esta vez con montaje superficial. No todos los elementos se pueden cambiar, para este punto del trabajo nos basamos en los elementos que se pueden representar según el software Proteus. Si se busca en los catálogos probablemente se encuentren más elementos superficiales pero la idea es la de mostrar una visualización del circuito final por lo que nos limitamos a colocar componentes existentes en dicho software.

Los elementos que se reemplazaron por su similar de montaje superficial (Fig 6-1) fueron los siguientes:

- El puente H L293D.
- El ATMEGA 328P-PU.

- Todos los resistores.
- El condensador número dos del esquema.

a)



b)

c)

d)

Fig 6-1 a) L293DD b) ATMEGA 328 c) Resistores de montaje superficial d) condensadores de montaje superficial.

Algunos de los elementos que no se encontraron para la visualización pero existen en el mercado se muestran en la Fig 6-2.

a)

b)

c)

Fig 6-2 a) Cristal superficial b) LM324 (amplificador operacional) c) Sensor de temperatura LM35

6.1.2 Diseño de la placa del circuito impreso a doble cara y visualización con los componentes puestos.

Para el diseño de la placa (PCB) se utilizó el mismo software para circuitos impresos, ARES. Esta vez se indicó que el trazo de las pistas fuera por ambos lados; respetando el grueso de las pistas para voltaje y tierra. De igual forma se colocaron los elementos de montaje superficial.

A continuación se muestra un ejemplo de cómo sería el PCB con elementos de montaje superficial (Fig 6-3).

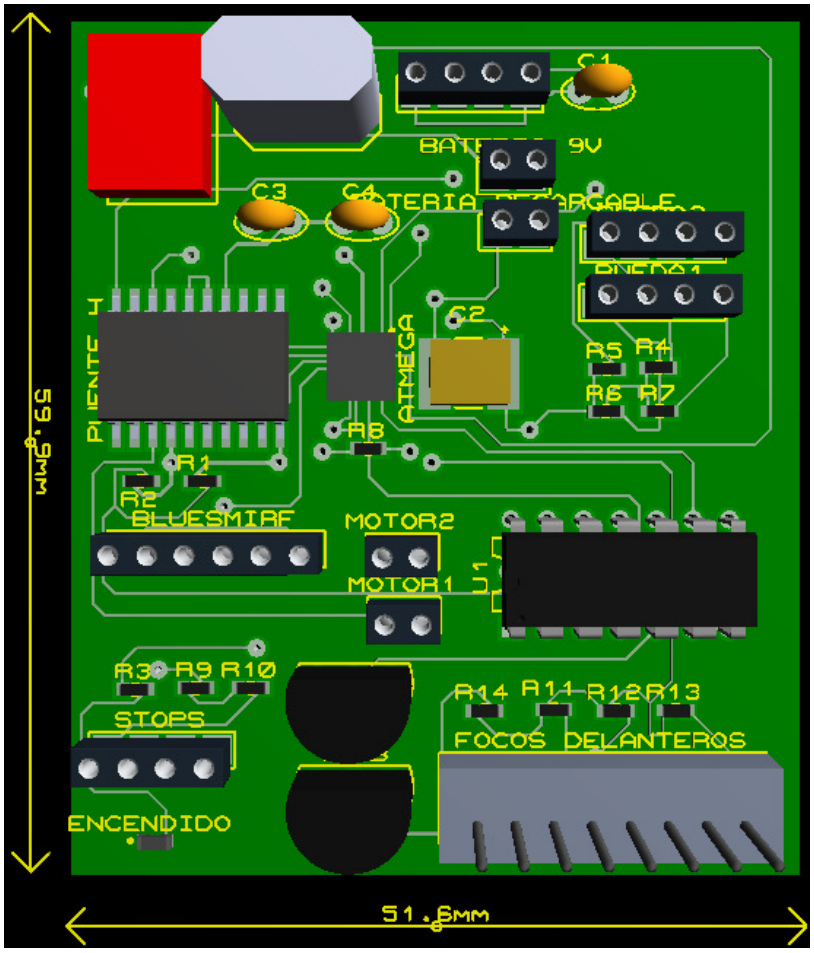


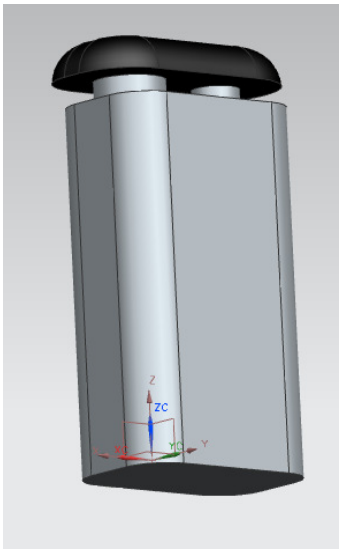
Fig 6-3 Diseño ideal de circuito con elementos de montaje superficial.

Como se observa en la imagen anterior las dimensiones son menores que las que se mostraron en el capítulo 3 de este trabajo. Con un circuito de este tamaño se hace un sistema más robusto y las dimensiones totales del vehículo pueden reducirse. Así mismo se puede proteger de mejor manera al estar en una sola pieza y no en dos módulos.

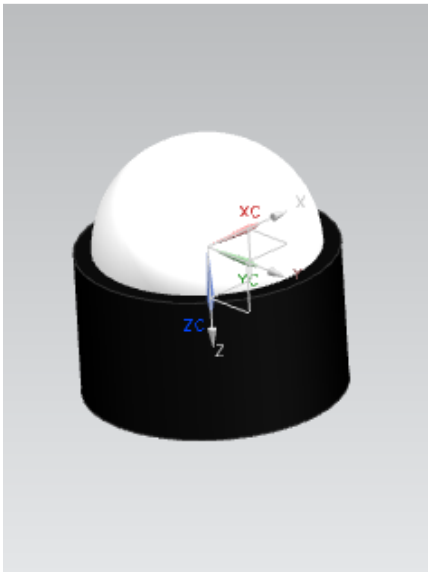
6.2 Construcción del carro

6.2.1 Diseño asistido por computadora del chasis

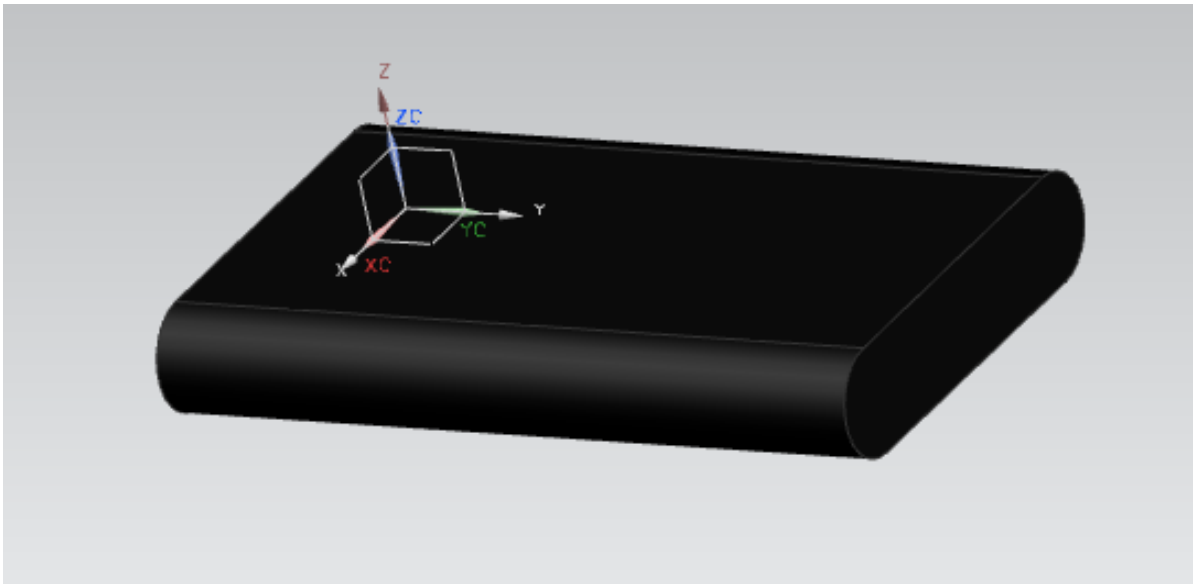
Para poder hacer un modelo por computadora de los prototipos a realizar, primeramente se modelaron los componentes que se utilizaran. A continuación se muestran los componentes para el primer diseño:



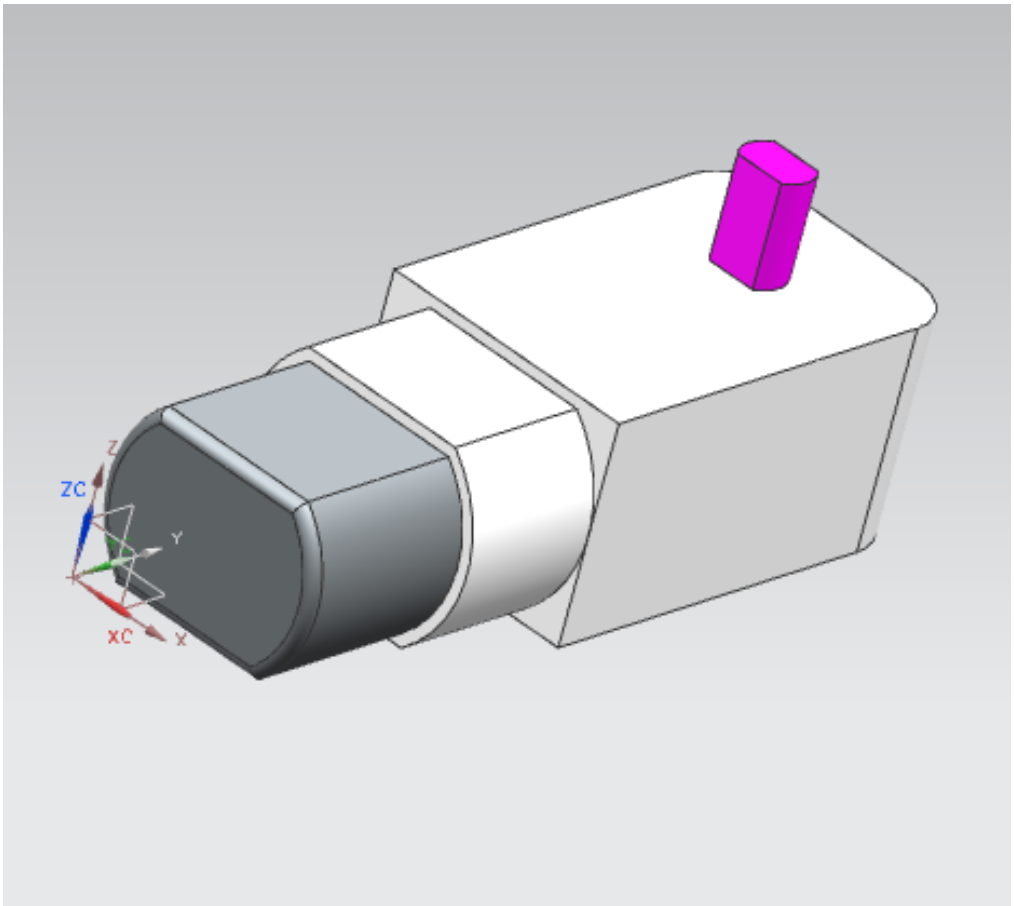
Batería común de 9v con conector de plástico para la alimentación del circuito en general.



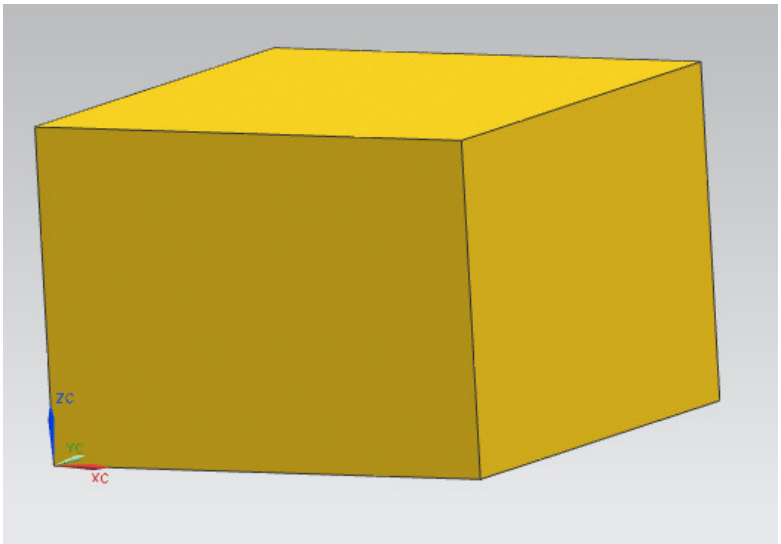
Rueda loca de plástico que nos brinda soporte y facilidad de movimiento al momento de maniobrar el carro



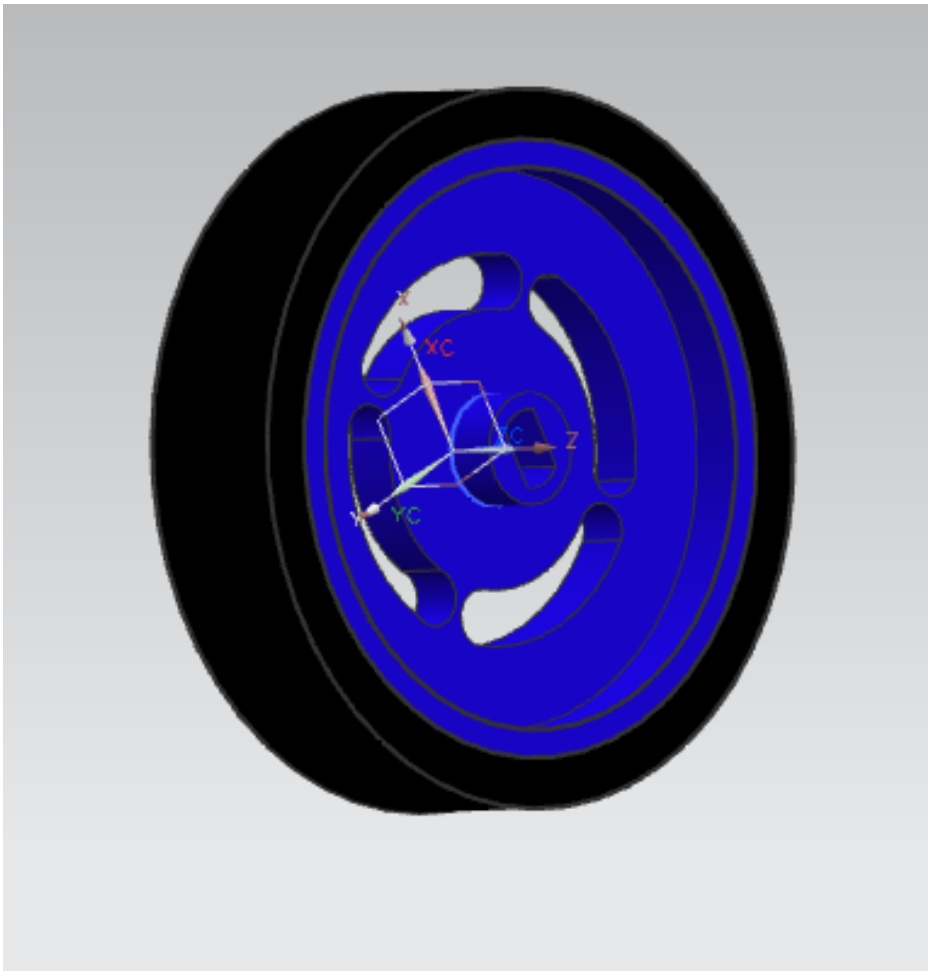
Batería recargable de vehículo radio-control utilizada para alimentar los motores.



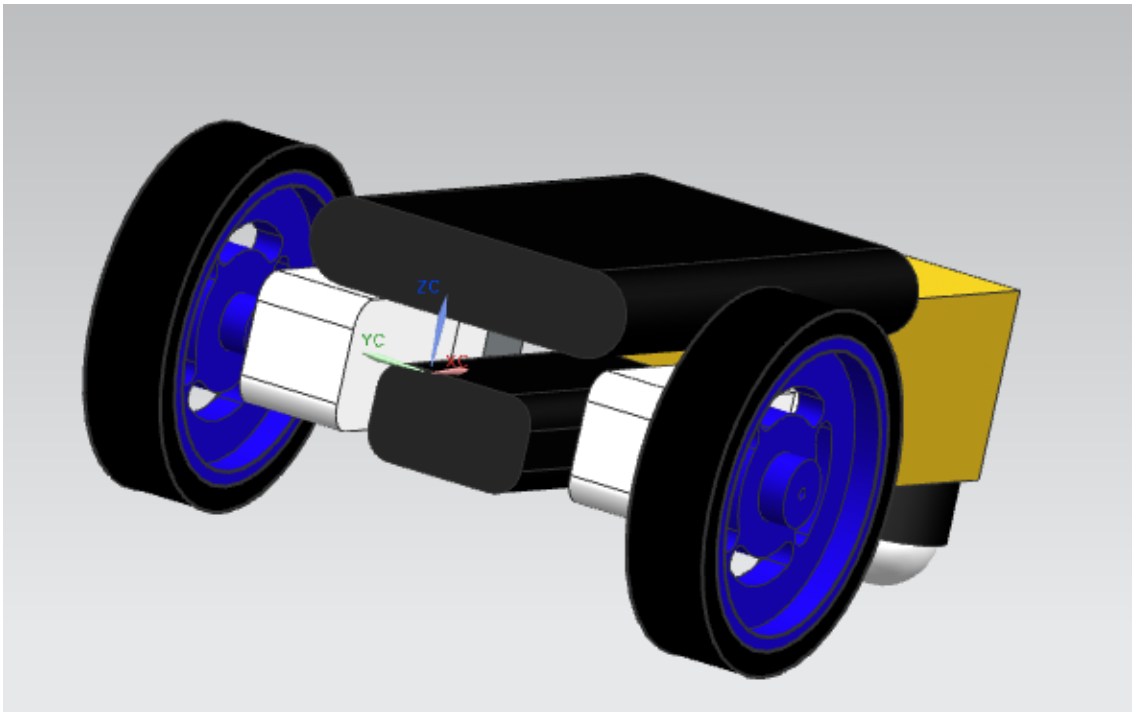
Motorreductor eléctrico de plástico con reducción 1:48.



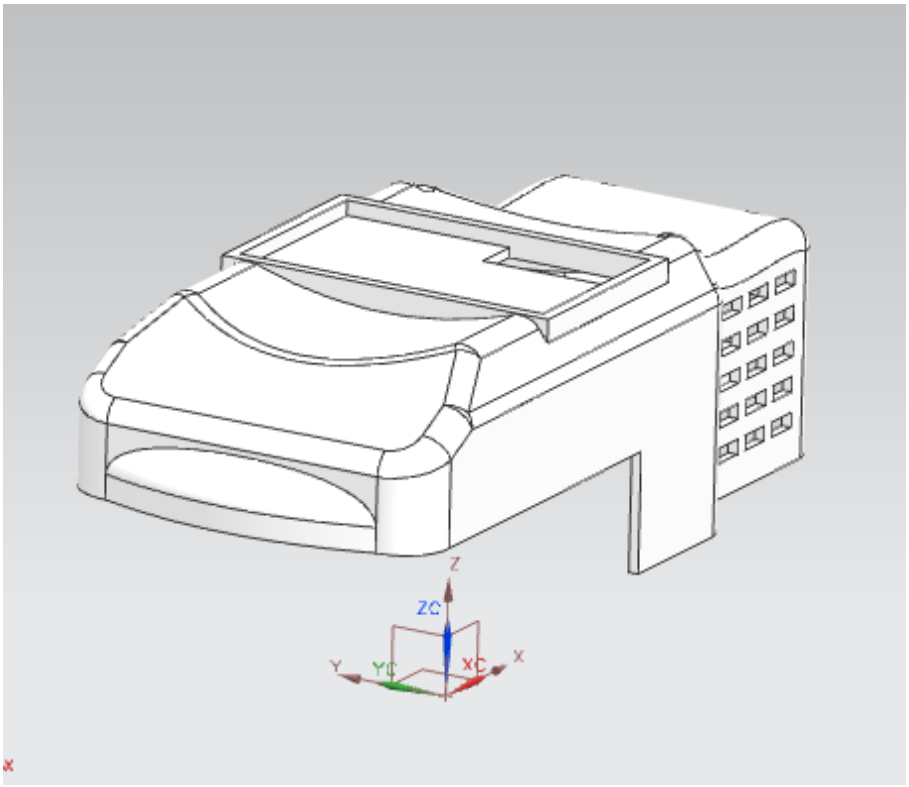
Primer modelo del circuito electrónico (solo para establecer dimensiones máximas de componentes).



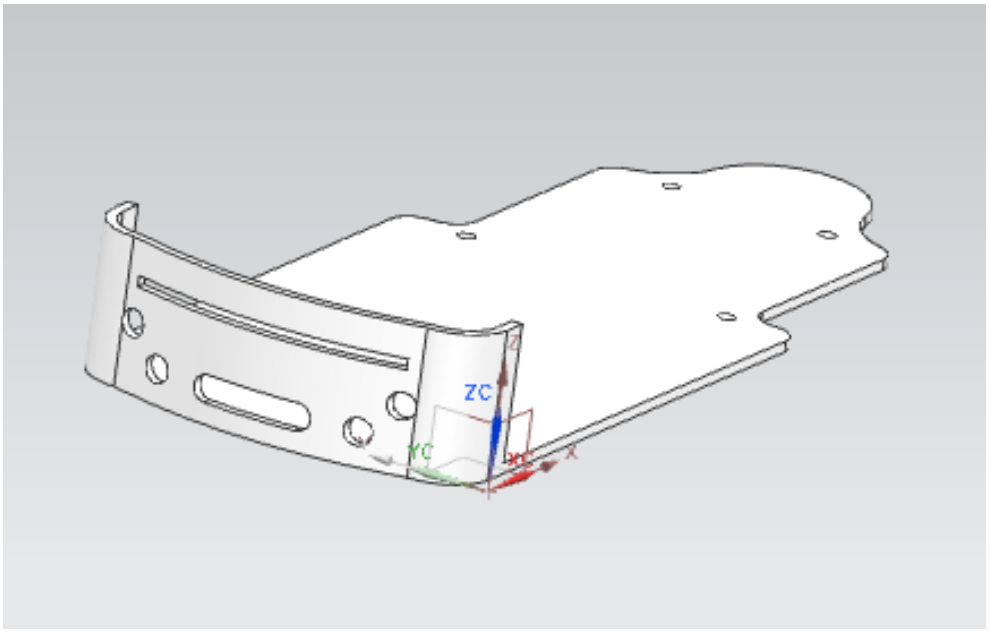
Rueda de plástico con aro de goma



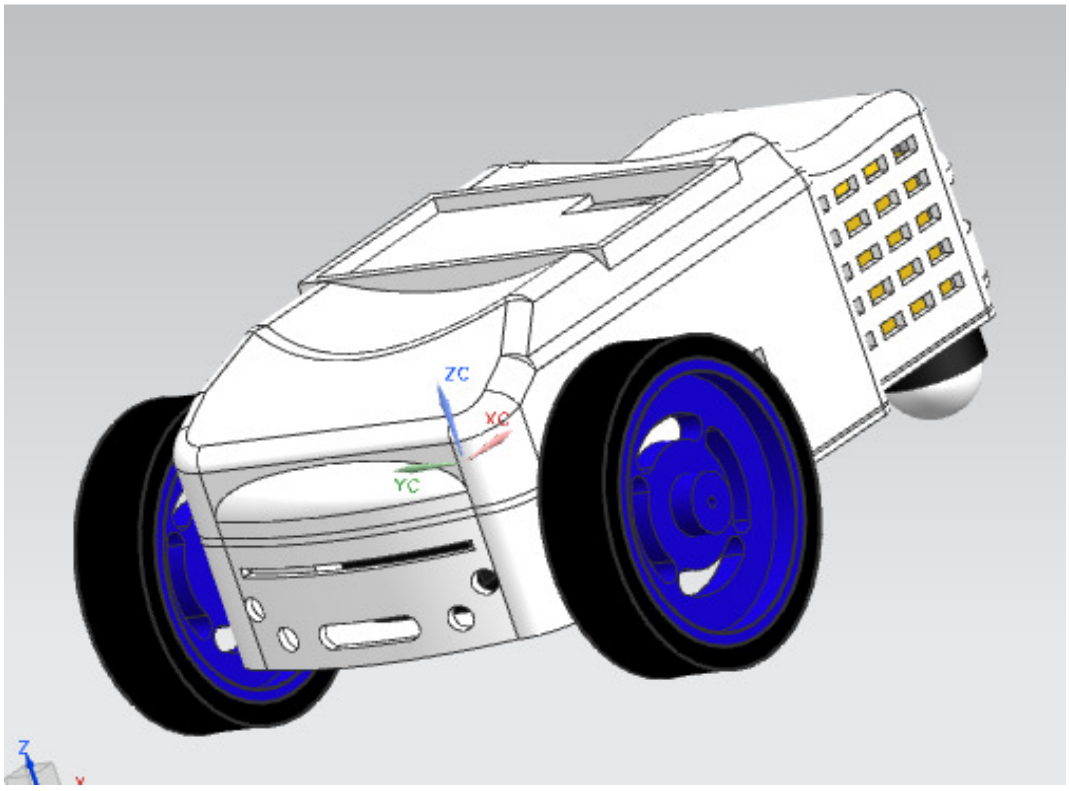
Primer ensamble de componentes para deducir las dimensiones óptimas del chasis



Primer propuesta de la parte superior del chasis

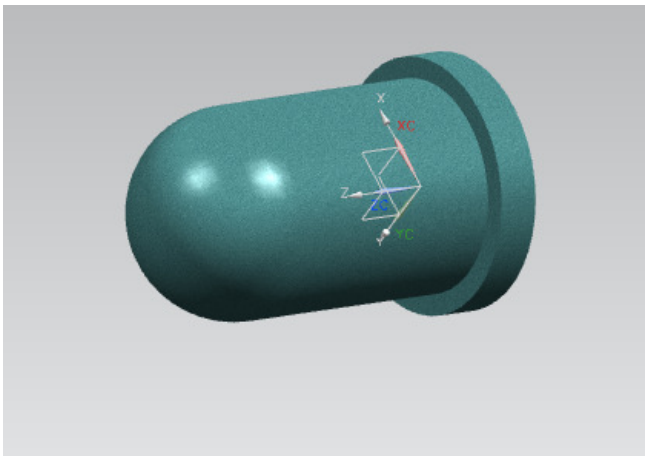


Primer propuesta de la parte inferior del chasis.

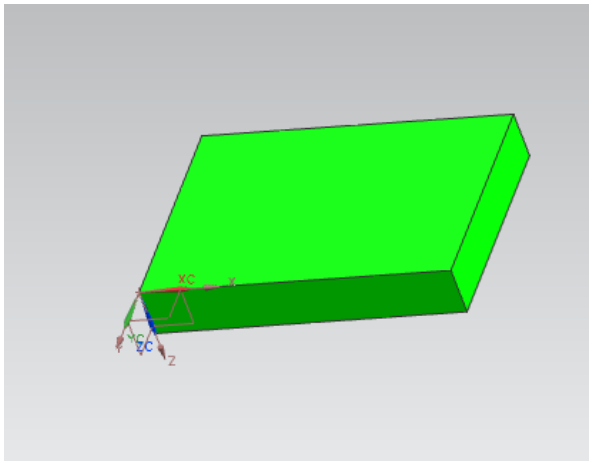


Ensamble completo de la primera propuesta de prototipo

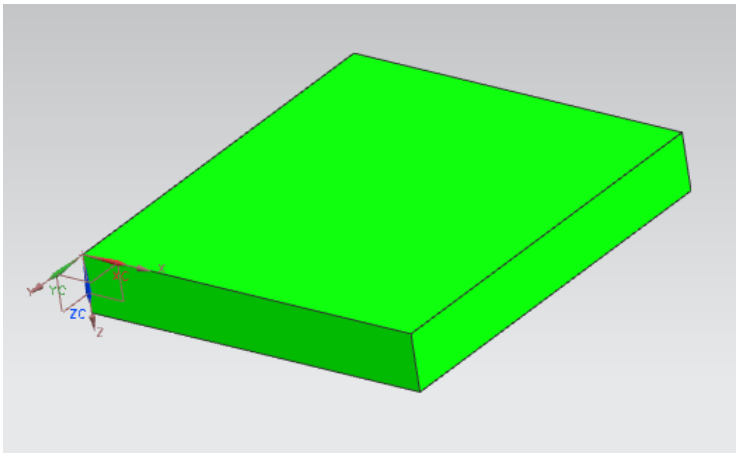
Se decidió agregarle nuevos componentes como el celular para poder usar su cámara, agregarle LEDs en la parte frontal y en la parte trasera para tener visión por parte de la cámara aún en lugares oscuros y por seguridad del vehículo. Esta modificación hizo que el diseño de la placa del circuito eléctrico también cambiara para adecuar el uso de los LEDs (se emplean 2 placas para simplificar el diseño de las pistas) al igual que el diseño del chasis.



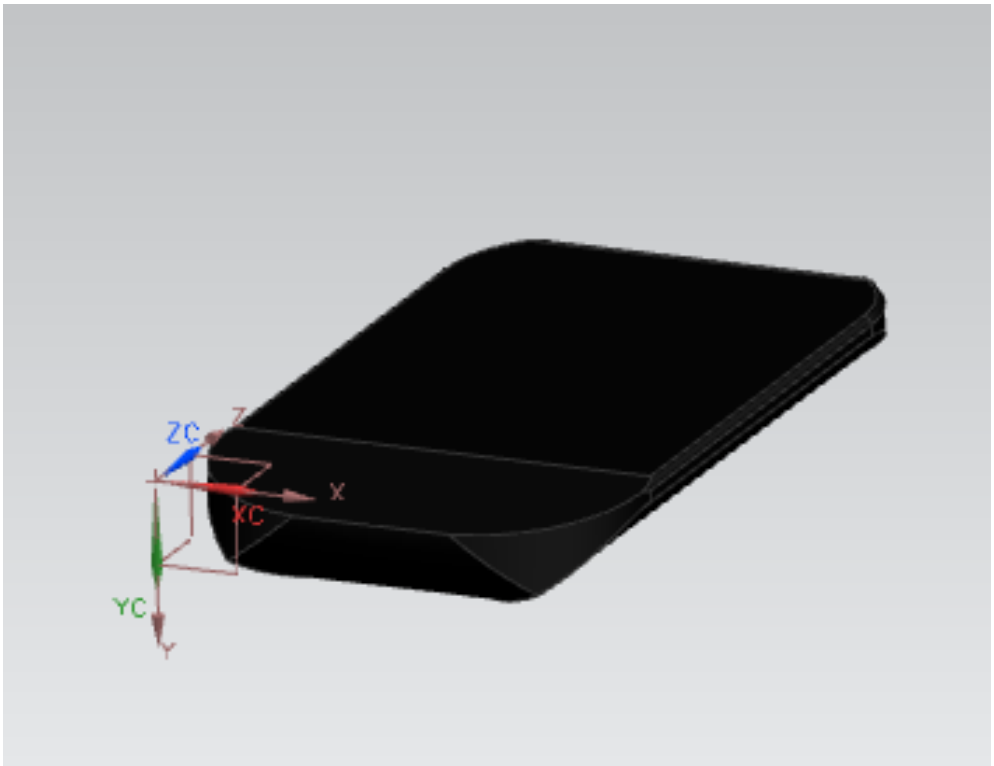
LED que se utilizará en el vehículo.



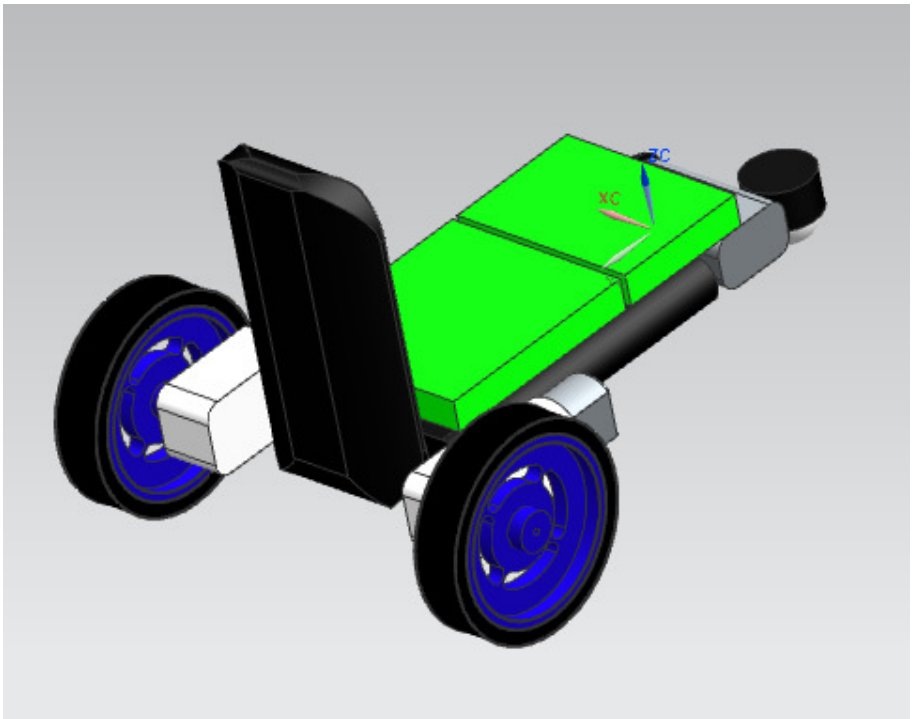
Placa del circuito eléctrico de 62.3 x 44.6 mms.



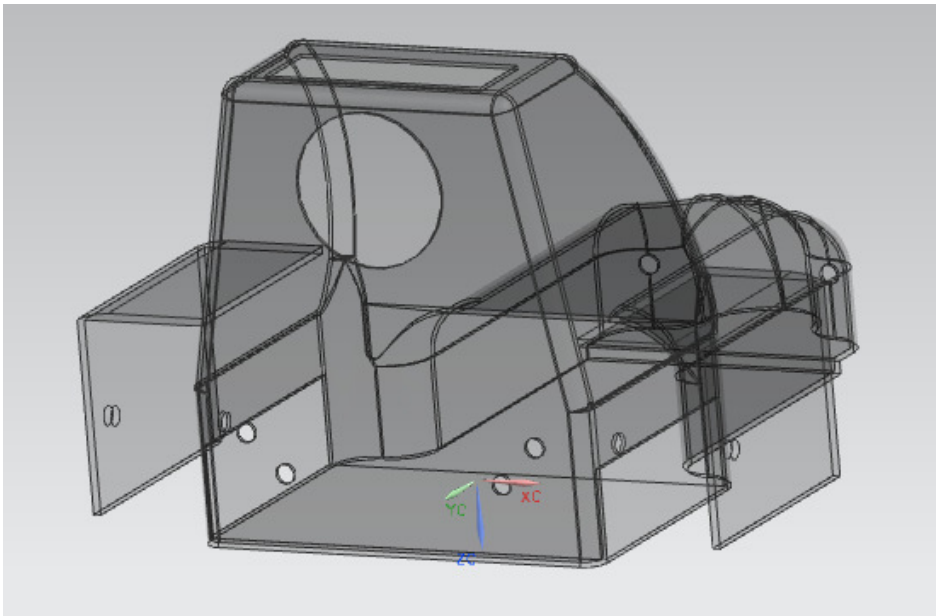
Placa del circuito eléctrico de 62.3 x 67.6 mms.



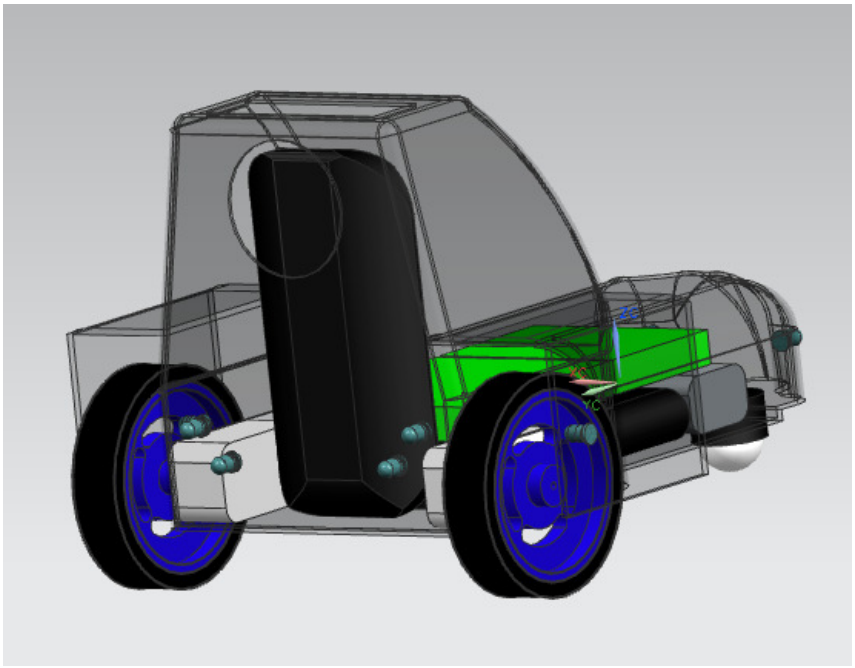
Teléfono celular Samsung GT-I5500L



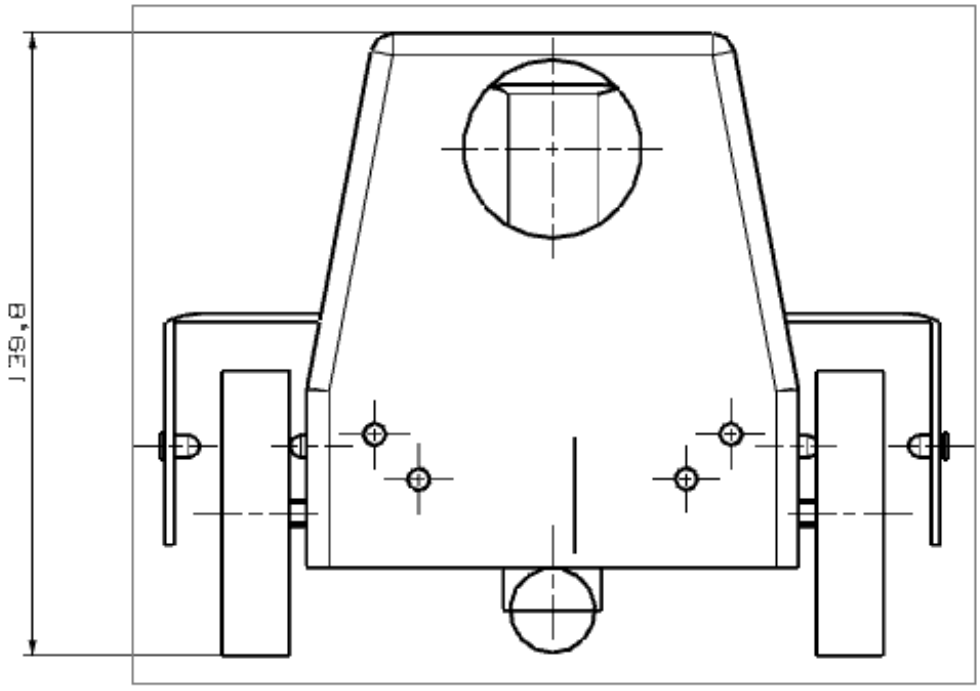
Segundo ensamble de componentes para deducir las dimensiones óptimas del chasis



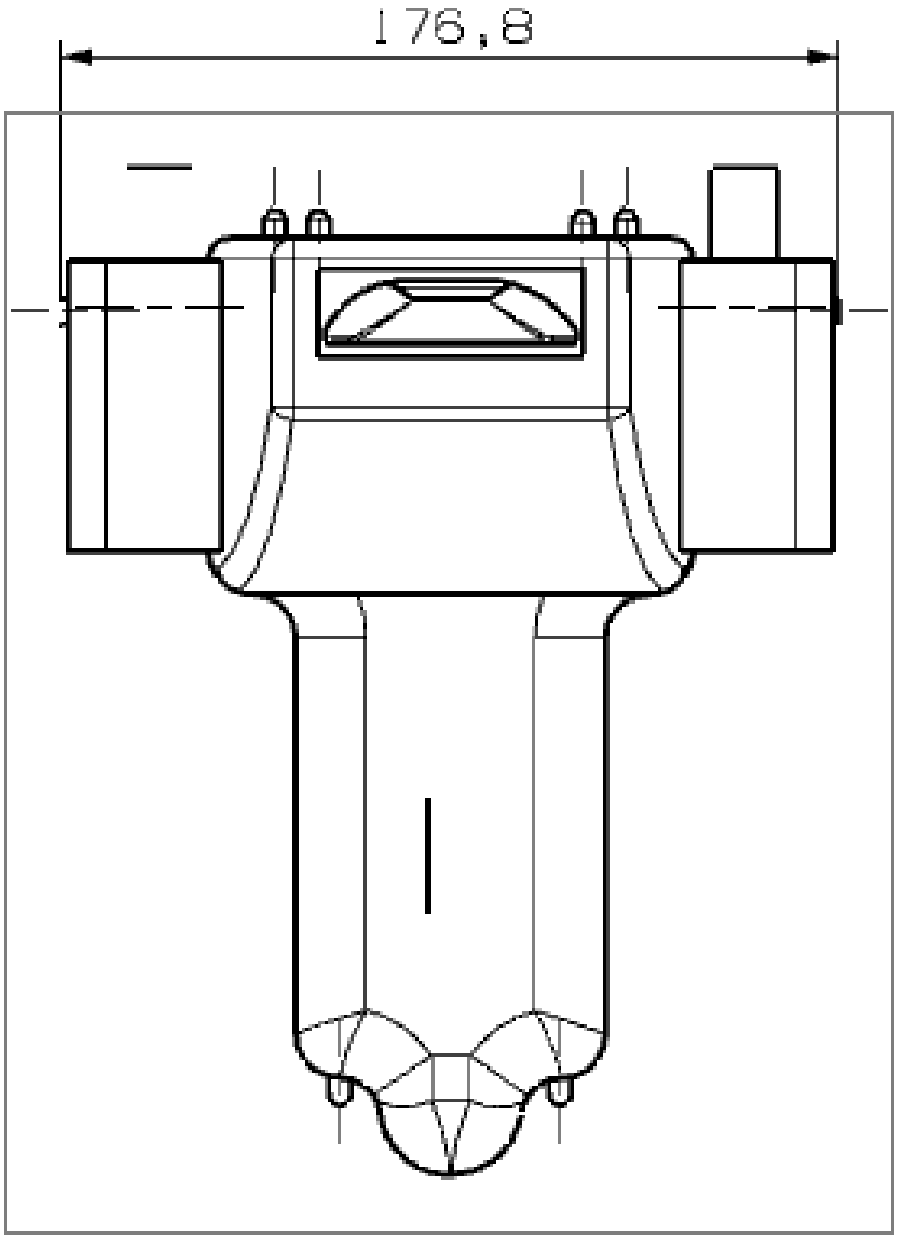
Propuesta final del chasis.



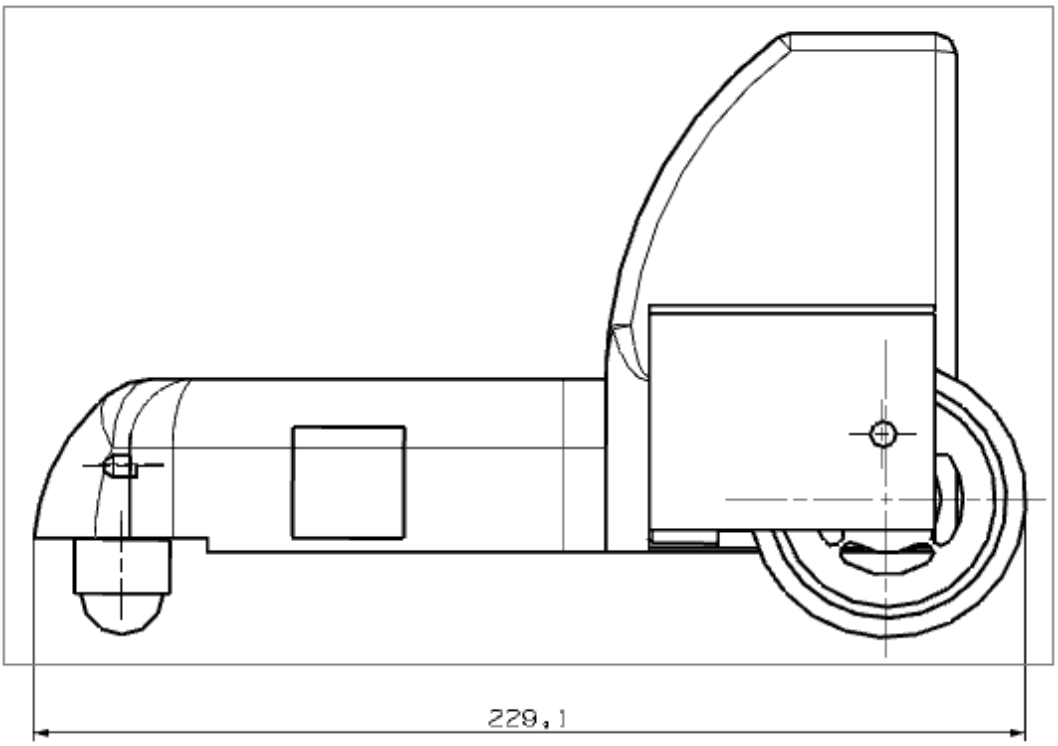
Ensamble completo de la propuesta final del prototipo



Dimensión de la altura completo del vehículo (milímetros).



Dimensión del ancho completo del vehículo (milímetros).



Dimensión del largo completo del vehículo (milímetros).

6.2.2 Selección del método de fabricación

Primeramente se decidió que el tipo de material para la elaboración del chasis sería algún tipo de plástico ya que, en general, es fácil de moldear, ligero, barato, y puede brindar un mejor aspecto a nuestro vehículo.

Después se eligió el proceso de tratamiento del plástico para poder fabricar el prototipo. Debido a la forma del prototipo, varios procesos quedaron descartados por ser exclusivos para características particulares. Dentro de los que más se adecuaron están el método de inyección y el de termoformado.

En el proceso de moldeo por inyección, se deposita cierta cantidad de material plástico en la cavidad inferior de un molde (preferentemente de aluminio) la cual es calentada, al igual que la parte superior del molde, se cierran las dos partes del molde ejerciendo una presión adecuada por medio de un pistón hidroneumático esperando a que la pieza se endurezca y finalmente se obtiene la pieza.

En el de termoformado, el proceso consiste en colocar una lámina de material plástico prensada en un marco, reblandecerla por medio de calor y obligarla a tomar la forma del molde. Este último paso se logra de distintas maneras como empleando aire a presión, generando vacío, de manera mecánica o mediante la combinación de alguno de los métodos anteriores.

Existe el moldeado al vacío (Fig 6-3) en el que se baja el marco con la lámina reblandecida y se aplica succión del otro lado para que la pieza tome la forma del molde. Las máquinas para realizar esta técnica son muy baratas.

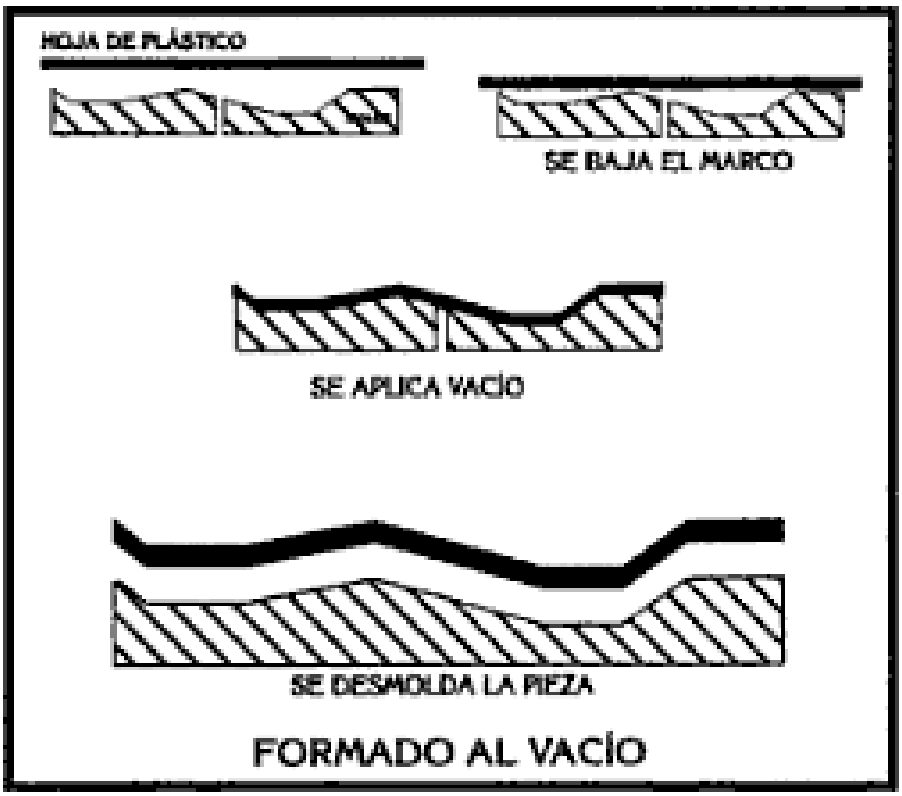


Fig 6-3 Representación del formado al vacío

Está también el moldeado por presión (Fig 6-4) el cual es el método inverso al del vacío: el aire presiona sobre la parte superior de la hoja contra el molde.

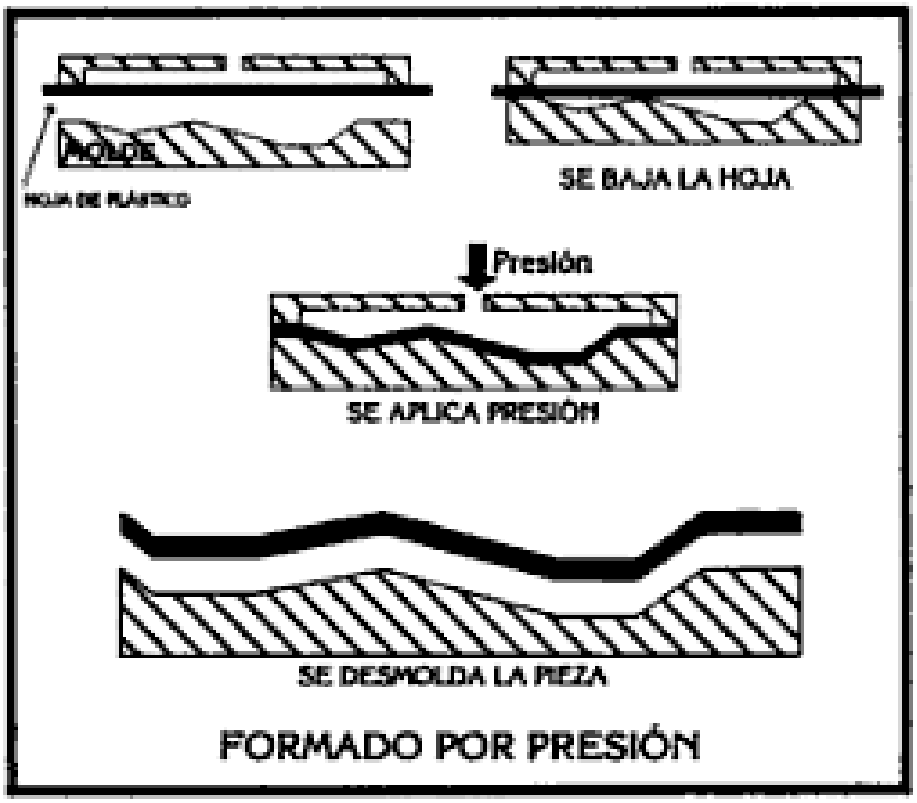


Fig 6-4 Representación del formado a presión

Se seleccionó el método de termoformado por vacío, ya que el método de inyección necesita un molde preferentemente de aluminio lo cual elevaría el costo por la fabricación del prototipo (considerando que inicialmente solo se hará una pieza) y en el de termoformado se pueden emplear moldes de madera lo cúal es mucho más barato. Además, de no conseguir la máquina de termoformado prestada, se puede elaborar una máquina de termoformado al vacío casera, lo que sería más dificil si se tratara de una máquina de inyección.

Los plásticos más utilizados para el proceso de termoformado son: PAI, PET (*Poli Etilén Tereftalato)*, ABS, PEAD, PVC.

Para poder crear el chasis facilmente, se decidió dividirlo en tres secciones:

- Salpicaderas (Fig 6-5)
- Parte superior (Fig 6-6)
- Parte inferior (Fig 6-7)

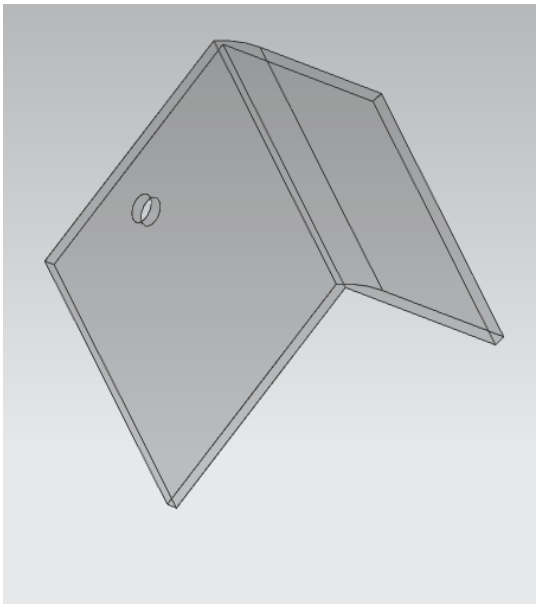


Fig 6-5 “Salpicadera” (2 piezas)

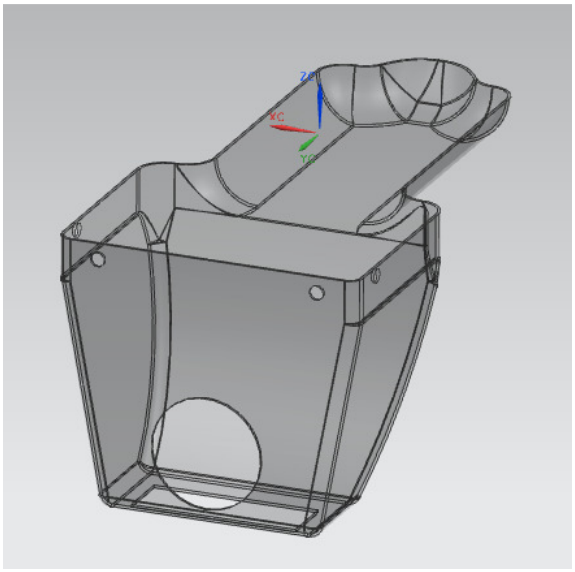


Fig 6-6 Parte superior (1 pieza)

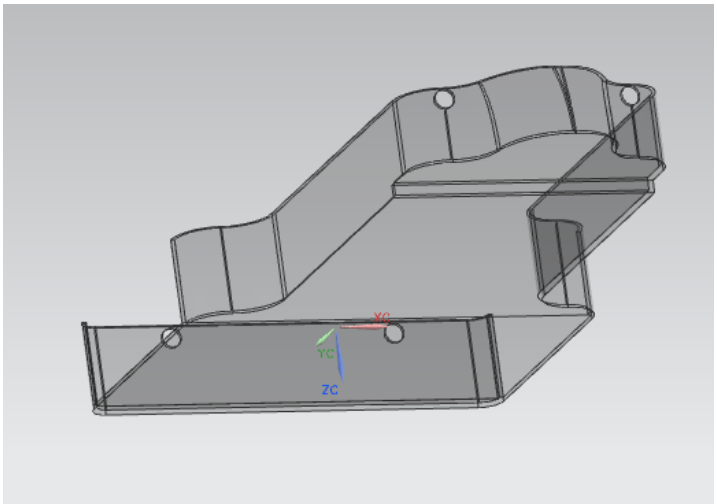


Fig 6-7 Parte inferior (1 pieza)

Para lograrlo, se creará un molde para cada tipo de pieza:

- Molde para salpicaderas (Fig 6-8)
- Molde para la parte inferior (Fig 6-9)
- Molde para la parte superior (Fig 6-10)

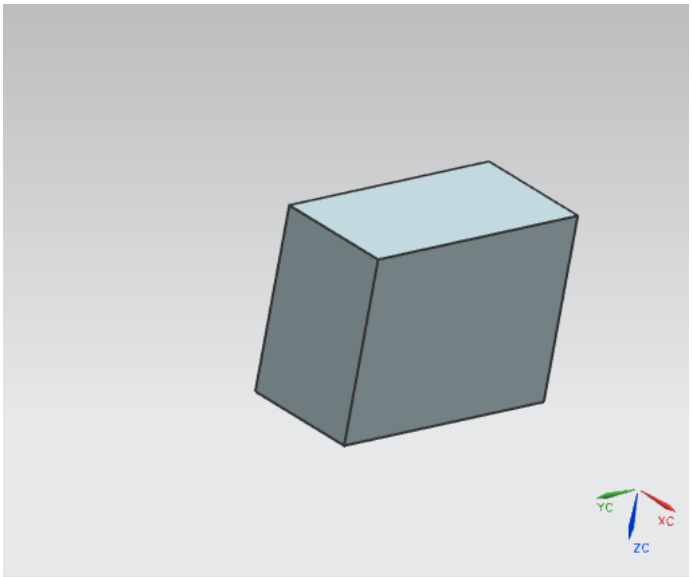


Fig 6-8 Molde para las “salpicaderas”

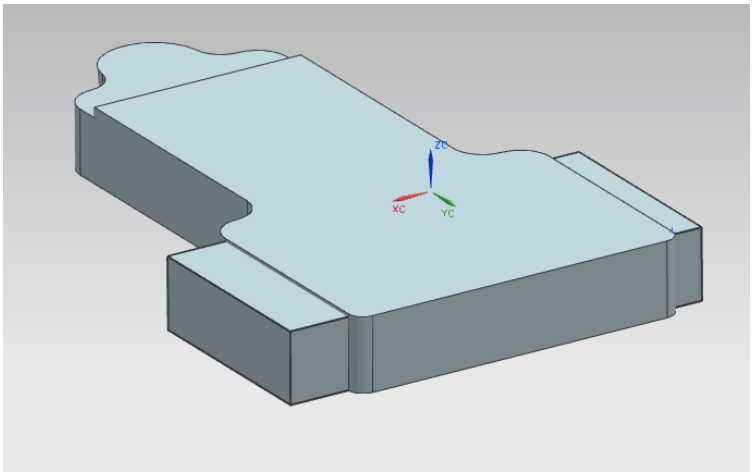


Fig 6-9 Molde para la parte inferior

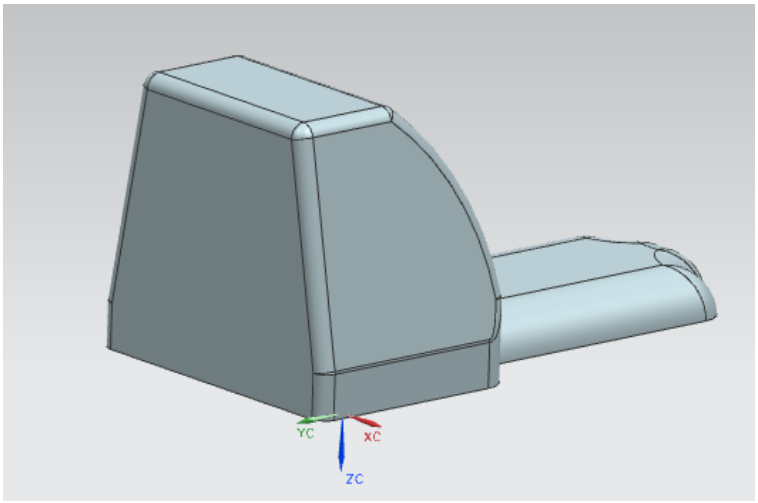


Fig 6-10 Molde para la parte superior

Cada uno de los moldes se construirá a partir de bloques de madera debido a que es más económico (como se mencionó anteriormente), es fácil de modelar y no requiere soportar grandes presiones ni altas temperaturas.

Para el molde de las salpicaderas, se tienen las siguientes dimensiones:

- Ancho: 32.6 mms. (Fig 6-11)
- Alto: 66.1 mms. (Fig 6-11)
- Largo: 50 mms. (Fig 6-12)

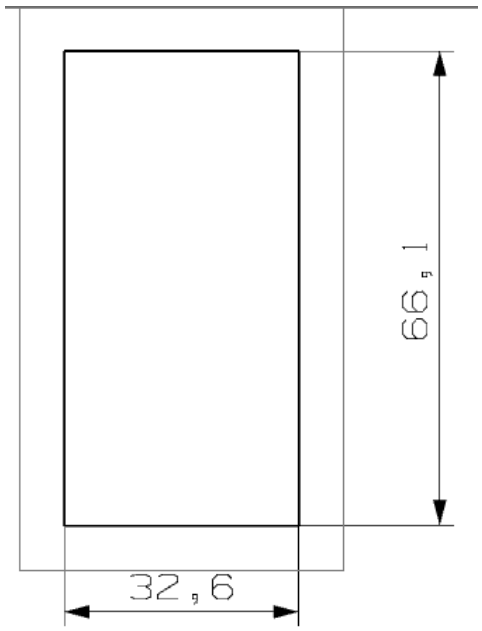


Fig 6-11 Dimensiones de altura y anchura que requiere el molde para las “salpicaderas”

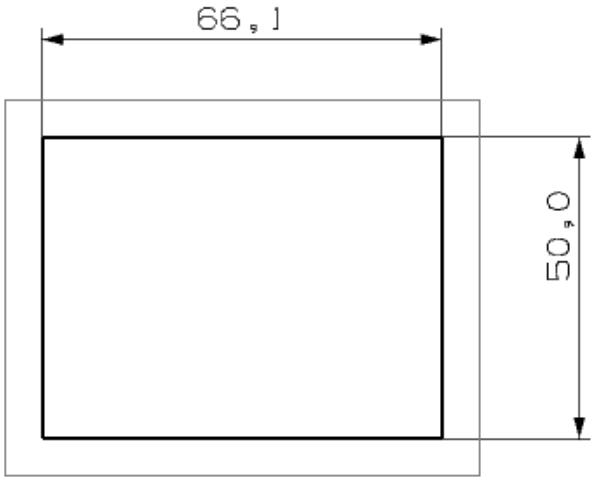


Fig 6-12 Dimension de largo que requiere el molde para las “salpicaderas”

Para el molde de la parte inferior, se pueden observar la Figura 6-13 y 6-14 (todas las dimensiones están en milímetros):

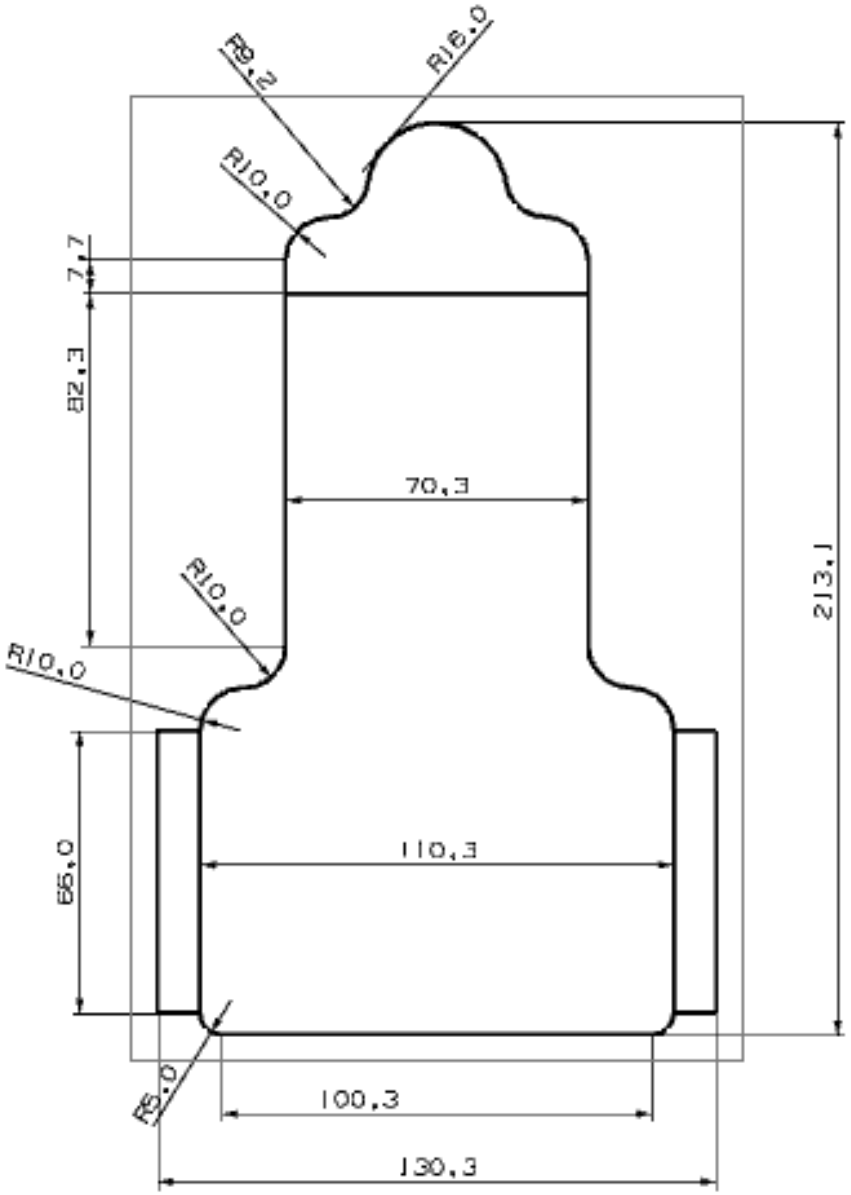


Fig 6-13 Dimensiones para el molde de la parte inferior

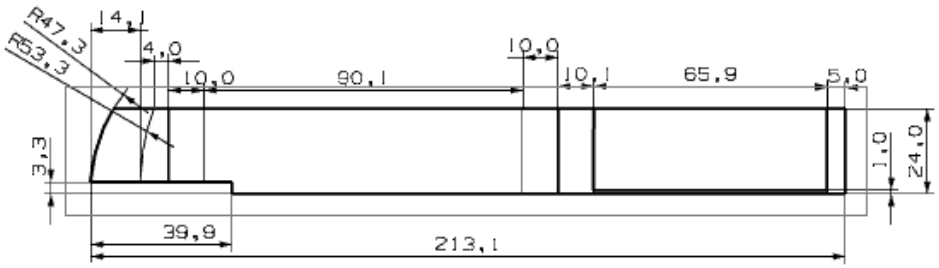


Fig 6-14 Dimensiones para el molde de la parte inferior

Para el molde de la parte superior, se pueden observar las figuras 6-15 y 6-16 (todas las dimensiones están en milímetros):

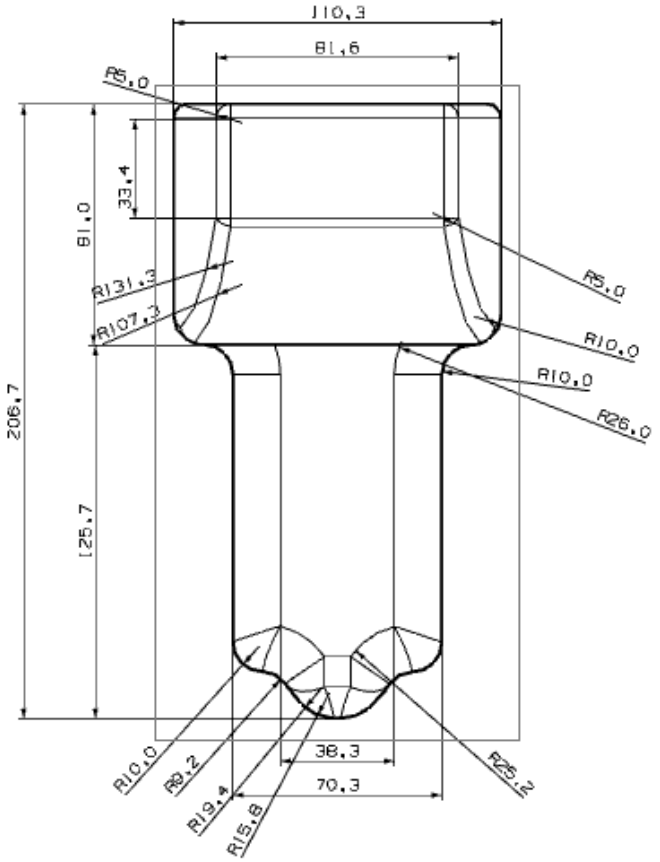


Fig 6-15 Dimensiones para el molde de la parte superior

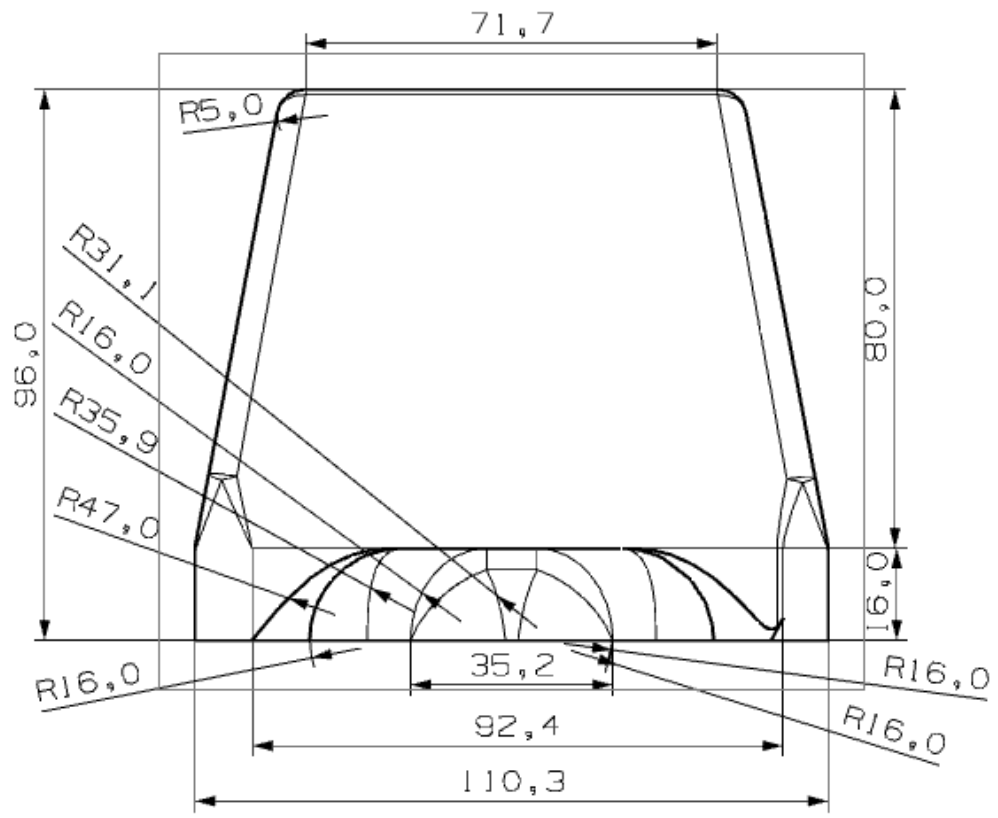


Fig 6-16 Dimensiones para el molde de la parte superior

6.3 Transferencia de video

Como se ha dicho antes, en lo que se hace referencia a la imagen que se transfiere de un dispositivo a otro, se hace mediante la toma de fotos secuenciales que se guardan, se envían y se abren en el otro dispositivo. La velocidad máxima entre captura de fotos y guardarlas en un dispositivo es de 800 milisegundos; las limitantes a las que nos enfrentamos fueron que, al ser una imagen de una calidad bastante aceptable, es el mejor tiempo que obtuvimos para poder hacer el proceso de tomar la foto y guardarla en la ruta deseada sin que la aplicación colapse.

Esta no fue nuestra mayor limitante en cuanto al tiempo: el intervalo de tiempo que realmente nos limita es el que hay entre mandar la foto desde el primer dispositivo, guardarla en el segundo y abrirla en la pantalla del mismo. Este tiempo es de 1300 milisegundos. Dentro de este tiempo, se establece la conexión por medio de la IP y del puerto usados por los dispositivos dentro de la red local, se envía/reciba la imagen, se guarda en el celular de control, se decodifica el archivo de JPEG a mapa de bits, se rota la imagen para que la imagen se ponga en posición horizontal al igual que el teléfono y

finalmente se despliega en la pantalla. Este método se logró establecer a partir de ejemplos encontrados en la bibliografía, donde por separado encontramos como capturar imágenes y guardarlas, como establecer comunicación a través de diferentes vías (Wi-Fi o Bluetooth), como mandar cadenas de texto o archivos guardados de un dispositivo a otro y como abrir imágenes que se tengan que ir cambiando.

Para usar el modo de video de la cámara, los ejemplos de códigos consultados nos limitan a solo poder utilizar el archivo de video hasta que se haya guardado completamente, es decir, hasta que se deje de grabar y no se modifique el archivo. Se pudiera manejar el uso de varios videos como se está realizando el manejo de las imágenes: tomar video durante cierto intervalo de tiempo, guardarlo y enviarlo, pero el archivo de video es mucho más pesado que una imagen y demoraría más en mostrarnos lo que realmente sucede frente al vehículo. Existen aplicaciones dentro del Market de Android que logran esto por lo que no es imposible como la IP Webcam para poder transmitir el video en streaming y el IP Cam Viewer para poder abrir dicho streaming en otro dispositivo. El problema es que, al ser aplicaciones con un costo dentro del Market y poder obtener dinero de ellas, es muy difícil obtener algún ejemplo útil y poder desarrollar algo similar que nos pueda funcionar.

El objetivo sería lograr comprender lo que hacen estas aplicaciones y poder desarrollarlas de manera similar para que nuestro vehículo contenga video en streaming y sea más interactivo el manejo de este.

CAPÍTULO 7. RESULTADOS

Se logró construir con éxito un prototipo funcional de acuerdo a las necesidades que se establecieron en el capítulo 2 del trabajo. Dentro de la parte del circuito eléctrico, como resultado se obtuvo una tarjeta que funcionara sin errores, transmitiera y recibiera datos al dispositivo controlador (Fig 7-1).

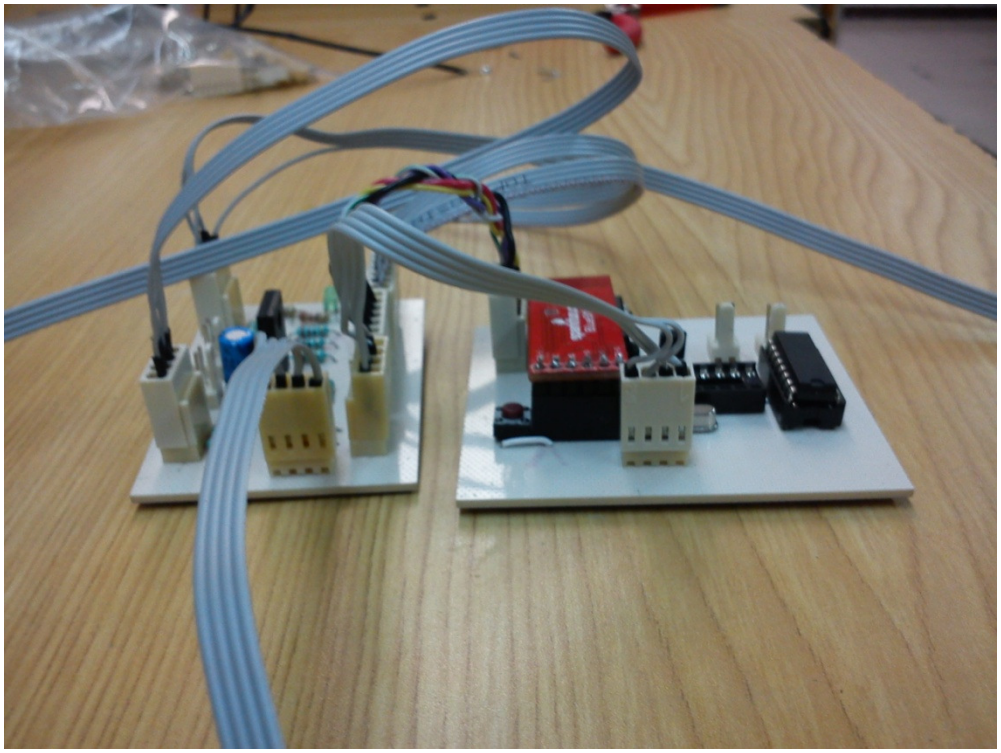


Fig 7-1 Circuito empleado en el prototipo funcional final.

En la figura anterior se ven los conectores y el módulo de Bluetooth ya en la posición donde van. Se usó cable plano para las señales y alambre de calibre 18 para los voltajes. El resultado fue exitoso y no se presento ruido ni interferencia en el flujo de las mismas. Este circuito se instaló sobre la batería recargable y se siguió la configuración de los elementos de acuerdo al diseño CAD. Como resultado el prototipo funcional tiene unas dimensiones de 25 x 17.5 cm. La trasmisión de datos no presentó ningún problema.

El mayor conflicto que se presentó fue la maniobrabilidad del vehículo ya que el cambio de los sensores internos del celular era muy rápido. Para resolverlo se establecieron valores acotados dentro de la aplicación de Android, es decir, no se utiliza el valor máximo

de PWM; en vez de ello se aumenta hasta llegar a la mitad del valor y sólo en caso de una vuelta a máxima velocidad el valor llega a ser el máximo.

Refiriéndonos a la aplicación, la velocidad máxima obtenida para el envío y recepción de fotos fue de 1.3 segundos, lo cual implica un retraso y no una visualización en tiempo real. El tiempo de duración de la batería fue de una hora y media con ambas conexiones encendidas.

Otro resultado favorable fue que se creó una red local para la interacción entre ambos equipos, lo cual significa un costo nulo al no utilizar el tráfico de datos que ofrecen las compañías. Sin embargo al hacer esto estamos sujetos al alcance de ambas tarjetas de red inalámbricas.

Finalmente el prototipo cumple con su finalidad de ser un vehículo de exploración controlado inalámbricamente por dispositivos con plataforma Android. El uso de las dos comunicaciones inalámbricas, WiFi y Bluetooth, presentó un reto pero se pudo solventar de buena manera haciendo que ambas funcionaran al mismo tiempo en beneficio del proyecto.

Imágenes del prototipo:

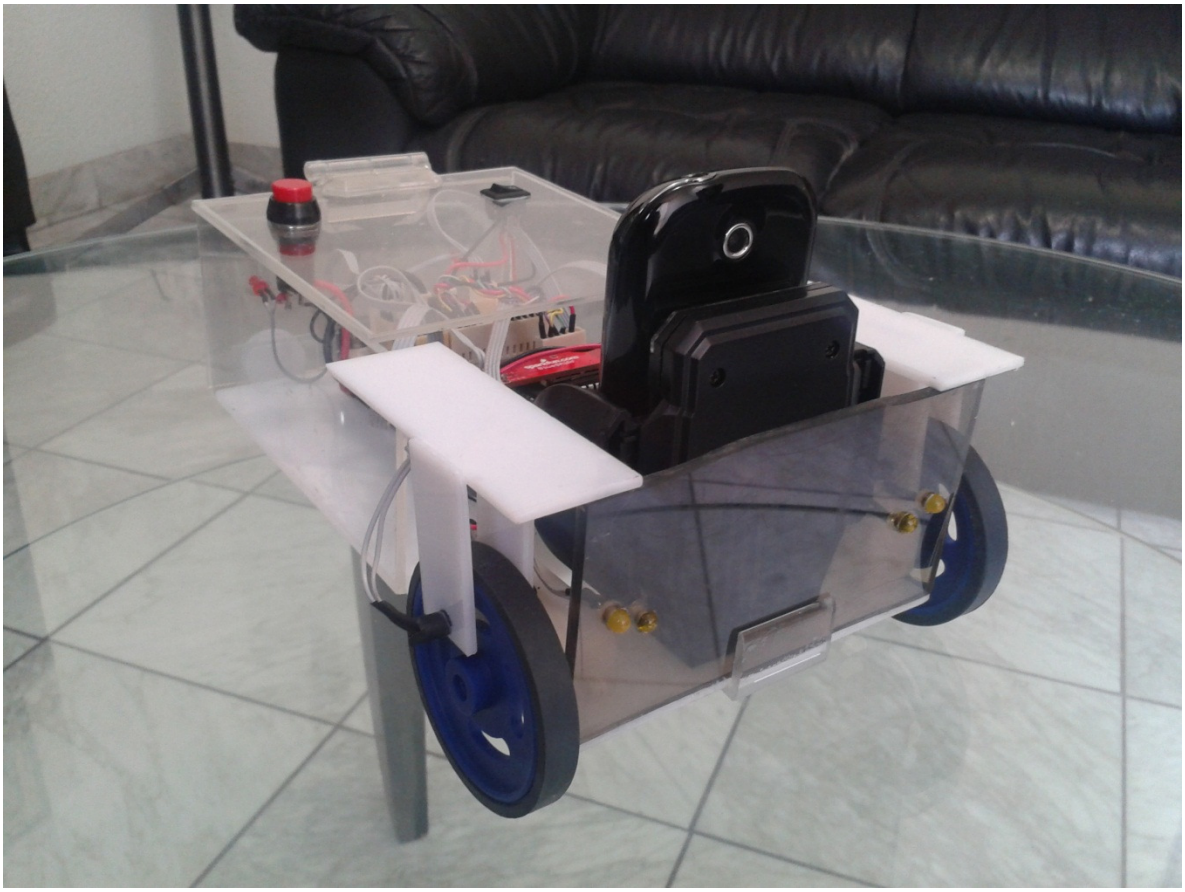


Fig 7-2 Prototipo funcional terminado

Para poder hacer un prototipo funcional sobre nuestra investigación de la manera más sencilla, económica y rápida posible, se optó por crearlo a base de acrílico.

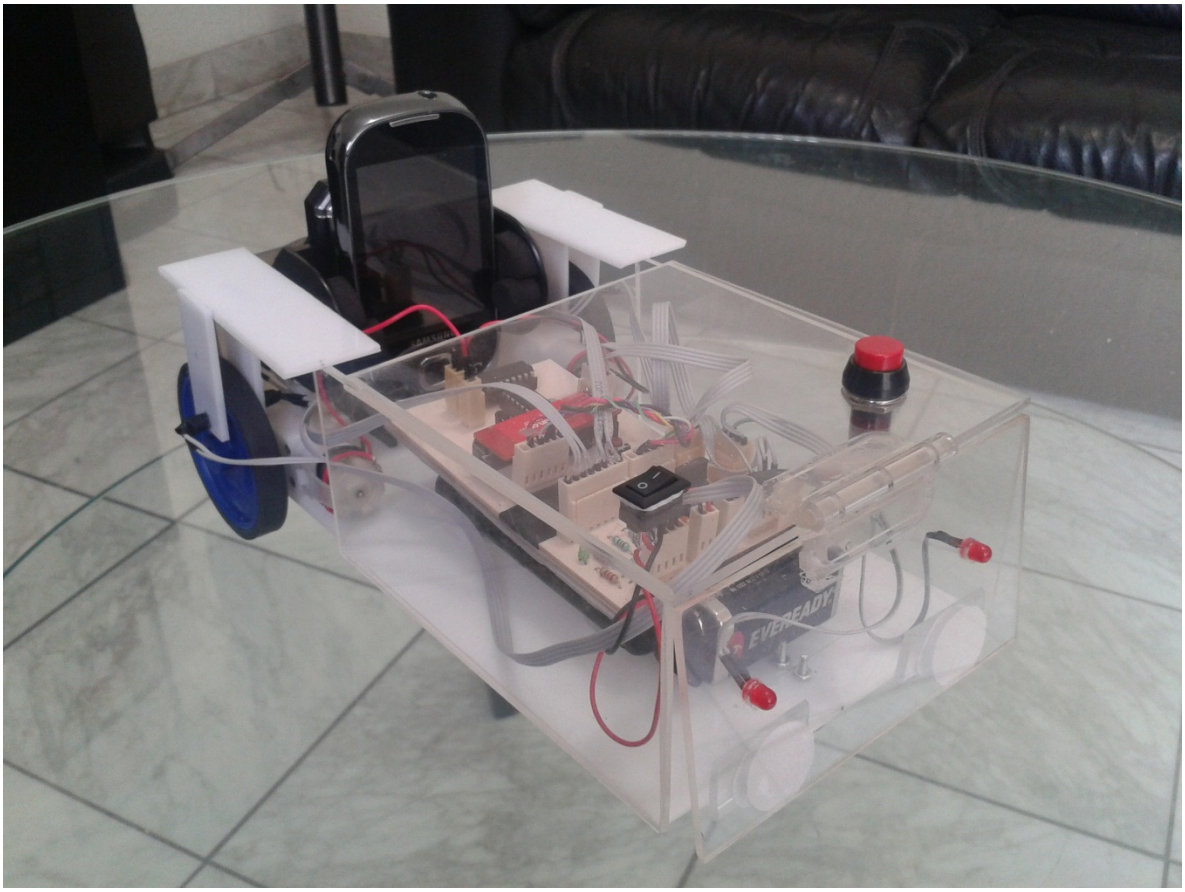


Fig 7-3 otra toma del prototipo funcional

Se unieron las diferentes secciones de acrílico con pegamento acrílico. Para fijar el circuito, la batería empleada para alimentar los motores y la tapa trasera del vehículo se empleó velcro. Otros componentes como los motores, los LEDs, la base para el celular y los botones fueron sujetados con ayuda de adhesivo cianocrilato. La pila de 9V fue fijada con cinta doble cara ya que se tiene que estar cambiando constantemente por ser desechable.

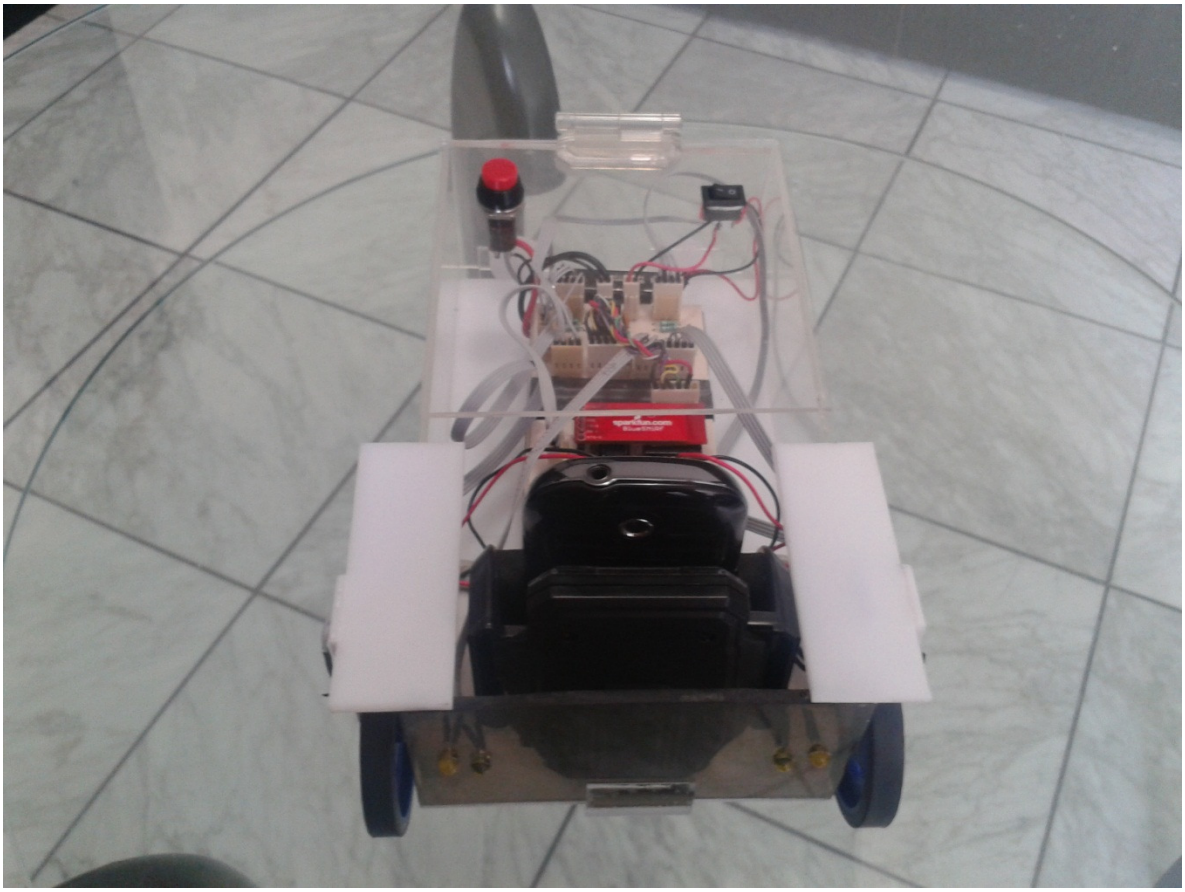


Fig 7-4 Vista desde la parte superior frontal

Las barras blancas de acrílico colocadas en la parte superior del chasis del carro nos permiten evitar interferencias que se puedan producir por focos situados en los techos sobre los sensores optoelectrónicos.

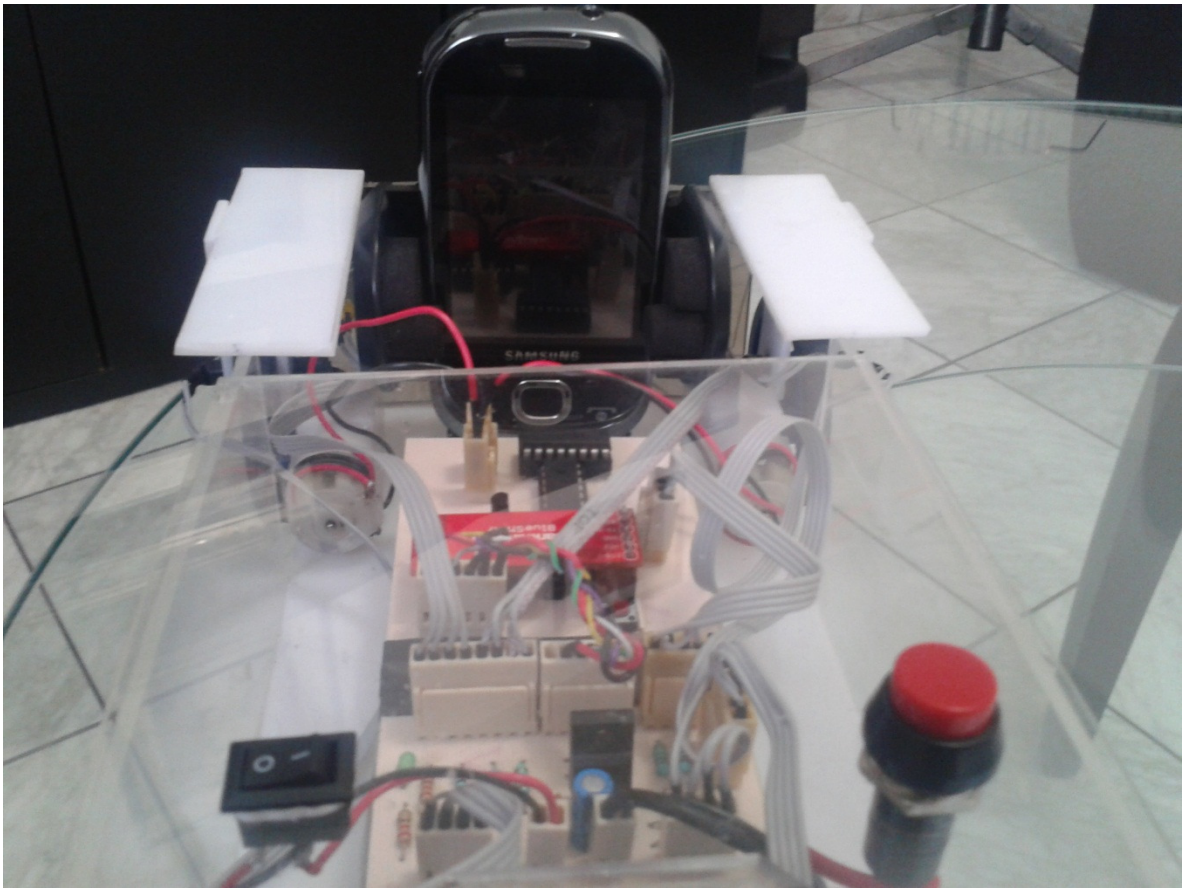


Fig 7-5 Vista desde la parte superior trasera

Como bien se puede apreciar en la figura 7-5, se colocaron dos botones: el del lado izquierdo nos permite cortar la corriente que alimenta la parte lógica del robot. El del lado derecho sirve para interrumpir la corriente que se dirige a los motores. Este botón resulta de una importancia relevante ya que, al ser la parte donde se concentra la mayor parte de la energía eléctrica de nuestro dispositivo, puede ayudarnos a evitar que se dañe algún componente de manera oportuna.

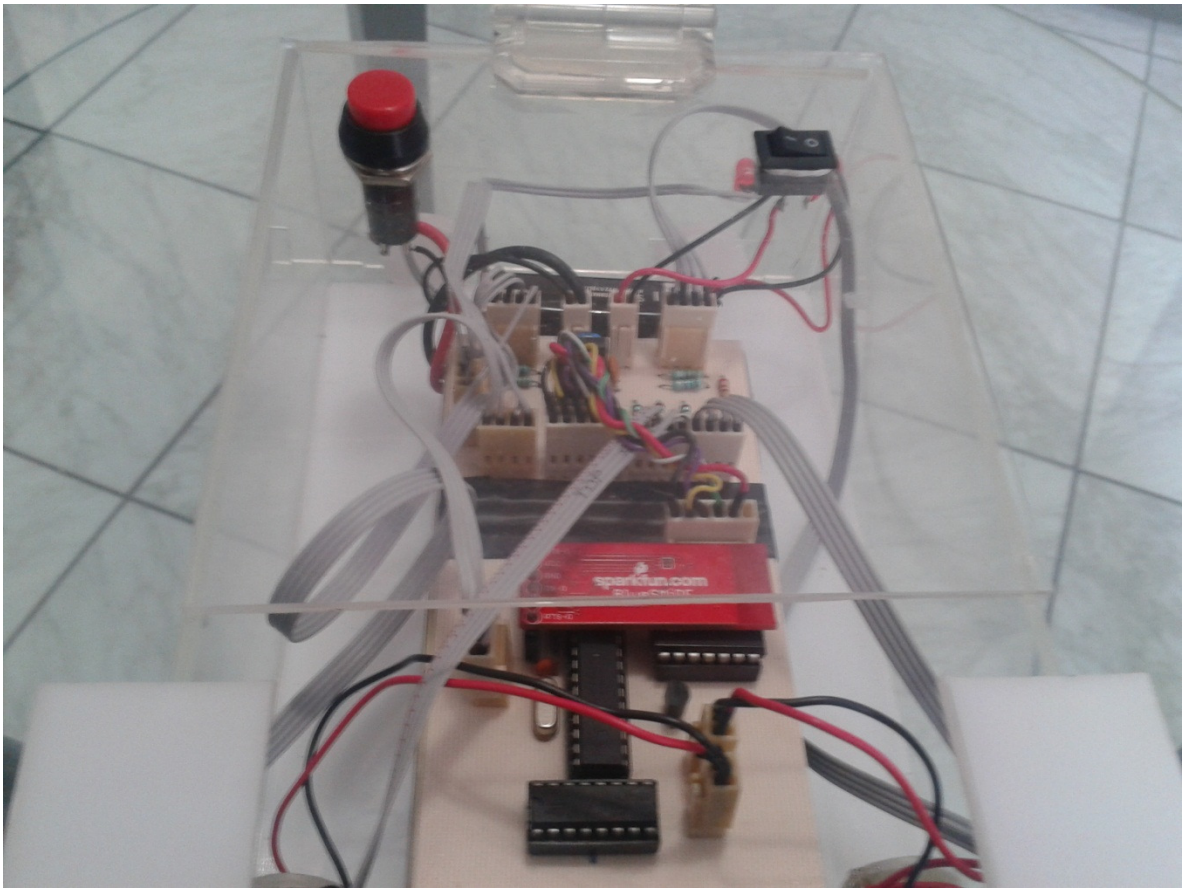


Fig 7-6 Vista de los componentes internos

La parte que rodea al circuito eléctrico se procuró hacer de acrílico transparente para poder apreciar con detalle los componentes del vehículo.

Fig 7-7 Vista frontal

En la figura 7-7 podemos apreciar cuatro LEDs que nos ayudaran en la iluminación al momento de acceder a zonas con baja intensidad luminosa a la vez que nos indicarán que el carro se mueve hacia adelante.

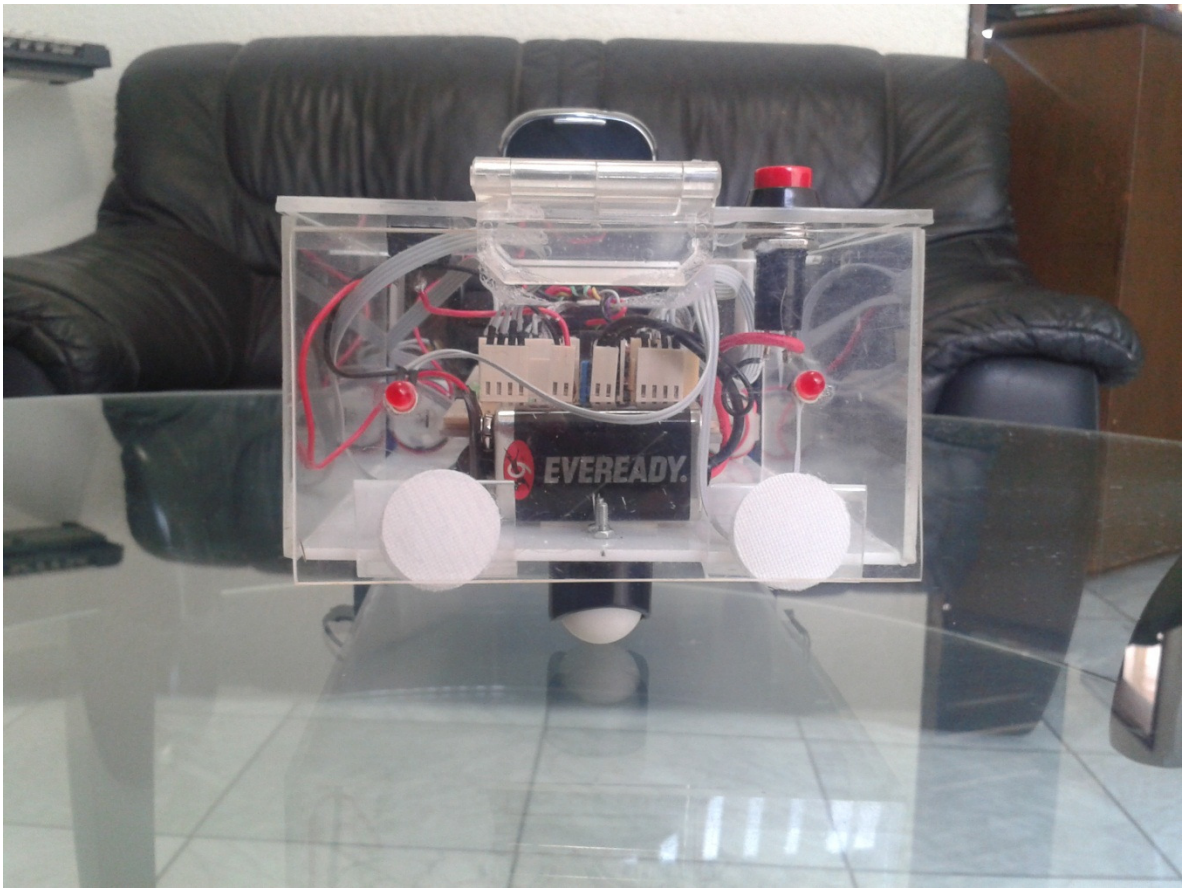


Fig 7-8 Vista trasera

En la figura 7-8 podemos apreciar los LEDs colocados en la parte posterior del vehículo que simulan focos de reversa como en un auto normal.

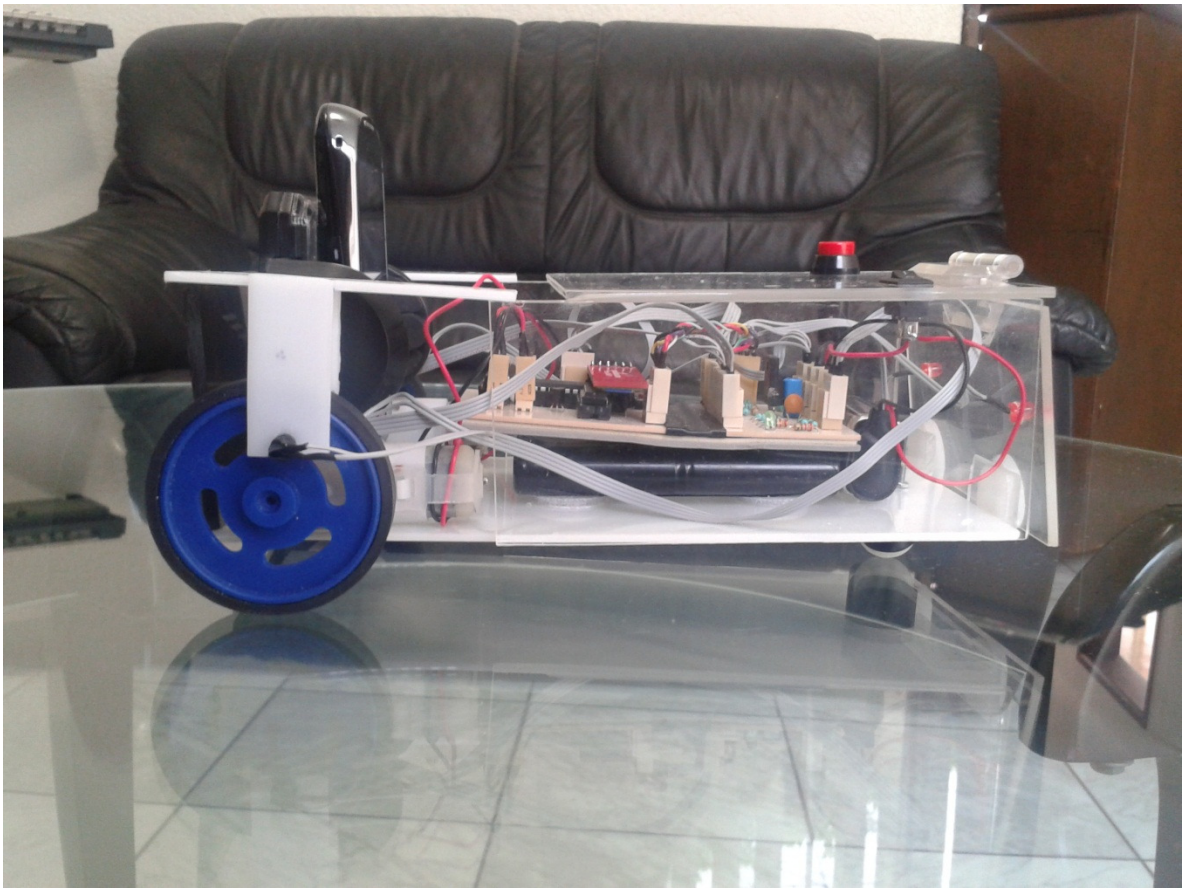


Fig 7-9 Vista lateral

De la figura 7-9 se puede ver que el circuito fue colocado sobre la batería de tracción para poder maximizar el espacio interior.

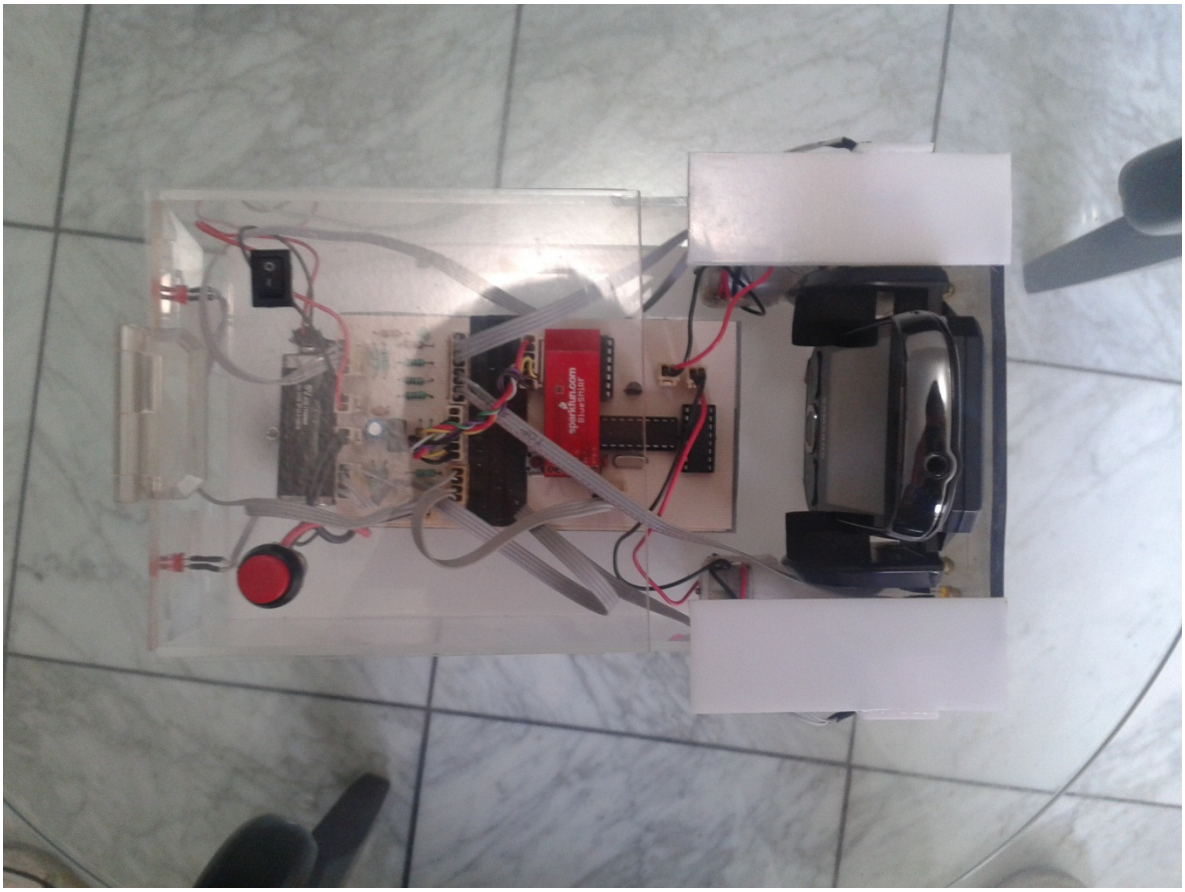


Fig 7-10 Vista superior

Como se puede observar en la figura 7-10, la antena de Bluetooth Bluesmirff es visible y con ello podemos saber el estado de conexión que existe con el teléfono controlador por si el observador se encuentra distante del mando.

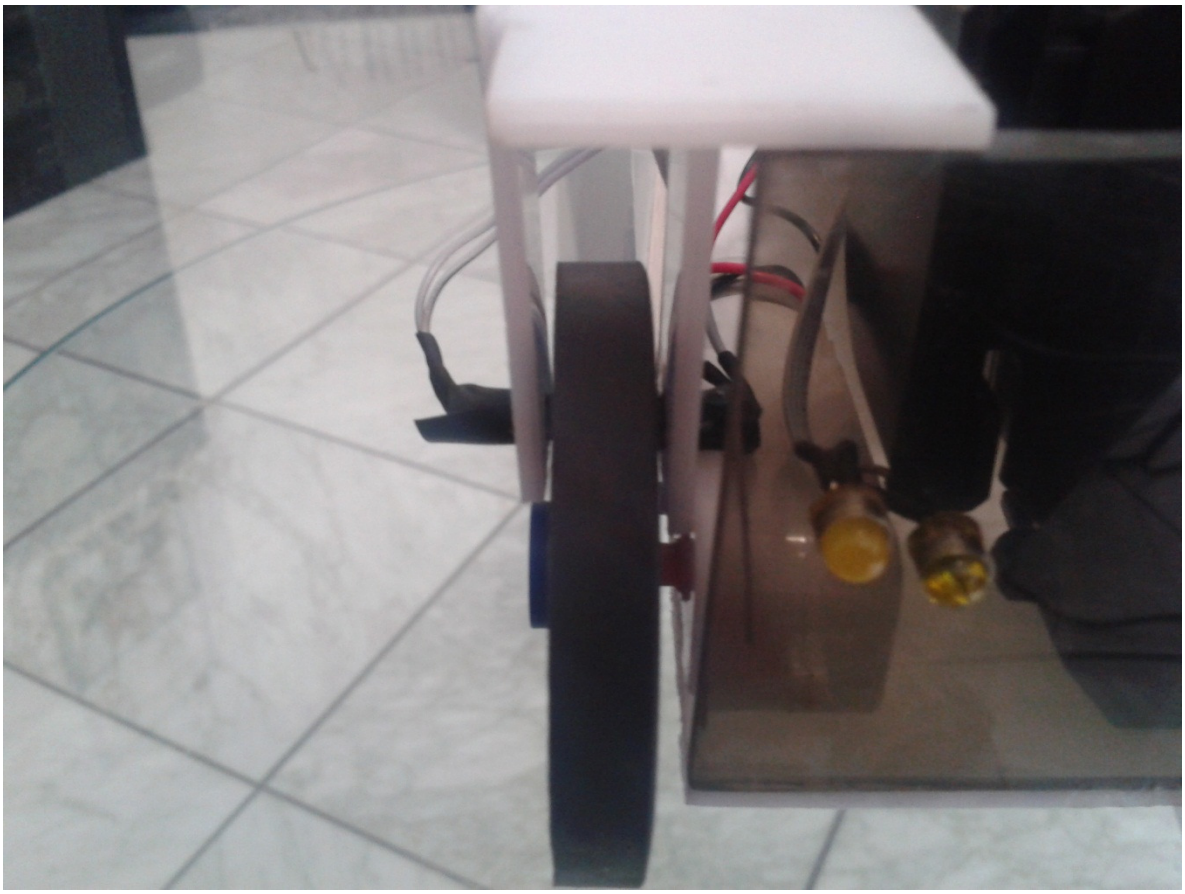


Fig 7-11 salpicadera diseñada para soportar sensor y LED

También los soportes donde fueron colocados el LED y el sensor optoelectrónico fueron construidos con acrílico blanco (Figura 7-11) para evitar en la medida de lo posible interferencia del medio ambiente. El sensor a su vez, fue aislado de manera que solo pueda percibir la luz emitida directamente por el LED.

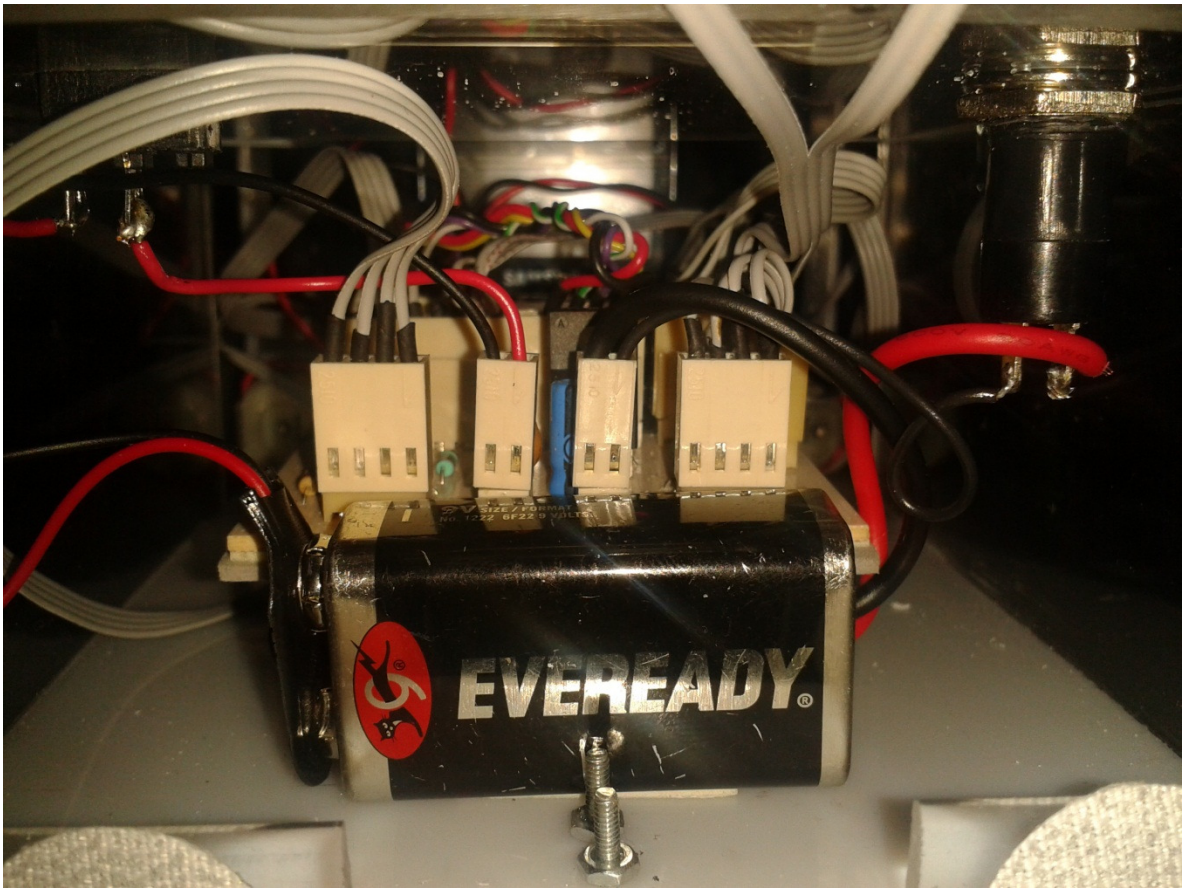


Fig 7-12 Batería de 9V y cables

Las conexiones fueron realizadas con cables de distinto calibre dependiendo de la demanda de corriente y con conectores tipo “Molex” para tener una mejor manipulación de los componentes.

Conclusiones

El Vehículo explorador desarrollado fue manipulado exitosamente con el uso de tecnologías como Bluetooth, Android y Wi-Fi, logrando a su vez que el carro tomara imágenes que se pudieran transmitir para tener un mejor panorama de lo que estaba ocurriendo en el entorno de este cuando se accedía a lugares donde la persona que lo controla no estuviera presente.

La maniobrabilidad que se obtuvo fue bastante buena, ya que se limitó la velocidad de los motores vía PWM para tener un control óptimo sin sacrificar velocidad del vehículo y que a su vez tuviera una respuesta rápida.

Las variables seleccionadas para describir el estado del carro son de gran utilidad, ya que podemos determinar cuándo se sobrecalienta la etapa de potencia y evitar seguir usando el vehículo para no quemar el puente “H” o checar que es lo que está funcionando mal comparando los valores de temperatura ambiental con el de la etapa de potencia, el estado de carga de batería para no usarla cuando tenga un nivel mínimo y así tenga una vida útil más prolongada. La distancia recorrida, velocidad promedio y las revoluciones por minuto nos ayudaron para saber que los sensores en las llantas funcionan de manera adecuada.

ANEXOS

Códigos empleados en el ARDUINO en el capítulo 3.3

1. *Adquisición de datos provenientes del dispositivo móvil por Bluetooth e imprimirlos en la pantalla.*

```
int  serInIndx  = 0;                                  //Índice del arreglo de caracteres.
char serInString[100];                                //Arreglo de 100 caracteres.
int  sb;                                              //Variable para la lectura serial.
int  bt;                                              //Variable para concatenar cada una de las letras
                                                      //provenientes del serial.

void setup()
{
Serial.begin(57600);                                  //Empieza la comunicación serial a 57600bps.
 }
void loop()
{
serInIndx = 0;                                        //Limpia el índice del arreglo para la lectura.
bt = 0;                                               //Inicializa bt = 0.

if( Serial.available() )                              //Pregunta si hay información disponible para
                                                      //leer.
{
while (Serial.available())                            //En este while se van acomodando los valores
                                                      //que llegan del serial en el  arreglo y se
                                                      //incrementa el índice.
{
sb = Serial.read();                                   //El valor del serial es guardado en sb.
serInString[serInIndx] = sb;                          //Guarda sb en el arreglo y asigna el índice.
serInIndx++;                                          //Incrementa el índice.
}

bt = (serInString[0] - 48)*10 + (serInString[1]- 48);  //Se convierten los valores y se despliegan en
                                                       //forma de número.
 Serial.println(bt);                                   //Imprime el valor concatenado en bt.
```

```
 }
delay(50);                                  //Espera para la siguiente lectura 100ms.
 }
```

2. *Movilidad y control de los motores usando los datos provenientes del dispositivo móvil por Bluetooth (PWM).*

```
//variables para el puente H de los motores.//

int PWMen1 = 11;                    //Pin para el enable 1 del PH.
int PWMen2 = 10;                    //Pin para el enable 2 del PH.
int in1a=9;                         //Pin para el input 1 del PH.
int in1b=8;                         //Pin para el input 2 del PH.
int in2a=7;                         //Pin para el input 3 del PH.
int in2b=6;                         //Pin para el input 4 del PH.

//variables para la lectura de los datos serial y el arreglo donde se almacenan.//

byte byteIn = 0;                    //Datos leídos del serial (entradas).
byte data[32];                      //Arreglo para guardar los datos del serial para los dos motores.

//variables para los sentidos de los motores y el PWM que se envía.//

byte motor1Dir;                     //Dirección del motor 1.
byte motor1PWM;                     //PWM para el motor 1.
byte motor2Dir;                     //Dirección del motor 2.
byte motor2PWM;                     //PWM para el motor 2.

void setup()
{
Serial.begin(57600);                //Inicia comunicación serial a 57600.
}

void loop() {
 int indice = 0;                    //Variable para el índice del arreglo de datos. Se inicializa en 0.
```

```
 for (;;) {                          //Ciclo for para la lectura del serial.

 if (Serial.available() > 0) {       //Condición mientras haya datos disponibles en el serial.
 byteIn = Serial.read();             //Se almacenan los datos del serial en la variable byteIn.

 if (byteIn == ';')                  //Si llega un ; por el serial quiere decir que ya tenemos los datos
                                       que queremos.
{
break;                               //Realiza el break al cumplirse la condición del if.
}
else                                 //Continua guardando los datos del serial.
{

data[indice++]=byteIn;               //Se incrementa el índice de la variable data para cada dato que
                                       recibe, de esta forma son de fácil acceso.
}
}
}

//Se establece que función realizarán los datos recibidos por el serial.

motor1Dir = data[0];                 //Dato para la dirección del motor 1.
motor1PWM = data[1];                 //Dato para el PWM del motor 1.
motor2Dir = data[2];                 //Dato para la dirección del motor 2.
motor2PWM = data[3];                 //Dato para el PWM del motor 2.

if (motor2Dir=='F')                          //Si el coche va hacia adelante.
{
digitalWrite(in1a, HIGH );                   ////////////////////////////////////
digitalWrite(in1b, LOW );                    //Con esta configuración del puente H
digitalWrite(in2a, HIGH );                   //los motores van hacia adelante.
digitalWrite(in2b, LOW );                    ////////////////////////////////////

if (motor1Dir == 'L') {                      //Si el coche va hacia adelante/izquierda. analogWrite(PWMen1, motor1PWM+motor2PWM);          //Se
asigna el valor de PWM para el enable 1.
analogWrite(PWMen2, motor2PWM);              //Se asigna el valor de PWM para el enable 2.
}
```

```
if (motor1Dir == 'R') {                          //Si el coche va hacia adelante/derecha.
analogWrite(PWMen2, motor1PWM+motor2PWM);        //Se asigna el valor de PWM para el enable 2.
analogWrite(PWMen1, motor2PWM);                  //Se asigna el valor de PWM para el enable 1.
}
}

if (motor2Dir=='B')                              //Si el coche va hacia atrás.
{
digitalWrite(in1a, LOW );                        ////////////////////////////////////
digitalWrite(in1b, HIGH );                       //Con esta configuración del puente H
digitalWrite(in2a, LOW );                        //los motores van hacia atrás.
digitalWrite(in2b, HIGH );                       ////////////////////////////////////

 if (motor1Dir == 'L') {                         //Si el coche va hacia atrás/izquierda.
analogWrite(PWMen1, motor1PWM+motor2PWM);        //Se asigna el valor de PWM para el enable 1.
analogWrite(PWMen2, motor2PWM);                  //Se asigna el valor de PWM para el enable 2.
}

if (motor1Dir == 'R') {                          //Si el coche va hacia atrás/derecha.
analogWrite(PWMen2, motor1PWM+motor2PWM);        //Se asigna el valor de PWM para el enable 2.
analogWrite(PWMen1, motor2PWM);                  //Se asigna el valor de PWM para el enable 1.
}
}
}
```

3. Adquisición y cálculo de la temperatura con los datos analógicos del sensor LM35DZ.

```
//variables para el sensor de temperatura ambiente.

int tambC = 0;                                   //Variable para la temperatura ambiente en ºC.
int tambPin = 1;                                 //Pin para la entrada analógica del sensor de temperatura.

//variables para el sensor de temperatura de la etapa de potencia.
```

```
int tpotC = 0;                          //Variable para la temp de la etapa de potencia en ºC.
int tpotPin = 2;                        //Pin para la entrada analógica del sensor de temperatura.

void setup()
{
Serial.begin(57600);                    //Inicia comunicación serial a 57600.
}

void loop()
{
tambC = analogRead(tambPin);            //Se hace la lectura del sensor de temperatura.
tambC = (5.0 * tambC * 100)/1024.0;     //Se hace la conversión a ºC.
tpotC = analogRead(tpotPin);            //Se hace la lectura del sensor de temperatura.
tpotC = (5.0 * (tpotC * 100))/1024.0;   //Se hace la conversión a ºC.
Serial.print(tambC);                    //Se imprime el valor de tambC.
Serial.print(" ");                      //Espacio.
Serial.println(tpotC);                  //Se imprime el valor de tpotC.
}
```

4. Adquisición y cálculo del estado de la batería con los datos analógicos provenientes del divisor de voltaje.

```
//variables para el estado de carga de la batería.

int carga = 0;                          //Variable para el estado de carga de la batería.
int voltPin = 0;                        //Pin para la entrada analógica de la batería.

void setup()
{
```

```
Serial.begin(57600);                    //Inicia comunicación serial a 57600.
}

void loop()
{
carga = analogRead(voltPin);            //Lee el voltaje en el pin analógico.
carga = (((5.0 * carga)/1024.0)*25)-25; //Hace la conversión para desplegar el V.
Serial.println(carga);                  //Imprime en la pantalla el valor de carga.
}
```

5. Adquisición y cálculo de la velocidad, revoluciones y distancia recorrida por el vehículo utilizando interrupciones.

```
//variables para los sensores IR y el cálculo de velocidad, revoluciones y distancia de la rueda 1.//

long   time1;                   //Variable para leer el tiempo que lleva la aplicación. Reloj interno.
int    sensor1  = 0;            //Se declara la interrupción en el pin button. Pin 2 o 3 digitales.
int    i = 0;                   //Variable para incrementar el índice de las lecturas.
long   resta1;                  //Variable para hacer las operaciones con el tiempo.
float  vel1;                    //Variable para las operaciones con el tiempo y obtener cm/s.
int    rev1;                    //Variable para las operaciones con el tiempo y obtener rev/min.
long   tiempo1[100];            //Arreglo para guardar los valores del tiempo.
int    x = 0;                   //Variable para medir las lecturas que hace el sensor.
float  dist1;                   //Variable para desplegar la distancia.

//variables para los sensores IR y el cálculo de velocidad, revoluciones y distancia de la rueda 2.//

long   time2;                   //Variable para leer el tiempo que lleva la aplicación. Reloj interno.
int    sensor2  = 0;            //Se declara la interrupción en el pin button. Pin 2 o 3 digitales.
int    j = 0;                   //Variable para incrementar el índice de las lecturas.
```

```
long   resta2;                          //Variable para hacer las operaciones con el tiempo.
float  vel2;                            //Variable para las operaciones con el tiempo y obtener cm/s.
int    rev2;                            //Variable para las operaciones con el tiempo y obtener rev/min.
long   tiempo2[100];                    //Arreglo para guardar los valores del tiempo.
int    y = 0;                           //Variable para medir las lecturas que hace el sensor.
float  dist2;                           //Variable para desplegar la distancia.

void setup() {
attachInterrupt (sensor1, VELChange1, FALLING);  //Agrega la interrupción cada que button pase de 0 a 5.
                                                   Llama a la rutina VELChange1.
attachInterrupt (sensor2, VELChange2, FALLING);  //Agrega la interrupción cada que button pase de 0 a 5.
                                                   Llama a la rutina VELChange2.
Serial.begin(57600);                             //Inicia comunicación serial a 57600.
}

void loop()                             //Dejamos vacío el loop para únicamente ver las interrupciones
{
}

//INTERRUPCIÓN//
void VELChange1() {

if (i<100)                              //Revisa el índice para ver que el arreglo no esté lleno.
{
dist1 = 0;                               //Inicializa la variable distancia en 0.
time1 = millis();                       //Lee el tiempo del reloj interno y lo guarda en la variable time.
x++;                                    //Incrementa el valor de x cada que da una vuelta.
tiempo1[i] = time1;                     //Se asigna time a cada una de las partes del arreglo.

resta1 = tiempo1[i]-tiempo1[i-1];       //Resta da como resultado el tiempo que paso entre una lectura y
                                          la otra.
vel1 = (19.4779/(resta1*5))*1000;       //Calcula la velocidad en cm/s.
rev1 = (60000/(resta1*5));              //Calcula las revoluciones por minuto.
```

```
dist1 = (19.4779*x)/5;                  //Calcula la distancia recorrida sin importar el sentido de giro.
Serial.print("Vel: ");                  //Imprime leyenda.
Serial.print (vel1);                    //Imprime el valor de vel1.
Serial.print ("cm/s  ");                //Imprime leyenda.
 Serial.print(rev1);                    //Imprime el valor de rev1.
Serial.print ("rev/min");              //Imprime leyenda.
Serial.print (dist1);                   //Imprime el valor de dist1.
Serial.println("cm.");                  //Imprime leyenda.
 i++;                                   //Incrementa el valor de i para la siguiente lectura de tiempo.
}
else
{
i = 0;                                  //Cuando se llena el índice lo reinicia.
}
}

//INTERRUPCIÓN 2//
 void VELChange2() {

if (j<100)                              //Revisa el índice para ver que el arreglo no esté lleno.
{
dist2 = 0;                              //Inicializa la variable distancia en 0.
time2 = millis();                       //Lee el tiempo del reloj interno y lo guarda en la variable time.
y++;                                    //Incrementa el valor de y cada que da una vuelta.
tiempo2[i] = time2;                     //Se asigna time a cada una de las partes del arreglo.
resta2 = tiempo2[i]-tiempo2[i-1];       //Resta da como resultado el tiempo que paso entre una lectura y
                                          la otra.
vel2 = (19.4779/(resta2*5))*1000;       //Calcula la velocidad en cm/s.
rev2 = (60000/(resta2*5));              //Calcula las revoluciones por minuto.
dist2 = (19.4779*y)/5;                  //Calcula la distancia recorrida sin importar el sentido de giro.
Serial.print("Vel: ");                  //Imprime leyenda.
Serial.print (vel2);                    //Imprime el valor de vel2.
```

```
Serial.print ("cm/s  ");                    //Imprime leyenda.
Serial.print(rev2);                         //Imprime el valor de rev2.
Serial.print ("rev/min");                   //Imprime leyenda.
 Serial.print (dist1);                      //Imprime el valor de dist2.
Serial.println("cm.");                      //Imprime leyenda.
j++;                                        //Incrementa el valor de j para la siguiente lectura de tiempo.
}
 else
{
 j = 0;                                     //Cuando se llena el índice lo reinicia.
}
}
```

CÓDIGO FINAL

```
//********  Código para controlar el coche (2 motores). Leer velocidad, revoluciones y distancia de 2 ruedas con interrupts.
Leer Tpotencia, Tamb y estado de carga ********//
//********  De igual forma el vehículo se detiene al perder la conexión Bluetooth mediante el uso de un contador. Versión 3 de
la tesis para Ing. Mecatrónica.      ********//
//********  Programmers: FALM, VMMA    Institutión: Universidad Nacional Autónoma de México UNAM, Facultad de Ingeniería FI,
Mecatrónica    Project: AndroidMóvil   ********//
//-------------------------------------------------------------------------------------------------------------------
--------------------------------------//

//DECLARACIÓN DE LAS VARIABLES//

//variables para los sensores IR y el cálculo de velocidad, revoluciones y distancia de la rueda 1.//
long   time1;             //Variable para leer el tiempo que lleva la aplicación. Reloj interno.
int    sensor1  = 0;      //Se declara la interrupción en el pin button. Pin 2 o 3 digital.
int    i = 0;             //Varible para incrementar el índice de las lecturas inicializado en 0.
long   resta1;            //Variable para hacer las operaciones con el tiempo.
float  vel1;              //Variable para hacer las operaciones con el tiempo y obtener cm/s.
int    rev1;              //Variable para hacer las operaciones con el tiempo y obtener rev/min.
long   tiempo1[100];      //Arreglo para cuardar los valores del tiempo.
int    x = 0;             //Variable para medir las lecturas que hace el sensor.
float  dist1;             //Variable para desplegar la distancia.

//variables para los sensores IR y el cálculo de velocidad, revoluciones y distancia de la rueda 2.//
long   time2;             //Variable para leer el tiempo que lleva la aplicación. Reloj interno.
int    sensor2  = 0;      //Se declara la interrupción en el pin button. Pin 2 o 3 digital.
int    j = 0;             //Varible para incrementar el índice de las lecturas inicializado en 0.
long   resta2;            //Variable para hacer las operaciones con el tiempo.
float  vel2;              //Variable para hacer las operaciones con el tiempo y obtener cm/s.
int    rev2;              //Variable para hacer las operaciones con el tiempo y obtener rev/min.
long   tiempo2[100];      //Arreglo para cuardar los valores del tiempo.
int    y = 0;             //Variable para medir las lecturas que hace el sensor.
float  dist2;             //Variable para desplegar la distancia.

//variables para el sensor de temperatura ambiente.
int tambC = 0;           //Variabe para la temperatura en ºC.
```

```
int tambPin = 1;         //Pin para la entrada analógica del sensor de temperatura.

//variables para el sensor de temperatura de la etapa de potencia.
int tpotC = 0;           //Variabe para la temperatura en ºC.
int tpotPin = 2;         //Pin para la entrada analógica del sensor de temperatura.

//variables para el estado de carga de la bateria.
int carga = 0;           //Variable para el estado de carga de la bateria.
int voltPin = 0;         //Pin para la entrada analógica de la bateria.

//variables para el puente H de los motores.//
int PWMen1 = 11;          //Pin para el enable 1 del PH.
int PWMen2 = 10;          //Pin para el enable 2 del PH.
int in1a=9;               //Pin para el input 1 del PH.
int in1b=8;               //Pin para el input 2 del PH.
int in2a=7;               //Pin para el input 3 del PH.
int in2b=6;               //Pin para el input 4 del PH.

//variables para la lectura de los datos serial y el arreglo donde se almacenan.//
byte byteIn = 0;          //Datos leídos del serial (entradas).
byte data[32];            //Arreglo para guardar los datos del serial para los dos motores.

//variables para los sentidos de los motores y el PWM que se envía.//
byte motor1Dir;           //Dirección del motor 1.
byte motor1PWM;           //PWM para el motor 1.
byte motor2Dir;           //Dirección del motor 2.
byte motor2PWM;           //PWM para el motor 2.

//variables para los leds delanteros y traseros.//
int stops=4;              //Pin para los stops.
int focos=5;              //Pin para los focos.


void setup()
{
  pinMode(stops,OUTPUT);
  pinMode(focos,OUTPUT);
  pinMode(in1a,OUTPUT);
```

```
  pinMode(in1b,OUTPUT);
  pinMode(in2a,OUTPUT);
  pinMode(in2b,OUTPUT);
  attachInterrupt (sensor1, VELChange1, FALLING);  //Agrega la interrupción cada que button pase de 0 a 5. Llama a la rutina
VELChange.
  attachInterrupt (sensor2, VELChange2, FALLING);  //Agrega la interrupción cada que button pase de 0 a 5. Llama a la rutina
VELChange.
  Serial.begin(57600);                             //Inicia comunicación serial a 57600.
}

void loop()
{
  int indice;                                      //Variable para el índice del arreglo de datos.
  unsigned long cont = 0;                          //Variable para saber si se perdió la conexión inicializada en 0.
  data[0]= 0;                                      //Se inicializa en 0
  data[1]= 0;                                      //Se inicializa en 0
  data[2]= 0;                                      //Se inicializa en 0
  data[3]= 0;                                      //Se inicializa en 0
  motor1Dir = data[0];                             //Dato para la dirección del motor 1.
  motor1PWM = data[1];                             //Dato para el PWM del motor 1.
  motor2Dir = data[2];                             //Dato para la dirección del motor 2.
  motor2PWM = data[3];                             //Dato para el PWM del motor 2.

  tambC = analogRead(tambPin);                     //Se hace la lectura del sensor de temperatura.
  tambC = (5.0 * tambC * 100)/1024.0;              //Se hace la conversión a ºC.
  tpotC = analogRead(tpotPin);                     //Se hace la lectura del sensor de temperatura.
  tpotC = (5.0 * (tpotC * 100))/1024.0;            //Se hace la conversión a ºC.
  carga = analogRead(voltPin);                     //Lee el voltaje en el pin analógico.
  carga = (((5.0 * carga)/1024.0)*25)-25;          //Hace la conversión para desplegar el V.

  Serial.print(carga);                             ////////////////////////////////////
  Serial.print("/");                               //Datos que se envían al celular por
  Serial.print(tambC);                             //el puerto serial, separados con /
  Serial.print("/");                               //Se despliegan en la pantalla del cel.
  Serial.println(tpotC);                           ////////////////////////////////////

    for (;;)                          //Ciclo for para la lectura del serial.
    {
```

```
    cont++;                          //Aumenta contador cada que repite el ciclo. Si se pierde la conexión sigue contando.
    if (cont > 50000)                //Si contador para 50000 entra en este if. Este if sirve para detener los motores si se
pierde la conexión.
    {
    analogWrite(PWMen1, 0);          //Instrucción para apagar en1.
    analogWrite(PWMen2, 0);          //Instrucción para apagar en2.
    digitalWrite(focos,LOW);         //Los leds de adelante se apagan.
    digitalWrite(stops,LOW);         //Los leds de stop se apagan.
    cont = 0;                        //Reinicia contador.
    break;                           //Sale del ciclo for.
    }
    if (Serial.available() > 0) {    //Condición mientras haya datos disponibles en el serial.
    cont = 0;                        //Reinicia contador.
    byteIn = Serial.read();          //Se almacenan los datos del serial en la variable byteIn.
    if (byteIn == ';')               //Si llega un ; por el serial quiere decir que ya tenemos los datos que queremos.
    {
    indice = 0;                      //Reinicia el índice.
    break;                           //Sale del ciclo for.
    }
    else                             //Continua guardando los datos del serial.
    {
    data[indice++]=byteIn;           //Se incrementa el índice de la variable data para cada dato que recibe, dee esta forma
son de fácil acceso.
    }
    }
    }

  //Se establece que función realizarán los datos recibidos por el serial.
  motor1Dir = data[0];               //Dato para la dirección del motor 1.
  motor1PWM = data[1];               //Dato para el PWM del motor 1.
  motor2Dir = data[2];               //Dato para la dirección del motor 2.
  motor2PWM = data[3];               //Dato para el PWM del motor 2.

  if (motor2Dir=='F')                          //Si el coche va hacia adelante.
  {
    digitalWrite(focos,HIGH);                  //Los leds de adelante se encienden.
    digitalWrite(stops,LOW);                   //Los leds de stop de apagan.
```

```
  digitalWrite(in1a, HIGH );                    ////////////////////////////////////
  digitalWrite(in1b, LOW );                     //Con esta configuración del puente H
  digitalWrite(in2a, HIGH );                    //los motores van hacia adelante.
  digitalWrite(in2b, LOW );                     ////////////////////////////////////


if (motor1Dir == 'L')                           //Si el coche va hacia adelante/izquierda.
{
  analogWrite(PWMen1, motor1PWM+motor2PWM);     //Se asigna el valor de PWM para el enable 1.
  analogWrite(PWMen2, motor2PWM);               //Se asigna el valor de PWM para el enable 2.
}
if (motor1Dir == 'R')                           //Si el coche va hacia adelante/derecha.
{
  analogWrite(PWMen2, motor1PWM+motor2PWM);     //Se asigna el valor de PWM para el enable 2.
  analogWrite(PWMen1, motor2PWM);               //Se asigna el valor de PWM para el enable 1.
}
}

if (motor2Dir=='B')                             //Si el coche va hacia atrás.
{
  digitalWrite(focos,LOW);                      //Los leds de adelante se apagan.
  digitalWrite(stops,HIGH);                     //Los leds de stop de encienden.

  digitalWrite(in1a, LOW );                     ////////////////////////////////////
  digitalWrite(in1b, HIGH );                    //Con esta configuración del puente H
  digitalWrite(in2a, LOW );                     //los motores van hacia atrás.
  digitalWrite(in2b, HIGH );                    ////////////////////////////////////

if (motor1Dir == 'L')                           //Si el coche va hacia atrás/izquierda.
{
  analogWrite(PWMen1, motor1PWM+motor2PWM);     //Se asigna el valor de PWM para el enable 1.
  analogWrite(PWMen2, motor2PWM);               //Se asigna el valor de PWM para el enable 2.
}
if (motor1Dir == 'R')                           //Si el coche va hacia atrás/derecha.
{
  analogWrite(PWMen2, motor1PWM+motor2PWM);     //Se asigna el valor de PWM para el enable 2.
  analogWrite(PWMen1, motor2PWM);               //Se asigna el valor de PWM para el enable 1.
}
```

```
  }
  }

//INTERRUPCIÓN//
  void VELChange1()
  {
    if (i<100)                        //Revisa el índice para ver que el arreglo no este lleno.
    {
    dist1 = 0;                        //Inicializa la variable distancia en 0.
    time1 = millis();                 //Lee el tiempo del reloj interno y lo guarda en la variable time.
    x++;                              //Incrementa el valor de x cada que da una vuelta.
    tiempo1[i] = time1;               //Se asigna time a cada una de las partes del arreglo.
    resta1 = tiempo1[i]-tiempo1[i-1]; //Resta da como resultdo el tiempo que paso entre una lectura y la otra.
    vel1 = (19.4779/(resta1*5))*1000; //Calcula la velocidad en cm/s.
    rev1 = (60000/(resta1*5));        //Calcula las revoluciones por minuto.
    dist1 = (19.4779*x)/4;            //Calcula la distancia recorrida sin importar el sentido de giro.

    Serial.print("Vel: ");
    Serial.print (vel1);
    Serial.print ("cm/s  ");
    Serial.print(rev1);
    Serial.print ("rev/min");
    Serial.print (dist1);
    Serial.println("cm.");

    i++;                             //Incrementa el valor de i para la siguiente lectura de tiempo.
   }
   else
   {
     i = 0;                         //cuando se llena el índice lo reinicia.
   }
}

//INTERRUPCIÓN 2//
  void VELChange2()
  {
    if (j<100)                        //Revisa el índice para ver que el arreglo no este lleno.
    {
```

```
    dist2 = 0;                          //Inicializa la variable distancia en 0.
    time2 = millis();                   //Lee el tiempo del reloj interno y lo guarda en la variable time.
    y++;                                //Incrementa el valor de x cada que da una vuelta.
    tiempo2[i] = time2;                 //Se asigna time a cada una de las partes del arreglo.
    resta2 = tiempo2[i]-tiempo2[i-1]; //Resta da como resultdo el tiempo que paso entre una lectura y la otra.
    vel2 = (19.4779/(resta2*5))*1000; //Calcula la velocidad en cm/s.
    rev2 = (60000/(resta2*5));        //Calcula las revoluciones por minuto.
    dist2 = (19.4779*y)/4;            //Calcula la distancia recorrida sin importar el sentido de giro.

    Serial.print("Vel: ");
    Serial.print (vel2);
    Serial.print ("cm/s  ");
    Serial.print(rev2);
    Serial.print ("rev/min");
    Serial.print (dist2);
    Serial.println("cm.");

    j++;                             //Incrementa el valor de i para la siguiente lectura de tiempo.
   }
   else
   {
     j = 0;                         //cuando se llena el índice lo reinicia.
   }
}
```

Códigos empleados para las aplicaciones Android tanto en el dispositivo controlador como el que envía imagen.

Manifest de la aplicación que controla el vehículo y recibe la imagen del otro dispositivo:

```
<?xml version="1.0" encoding="utf-8"?>                                  //Validación de XML.
<manifest xmlns:android="http://schemas.android.com/apk/res/android"    //Define el nombre del espacio XML.
package="com.controlarecibeabre"                                        //Nombre del paquete donde se guarda
                                                                          información de la organización de las
                                                                          clases de java y del proyecto.
android:versionCode="1"                                                 //Versión de la aplicación.
android:versionName="1.0" >                                             //Nombre de la versión de la aplicación.
<uses-sdk android:minSdkVersion="7" />                                  //Versión minima de SDK que se require.
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />   //Permisos para utilizar el Bluetooth.
<uses-permission android:name="android.permission.BLUETOOTH" />         //Permisos para utilizar el Bluetooth.
<uses-permission android:name="android.permission.INTERNET" />          //Permisos para utilizar el Internet.
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/> //Permisos para escribir en dispositivos
                                                                          de almacenamiento externo.
<application
android:icon="@drawable/ic_launcher"                                    //Ocupa el logo ic_launcher almacenado en
                                                                          la carpeta drawable.
android:label="@string/app_name"                                        //Ocupa la etiqueta app_name almacenada
                                                                          en la carpeta string.
android:debuggable="true">                                              //Permite que la compilación del código
                                                                          se haga directamente en el teléfono.
<activity
android:label="@string/app_name"                                        //Ocupa la etiqueta app_name almacenada
                                                                          en la carpeta string.
android:name=".controlarecibeabre"                                      //Establece el nombre de la aplicación.
android:screenOrientation="portrait">                                   //Orientación.
<intent-filter >
<action android:name="android.intent.action.MAIN" />                    //Ocupa recursos del MAIN.
<category android:name="android.intent.category.LAUNCHER" />            //Ocupa recursos del LAUNCHER.
</intent-filter>
</activity>
</application>
</manifest>
```

Main layout de la aplicación que controla el vehículo y recibe la imagen del otro dispositivo:

```
<?xml version="1.0" encoding="utf-8"?>                          //Validación de XML.
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"     //Define el nombre del espacio XML.
android:layout_width="fill_parent"                              //Se ajusta al tamaño del contenedor.
android:layout_height="fill_parent"                             //Se ajusta al tamaño del contenedor.
android:orientation="vertical" >                                //Orientación vertical.

<TextView                                                       //PARA EL TEXTO.
android:id="@+id/txtcomentarios"                                //Identificador.
android:layout_width="wrap_content"                             //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                            //Se ajusta al tamaño del contenido.
android:text="Comentarios" />                                   //Texto que aparecerá.

<RelativeLayout                                                 //PARA LA VISTA RELATIVA.
android:id="@+id/relativeLayout1"                               //Identificador.
android:layout_width="fill_parent"                              //Se ajusta al tamaño del contenedor.
android:layout_height="119dp"                                   //Dimensiones del alto.
android:layout_weight="0.71" >                                  //Dimensiones del ancho.

<ImageView                                                      //PARA LA IMÁGEN.
android:id="@+id/imgabrirfoto"                                  //Identificador.
android:layout_width="fill_parent"                              //Se ajusta al tamaño del contenedor.
android:layout_height="fill_parent"                             //Se ajusta al tamaño del contenedor.
android:layout_centerHorizontal="true"                          //Centrar horizontalmente.
android:layout_centerVertical="true"                            //Centrar verticalmente.
android:src="@drawable/ic_launcher" />                          //Se guarda en la carpeta drawable

<TextView                                                       //PARA EL TEXTO.
android:id="@+id/txtcargabateria"                               //Identificador.
android:layout_width="wrap_content"                             //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                            //Se ajusta al tamaño del contenido.
android:layout_alignParentLeft="true"                           //Alineación izquierda.
android:layout_alignParentTop="true"                            //Alineación superior.
android:text="Carga de bateria" />                              //Texto que aparecerá.
```

```
<TextView                                                   //PARA EL TEXTO.
android:id="@+id/txtvalorcargabateria"                      //Identificador.
android:layout_width="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                        //Se ajusta al tamaño del contenido.
android:layout_alignParentTop="true"                        //Alineación superior.
android:layout_centerHorizontal="true"                      //Centrar horizontalmente.
android:text="% bateria" />                                 //Texto que aparecerá.

<TextView                                                   //PARA EL TEXTO.
android:id="@+id/txttemperaturaambiente"                    //Identificador.
android:layout_width="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                        //Se ajusta al tamaño del contenido.
android:layout_alignParentLeft="true"                       //Alineación izquierda.
android:layout_below="@+id/txtcargabateria"                 //Identificador.
android:text="Temperatura ambiente" />                      //Texto que aparecerá.

<TextView                                                   //PARA EL TEXTO.
android:id="@+id/txtvalortemperaturaambiente"               //Identificador.
android:layout_width="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                        //Se ajusta al tamaño del contenido.
android:layout_below="@+id/txtvalorcargabateria"             //Identificador.
android:layout_centerHorizontal="true"                      //Centrar horizontalmente.
android:text="Temperatura ambiental" />                     //Texto que aparecerá.

<TextView                                                   //PARA EL TEXTO.
android:id="@+id/txttemperaturapotencia"                    //Identificador.
android:layout_width="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                        //Se ajusta al tamaño del contenido.
android:layout_alignParentLeft="true"                       //Alineación izquierda.
android:layout_below="@+id/txttemperaturaambiente"          //Identificador.
android:text="Temperatura potencia" />                      //Texto que aparecerá.

<TextView                                                   //PARA EL TEXTO.
android:id="@+id/txtvalortemperaturapotencia"               //Identificador.
android:layout_width="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                        //Se ajusta al tamaño del contenido.
```

```
android:layout_alignBaseline="@+id/txttemperaturapotencia"          //Identificador.
android:layout_alignBottom="@+id/txttemperaturapotencia"            //Identificador.
android:layout_alignRight="@+id/txtvalortemperaturaambiente"        //Identificador.
android:text="Temperatura potencial" />                             //Texto que aparecerá.

<TextView                                                           //PARA EL TEXTO.
android:id="@+id/txtvelocidadcarro"                                 //Identificador.
android:layout_width="wrap_content"                                 //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                                //Se ajusta al tamaño del contenido.
android:layout_alignParentLeft="true"                               //Alineación izquierda.
android:layout_below="@+id/txttemperaturapotencia"                  //Identificador.
android:text="Velocidad promedio" />                                //Texto que aparecerá.

<TextView                                                           //PARA EL TEXTO.
android:id="@+id/txtvalorrevolucionescarro"                         //Identificador.
android:layout_width="wrap_content"                                 //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                                //Se ajusta al tamaño del contenido.
android:layout_below="@+id/txtvalordistanciacarro"                  //Identificador.
android:layout_centerHorizontal="true"                              //Centrar horizontalmente.
android:text="RPM" />                                               //Texto que aparecerá.

<TextView                                                           //PARA EL TEXTO.
android:id="@+id/txtvalordistanciacarro"                            //Identificador.
android:layout_width="wrap_content"                                 //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                                //Se ajusta al tamaño del contenido.
android:layout_below="@+id/txtvalorvelocidadpromedio"               //Identificador.
android:layout_centerHorizontal="true"                              //Centrar horizontalmente.
android:text="distancia " />                                        //Texto que aparecerá.

<TextView                                                           //PARA EL TEXTO.
android:id="@+id/txtvalorvelocidadpromedio"                         //Identificador.
android:layout_width="wrap_content"                                 //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                                //Se ajusta al tamaño del contenido.
android:layout_alignBaseline="@+id/txtvelocidadcarro"               //Identificador.
android:layout_alignBottom="@+id/txtvelocidadcarro"                 //Identificador.
android:layout_alignLeft="@+id/txtvalorcargabateria"                //Identificador.
android:text="velocidad" />                                         //Texto que aparecerá.
```

```
<TextView                                                    //PARA EL TEXTO.
android:id="@+id/txtdistanciacarro"                          //Identificador.
android:layout_width="wrap_content"                          //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_alignParentLeft="true"                        //Alineación izquierda.
android:layout_below="@+id/txtvelocidadcarro"                //Identificador.
android:text="Distancia recorrida" />                        //Texto que aparecerá.

<TextView                                                    //PARA EL TEXTO.
android:id="@+id/txtrevolucionescarro"                       //Identificador.
android:layout_width="wrap_content"                          //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_alignParentLeft="true"                        //Alineación izquierda.
android:layout_below="@+id/txtdistanciacarro"                //Identificador.
android:text="revoluciones" />                               //Texto que aparecerá.

</RelativeLayout>                                            //PARA LA VISTA RELATIVA.
<RelativeLayout                                              //PARA LA VISTA RELATIVA.
android:id="@+id/relativeLayout2"                            //Identificador.
android:layout_width="fill_parent"                           //Se ajusta al tamaño del contenedor.
android:layout_height="wrap_content" >                       //Se ajusta al tamaño del contenido.

<Button                                                      //PARA EL BOTÓN.
android:id="@+id/btnconectarwifi"                            //Identificador.
android:layout_width="wrap_content"                          //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_alignParentRight="true"                       //Alineación derecha.
android:layout_alignParentTop="true"                         //Alineación superior.
android:text="Conectar con la camara" />                     //Texto que aparecerá.

<Button                                                      //PARA EL BOTÓN.
android:id="@+id/btnconectarbt"                              //Identificador.
android:layout_width="wrap_content"                          //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_alignParentLeft="true"                        //Alineación izquierda.
android:layout_alignParentTop="true"                         //Alineación superior.
android:text="Conectar con el carro" />                      //Texto que aparecerá.
<Button                                                      //PARA EL BOTÓN.
```

```
android:id="@+id/btnestadodelcarro"                          //Identificador.
android:layout_width="wrap_content"                          //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_alignParentRight="true"                       //Alineación derecha.
android:layout_alignParentTop="true"                         //Alineación superior.
android:layout_marginRight="18dp"                            //Margen.
android:text="Ver estado del carro" />                       //Texto que aparecerá.

<Button                                                      //PARA EL BOTÓN.
android:id="@+id/btnregresardelestadodelcarro"               //Identificador.
android:layout_width="wrap_content"                          //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_alignParentLeft="true"                        //Alineación izquierda.
android:layout_alignParentTop="true"                         //Alineación superior.
android:layout_marginLeft="28dp"                             //Margen.
android:text="regresar" />                                   //Texto que aparecerá.

<ImageView                                                   //PARA LA IMÁGEN.
android:id="@+id/imgestadoconexion"                          //Identificador.
android:layout_width="wrap_content"                          //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                         //Se ajusta al tamaño del contenido.
android:layout_centerHorizontal="true"                       //Centrar horizontalmente.
android:layout_centerVertical="true"                         //Centrar verticalmente.
android:src="@drawable/rojo" />                              //Identificador.

</RelativeLayout>                                            //PARA LA VISTA RELATIVA.

</LinearLayout>                                              //Despliega la vista en una misma
                                                               Dirección (lineal).
```

Clase principal de la aplicación que controla el vehículo y recibe la imagen del otro dispositivo:

//Código para el controlador/////////////////////////////////////

```
package com.controlarecibeabre;                                          //Contiene información de la aplicación.

import java.io.BufferedReader;                                           //Librería Java.
import java.io.FileOutputStream;                                         //Librería Java.
import java.io.IOException;                                              //Librería Java.
import java.io.InputStream;                                              //Librería Java.
import java.io.InputStreamReader;                                        //Librería Java.
import java.io.OutputStream;                                             //Librería Java.
import java.net.InetAddress;                                             //Librería Java.
import java.net.Socket;                                                  //Librería Java.
import java.util.UUID;                                                   //Librería Java.
import android.app.Activity;                                             //Librería Java.
import android.bluetooth.BluetoothAdapter;                               //Librería Android.
import android.bluetooth.BluetoothDevice;                                //Librería Android.
import android.bluetooth.BluetoothSocket;                                //Librería Android.
import android.content.pm.ActivityInfo;                                  //Librería Android.
import android.graphics.Bitmap;                                          //Librería Android.
import android.graphics.BitmapFactory;                                   //Librería Android.
import android.graphics.Matrix;                                          //Librería Android.
import android.graphics.drawable.BitmapDrawable;                         //Librería Android.
import android.hardware.Sensor;                                          //Librería Android.
import android.hardware.SensorEvent;                                     //Librería Android.
import android.hardware.SensorEventListener;                             //Librería Android.
import android.hardware.SensorManager;                                   //Librería Android.
import android.os.Bundle;                                                //Librería Android.
import android.os.Handler;                                               //Librería Android.
import android.util.Log;                                                 //Librería Android.
import android.view.View;                                                //Librería Android.
import android.view.View.OnClickListener;                                //Librería Android.
import android.view.Window;                                              //Librería Android.
import android.view.WindowManager;                                       //Librería Android.
import android.widget.Button;                                            //Librería Android.
import android.widget.TextView;                                          //Librería Android.
```

```
import android.widget.Toast;                                    //Librería Android.
import android.widget.ImageView;                                //Librería Android.

public class controlarecibeabre extends Activity implements     //Clase controlarecibeabre que hereda
SensorEventListener,  OnClickListener {                           los atributos de la clase Activity
                                                                  para usar una interfaz y a su vez llama
                                                                  a los métodos SensorEventListener y
                                                                  OnClickListener.
private TextView txtcomentarios;                                //Declaración de variable private.
private TextView txtvalortemperaturapotencia;                   //Declaración de variable private.
private TextView txtvalorcargabateria;                          //Declaración de variable private.
private TextView txtvalortemperaturaambiente;                   //Declaración de variable private.
private TextView txttemperaturaambiente;                        //Declaración de variable private.
private TextView txtcargabateria;                               //Declaración de variable private.
private TextView txttemperaturapotencia;                        //Declaración de variable private.
private TextView txtvelocidadcarro;                             //Declaración de variable private.
private TextView txtdistanciacarro;                             //Declaración de variable private.
private TextView txtrevolucionescarro;                          //Declaración de variable private.
private TextView txtvalorvelocidadcarro;                        //Declaración de variable private.
private TextView txtvalordistanciacarro;                        //Declaración de variable private.
private TextView txtvalorrevolucionescarro;                     //Declaración de variable private.
private ImageView imgestadoconexion;                            //Declaración de variable private.
private ImageView imgabrirfoto;                                 //Declaración de variable private.
private Button btnconectarwifi;                                 //Declaración de variable private.
private Button btnconectarbt;                                   //Declaración de variable private.
private Button btnestadodelcarro;                               //Declaración de variable private.
private Button btnregresardelestadodelcarro;                    //Declaración de variable private.
private Handler conectarwifihandler = null;                     //Declaración de variable private.
private Runnable conectarwifirunner = null;                     //Declaración de variable private.
private Handler abririmagenhandler = null;                      //Declaración de variable private.
private Runnable abririmagenrunner = null;                      //Declaración de variable private.
private Handler sendbthandler = null;                           //Declaración de variable private.
private Runnable sendbtrunnable = null;                         //Declaración de variable private.
private Handler connectbthandler = null;                        //Declaración de variable private.
private Runnable connectbtrunnable = null;                      //Declaración de variable private.
private Handler receivebthandler = null;                        //Declaración de variable private.
private Runnable receivebtrunnable = null;                      //Declaración de variable private.
private String ruta1 = "/sdcard/recibido1.jpg";                  //Ruta de la imagen.
```

```
private String ruta2 = "/sdcard/recibido2.jpg";                                  //Ruta de la imagen.
public static String  lineinbt = null;                                          //Declaración de variable public.
public String [] valoresinbt = null;                                            //Declaración de variable public.
public BufferedReader bfreaderbt =null;                                         //Declaración de variable public.
public boolean recibirimg = false;                                              //Declaración de variable public.
public boolean abririmg = false;                                                //Declaración de variable public.
public boolean abririmg1 = true;                                                //Declaración de variable public.
public boolean abririmg2 =false;                                                //Declaración de variable public.
public boolean recibirimg1 = true;                                              //Declaración de variable public.
public boolean recibirimg2 =false;                                              //Declaración de variable public.
public boolean boolsendbt = true;                                               //Declaración de variable public.
public boolean boolconnectbt = false;                                           //Declaración de variable public.
public boolean boolreceivebt = false;                                           //Declaración de variable public.
public boolean boolparomotor = false;                                           //Declaración de variable public.
public boolean connectionbt=false;                                              //Declaración de variable public.
private static final int BYTE_MAX_VALUE = 127;                                  //Valor máximo de velocidad.
private static final int SEND_RATE_IN_MS = 100;                                 //Velocidad máxima de envío.
private static final int TIME_BETWEEN_CONNECTIONS_WIFI_IN_MS = 1300;            //Tiempo entre las conexiones WIFI.
private static final int TIME_TO_WAIT_BT = 5000;                                //Tiempo de espera para el BT.
private static final int RECEIVE_RATE_IN_MS = 1000;                             //Velocidad máxima de recepción.
private static final float RAWMAXRANGE = 10.0f;                                 //Trunca el valor del acelerómetro.
private static final float RAWMINRANGE = -10.0f;                                //Trunca el valor del acelerómetro.
private float fAccelX = 0.0f;                                                   //Variable para los acelerómetros.
private float fAccelY = 0.0f;                                                   //Variable para los acelerómetros.
private float fAccelZ = 0.0f;                                                   //Variable para los acelerómetros.
private byte byteY = 0;                                                         //Variable para el envío de información.
private byte byteZ = 0;                                                         //Variable para el envío de información.
private byte dirY = 0;                                                          //Variable para el envío de información.
private byte dirZ = 0;                                                          //Variable para el envío de información.
private String serverIpAddress = "192.168.43.1";                                //Dirección IP para conectar.
private int port = 8080;                                                        //Puerto a utilizar.
private SensorManager mSensorManager = null;                                    //Declaración de variable private.
private Sensor mAccelerometer = null;                                           //Declaración de variable private.
private BluetoothAdapter mBluetoothAdapter = null;                              //Declaración de variable private.
private BluetoothSocket btSocket = null;                                        //Declaración de variable private.
private OutputStream outstreambt = null;                                        //Declaración de variable private.
```

```
private static final UUID MY_UUID = UUID.fromString                          //Se asigna el identificador universal
("00001101-0000-1000-8000-00805F9B34FB");                                      único. Para la comunicación.
private static String MACbt = "00:06:66:04:DB:3B";                           //Dirección MAC del adaptador BT.

@Override                                                                    //Comienza la súper clase.
public void onCreate(Bundle savedInstanceState) {                            //Empieza con el último estado.
super.onCreate(savedInstanceState);                                          //Empieza con el último estado.
this.requestWindowFeature(Window.FEATURE_NO_TITLE);                          //Elimina el título de la aplicación.
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,             //Elimina la barra de estado del
WindowManager.LayoutParams.FLAG_FULLSCREEN);                                   teléfono móvil.
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);          //Orientación de la aplicación.
setContentView(R.layout.main);                                               //Asigna el archivo main como interfaz.

txtcomentarios  = (TextView) findViewById(R.id.txtcomentarios);                          //Asigna identificador.
txtvalorcargabateria = (TextView) findViewById(R.id.txtvalorcargabateria);               //Asigna identificador.
txtvalortemperaturaambiente = (TextView) findViewById(R.id.txtvalortemperaturaambiente); //Asigna identificador.
txttemperaturaambiente = (TextView) findViewById(R.id.txttemperaturaambiente);           //Asigna identificador.
txtcargabateria = (TextView) findViewById(R.id.txtcargabateria);                         //Asigna identificador.
txttemperaturapotencia  = (TextView) findViewById(R.id.txttemperaturapotencia);          //Asigna identificador.
txtvelocidadcarro = (TextView) findViewById(R.id.txtvelocidadcarro);                     //Asigna identificador.
txtdistanciacarro = (TextView) findViewById(R.id.txtdistanciacarro);                     //Asigna identificador.
txtrevolucionescarro = (TextView) findViewById(R.id.txtrevolucionescarro);               //Asigna identificador.
txtvalorvelocidadcarro = (TextView) findViewById(R.id.txtvalorvelocidadpromedio);        //Asigna identificador.
txtvalordistanciacarro = (TextView) findViewById(R.id.txtvalordistanciacarro);           //Asigna identificador.
txtvalorrevolucionescarro = (TextView) findViewById(R.id.txtvalorrevolucionescarro);     //Asigna identificador.
imgestadoconexion = (ImageView) findViewById(R.id.imgestadoconexion);                    //Asigna identificador.
imgabrirfoto = (ImageView) findViewById(R.id.imgabrirfoto);                              //Asigna identificador.
btnconectarwifi  = (Button) findViewById(R.id.btnconectarwifi);                          //Asigna identificador.
btnconectarbt = (Button) findViewById(R.id.btnconectarbt);                               //Asigna identificador.
btnestadodelcarro= (Button) findViewById(R.id.btnestadodelcarro);                        //Asigna identificador.
btnregresardelestadodelcarro = (Button) findViewById(R.id.btnregresardelestadodelcarro); //Asigna identificador.

btnconectarwifi.setOnClickListener(this);                                    //Switch y devolución para el botón.
btnconectarbt.setOnClickListener(this);                                      //Switch y devolución para el botón.
btnestadodelcarro.setOnClickListener(this);                                  //Switch y devolución para el botón.
btnregresardelestadodelcarro.setOnClickListener(this);                       //Switch y devolución para el botón.

pantallainicial();                                                           //Invoca el método.
```

```
Log.d("oncreate","se empieza con la actividad: threads");                    //Etiqueta para el logcat.
conectarwifihandler = new Handler();                                         //Nueva variable handler.
conectarwifirunner = new Runnable() {                                        //Nueva variable runnable.
public void run() {conectayrecibeimg();                                      //Ejecuta el método conectayrecibe con
conectarwifihandler.postDelayed(this,TIME_BETWEEN_CONNECTIONS_WIFI_IN_MS);     el tiempo establecido.
}
};
abririmagenhandler = new Handler();                                          //Nueva variable handler.
runOnUiThread(abririmagenrunner = new Runnable() {                           //Nueva variable runnable.
public void run() {abrir();                                                  //Ejecuta el método abrir y manda la
abririmagenhandler.post(this);                                                 información inmediatamente.
}});
Log.e("CONTROL", "*****ON CREATE*****");                                     //Etiqueta para el logcat.
mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);            //Información de sensores.
mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER); //Información acelerómetros.
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();                    //Adaptador por default.
if (mBluetoothAdapter == null) {                                             //Si el adaptador es null.
Toast.makeText(this,"No hay Bluetooth en este dispositivo",                  //Crea este mensaje.
Toast.LENGTH_LONG).show();                                                   //Muestra el mensaje.
finish();                                                                    //Acaba.
return;                                                                      //Regresa.
}
if (!mBluetoothAdapter.isEnabled()) {                                        //Si el adaptador está deshabilitado.
mBluetoothAdapter.enable();                                                  //Habilita al adaptador.
try {
Thread.sleep(TIME_TO_WAIT_BT);                                               //Espera al BT.
Toast.makeText(this,"Bluetooth activado por la aplicación",                  //Crea el mensaje.
Toast.LENGTH_LONG).show();                                                   //Muestra el mensaje.
} catch (InterruptedException e) {
e.printStackTrace();                                                         //Muestra etiqueta de error.
}} else {
Toast.makeText(this,"Bluetooth activado",                                    //Crea el mensaje.
Toast.LENGTH_LONG).show();                                                   //Muestra el mensaje.
}
sendbthandler = new Handler();                                               //Nueva variable handler.
sendbtrunnable = new Runnable() {                                            //Nueva variable runnable.
public void run() { sendData();                                              //Ejecuta el método sendData con los
sendbthandler.postDelayed(this, SEND_RATE_IN_MS);                              tiempos establecidos.
```

```
}
};
receivebthandler = new Handler();                                   //Nueva variable handler.
receivebtrunnable = new Runnable() {                                //Nueva variable runnable.
public void run() {receiveData();                                   //Ejecuta el método reciveData con los
receivebthandler.postDelayed(this, RECEIVE_RATE_IN_MS);               tiempos extablecido
}
};
connectbthandler = new Handler();                                   //Nueva variable handler.
connectbtrunnable = new Runnable() {                                //Nueva variable runnable.
public void run() {connect();                                       //Ejecuta el método connect y envía la
connectbthandler.post(this);                                          información inmediatamente.
}
};
}
@Override                                                           //Comienza la súper clase.
public void onResume() {                                            //Método onResume.
super.onResume();                                                   //Súper clase.
Log.e("CONTROL","*****ON RESUME*****");                             //Etiqueta para el logcat.
BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(MACbt);  //Dirección MAC a conectar.
try {
btSocket = device.createRfcommSocketToServiceRecord(MY_UUID);       //Se crea el socket del BT.
} catch (IOException e) {
Log.e("Socket BT", "ON RESUME: no se pudo crear el socket.", e);    //Etiqueta para el logcat
}
mBluetoothAdapter.cancelDiscovery();                                //Deja de buscar al adaptador.
connectionbt=false;                                                 //Valor de false a la variable.
mSensorManager.registerListener(this, mAccelerometer,               //Registra la lectura de los sensores
SensorManager.SENSOR_DELAY_GAME);                                     con una velocidad determinada.
new Thread (connectbtrunnable).start();                             //Declara y empieza el thread.
new Thread (sendbtrunnable).start();                                //Declara y empieza el thread.
new Thread (receivebtrunnable).start();                             //Declara y empieza el thread.
new Thread (conectarwifirunner).start();                            //Declara y empieza el thread.
new Thread (abririmagenrunner).start();                             //Declara y empieza el thread.
}
@Override                                                           //Comienza la súper clase.
public void onPause() {                                             //Método onPause.
super.onPause();                                                    //Súper clase.
```

```
Log.e("control", "*****ON PAUSE*****");                    //Etiqueta para el logcat
sendbthandler.removeCallbacks(sendbtrunnable);             //Deja de invocar.
mSensorManager.unregisterListener(this);                   //Deja de registrar.
if (outstreambt != null) {                                 //Si todavía hay datos.
try {
outstreambt.flush();                                       //Llama al método flush para eliminarlos.
} catch (IOException e) {
Log.e("BT", "ON PAUSE: no se pudo realizar el flush", e);  //Etiqueta para el logcat.
}
}
try {
btSocket.close();                                          //Cierra el socket.
connectionbt=false;                                        //Valor false a la variable.
} catch (IOException e2) {
Log.e("BT", "ON PAUSE: No se pudo cerrar el socket.", e2); //Etiqueta para el logcat.
connectionbt=false;                                        //Valor false a la variable.
}
}
@Override                                                  //Comienza la súper clase.
public void onStop() {                                     //Método onStop.
super.onStop();                                            //Súper clase.
Log.e("CONTROL", "*****ON STOP*****");                     //Etiqueta para el logcat.
}
@Override                                                  //Comienza la súper clase.
public void onDestroy() {                                  //Método onDestroy.
super.onDestroy();                                         //Súper clase.
Log.e("CONTROL", "*****ON DESTROY*****");                  //Etiqueta para el logcat.
}
public void onAccuracyChanged(Sensor sensor, int accuracy) {  //Método onAccuracyChanged.
}
public void onSensorChanged(SensorEvent event) {           //Método onSensorChanged.
synchronized (this) {                                      //Sincroniza variables para usarlas.


if (connectionbt)                                          //Si se realizó la conexión.
{
txtcomentarios.setText("Se ha establecido exitósamente la conexión          //Asigna el texto.
con el carrito. " + "Ahora Comienza la aplicación desde el otro
```

```
dispositivo Android " + "y aprieta el botón en ese dispositivo.
Después aprieta el botón de " + "conexión WiFi en esta aplicación");
imgestadoconexion.setImageResource(R.drawable.verde);                    //Asigna la imagen a mostrar.
}else if (!connectionbt){                                                //Si no se realizó la conexión.
imgestadoconexion.setImageResource(R.drawable.rojo);                     //Asigna la imagen a mostrar.
}
fAccelX = event.values[0];                                               //Toma datos del arreglo y asigna valor.
fAccelY = event.values[1];                                               //Toma datos del arreglo y asigna valor.
fAccelZ = event.values[2];                                               //Toma datos del arreglo y asigna valor.
if (fAccelX > RAWMAXRANGE) fAccelX = RAWMAXRANGE;                        //Acota valores de los acelerómetros.
if (fAccelX < RAWMINRANGE) fAccelX = RAWMINRANGE;                        //Acota valores de los acelerómetros.
if (fAccelY > RAWMAXRANGE) fAccelY = RAWMAXRANGE;                        //Acota valores de los acelerómetros.
if (fAccelY < RAWMINRANGE) fAccelY = RAWMINRANGE;                        //Acota valores de los acelerómetros.
if (fAccelZ > RAWMAXRANGE) fAccelZ = RAWMAXRANGE;                        //Acota valores de los acelerómetros.
if (fAccelZ < RAWMINRANGE) fAccelZ = RAWMINRANGE;                        //Acota valores de los acelerómetros.
if (fAccelY < 0.0f) {                                                    //Condición para fAccelY.
dirY = 'L';                                                              //dirY = L LEFT.
byteY = (byte) Math.round(((BYTE_MAX_VALUE * fAccelY))/RAWMINRANGE);     //Hace conversión para enviar datos.
}
else {
dirY = 'R';                                                              //dirY = R RIGHT.
byteY = (byte) Math.round(((BYTE_MAX_VALUE * fAccelY))/RAWMAXRANGE);     //Hace conversión para enviar datos.
}
if (fAccelZ < 0.0f) {                                                    //Condición para fAccelZ.
dirZ = 'B';                                                              //dirZ = B BACK.
byteZ = (byte) Math.round(((BYTE_MAX_VALUE * fAccelZ))/RAWMINRANGE);     //Hace conversión para enviar datos.
}
else {
dirZ = 'F';                                                              //dirZ = F FRONT.
byteZ = (byte) Math.round(((BYTE_MAX_VALUE * fAccelZ))/RAWMAXRANGE);     //Hace conversión para enviar datos.
}
}
}

public void connect(){                                                   //Método connect.
if (boolconnectbt){                                                      //Si boolconnectbt es true.
boolconnectbt = false;                                                   //Lo cambia a false.
btnconectarbt.setVisibility(View.GONE);                                  //Se deja de ver y libera recursos.
```

```
btnconectarwifi.setVisibility(View.VISIBLE);                                    //Se hace visible el botón.
try {
btSocket.connect();                                                             //Hace la conexión BT.
Log.d("BT", "Se ha establecido la conexión a la primera!");                     //Etiqueta para el logcat.
connectionbt=true;                                                              //Valor de true a la variable.
} catch (IOException e) {
try {
btSocket.connect();                                                             // Hace la conexión BT.
Log.d("BT", "Se estableció la conexión a la segunda!");                         //Etiqueta para el logcat.
connectionbt=true;                                                              //Valor de true a la variable.
} catch (IOException exception) {
txtcomentarios.setText("No se logró hacer la conexión");                        //Asigna texto.
exception.printStackTrace();                                                    // Muestra etiqueta de error.
try {
btSocket.close();
Log.e("BT", "Se ha cerrado el socket; no se logró establecer la conexión");  //Etiqueta para el logcat.
connectionbt=false;                                                             //Valor de false a la variable.
} catch (IOException e2) {
Log.e("BT", "No se cerró el Socket!!", e2);                                     //Etiqueta para el logcat.
}
}
}
try {
outstreambt = btSocket.getOutputStream();                                       //Instrucción para enviar información.
} catch (IOException e) {
Log.e("BT", "No se pudo crear el OutStream para el BT", e);                     //Etiqueta para el logcat.
}
try {
bfreaderbt = new BufferedReader(new InputStreamReader(                          //Lee la información para guardarla.
btSocket.getInputStream()),8*1024);                                             //Tamaño del buffer.
} catch (IOException e) {
e.printStackTrace();                                                            //Muestra etiqueta de error.
}
}
}
public void sendData () {                                                       //Método sendData.
if (connectionbt){                                                              //Si connectionbt es true.
synchronized (this) {                                                           //Sincroniza variables para usarlas.
```

```
byte[] msgBufferbt = new byte[5];                                   //Tamaño del arreglo.
if (boolparomotor){                                                 //Si boolparomotor es true.
msgBufferbt[0] = dirY;                                              //Asigna valor para el arreglo.
msgBufferbt[1] = 0;                                                 //Asigna valor para el arreglo.
msgBufferbt[2] = dirZ;                                              //Asigna valor para el arreglo.
msgBufferbt[3] = 0;                                                 //Asigna valor para el arreglo.
msgBufferbt[4] = ';';                                               //Asigna valor para el arreglo.
} else {
msgBufferbt[0] = dirY;                                              //Asigna valor para el arreglo.
msgBufferbt[1] = byteY;                                             //Asigna valor para el arreglo.
msgBufferbt[2] = dirZ;                                              //Asigna valor para el arreglo.
msgBufferbt[3] = byteZ;                                             //Asigna valor para el arreglo.
msgBufferbt[4] = ';';                                               //Asigna valor para el arreglo.
}
try {
outstreambt.write(msgBufferbt);                                     //Envía los datos del arreglo.
connectionbt=true;                                                  //Valor de true a la variable.
} catch (IOException e) {
connectionbt=false;                                                 //Valor de false a la variable.
Log.e("BT", "sendData(): Error al mandar datos", e);                //Etiqueta para el logcat.
}
}
}
}
public void receiveData(){                                          //Método receiveData.
if (boolreceivebt){                                                 //Si boolreceivebt es true.
try {
lineinbt = bfreaderbt.readLine();                                   //Lee la linea de datos del BT.
valoresinbt = lineinbt.split("/");                                  //Separa cada que lee un /
txtvalorcargabateria.setText("" + valoresinbt[0] + "%");            //Asigna valor al texto.
txtvalortemperaturaambiente.setText("" + valoresinbt[1] + "°c");    //Asigna valor al texto.
txtvalortemperaturapotencia.setText("" + valoresinbt[2] + "°c");    //Asigna valor al texto.
txtvalorvelocidadcarro.setText("" + valoresinbt[3] + "cm/s");       //Asigna valor al texto.
txtvalordistanciacarro.setText("" + valoresinbt[4] + "cm");         //Asigna valor al texto.
txtvalorrevolucionescarro.setText("" + valoresinbt[5] + "RPM");     //Asigna valor al texto.
} catch (IOException e) {
e.printStackTrace();                                                //Mensaje de error.
}
```

```
}
}
public void conectayrecibeimg () {                                  //Método conectayrecibeimg.
if (recibirimg){                                                    //Si recibirimg es true.
Log.i("conectayrecibe","comienzo");                                 //Etiqueta para el logcat.
txtcomentarios.setText("Ejecutando toda la aplicación");            //Asigna texto.
recibirimg = false;                                                 //Valor false para la variable.
try {
InetAddress serverAddr = InetAddress.getByName(serverIpAddress);    //Guarda la dirección IP en la variable.
Log.d("WiFi", "C: Conectando...");                                  //Etiqueta para el logcat.
Socket socketwifi = new Socket(serverAddr, port);                   //Guarda el Puerto en la variable.
if (recibirimg1){                                                   //Si recibirimg1 es true.
Log.i("WiFi","recibir1");                                           //Etiqueta para el logcat.
recibirimg1 = false;                                                //Cambia a false.
recibirimg2 = true;                                                 //Cambia a true.
try {
Log.d("WiFi"," recibiendo imagen");                                 //Etiqueta para el logcat.
InputStream inwifi = socketwifi.getInputStream();                   //Lee la información WIFI.
OutputStream outwifi = new FileOutputStream(ruta1);                 //Crea un archivo en la ruta establecida.
intoout(inwifi, outwifi);                                           //Pasa de input a outputstream.
socketwifi.close();                                                 //Cierra el socket WIFI.
outwifi.close();                                                    //Cierra el outwifi.
inwifi.close();                                                     //Cierra el inwifi.
abririmg = true;                                                    //Valor de true a la variable.
} catch (Exception e) {
e.printStackTrace();                                                //Mensaje de error.
}
}else if (recibirimg2){                                             //Si recibirimg2 es true.
Log.i("WiFi","recibir2");                                           //Etiqueta para el logcat.
recibirimg1 = true;                                                 //Cambia a true.
recibirimg2 = false;                                                //Cambia a false.
try {
Log.d("WiFi"," recibiendo imagen");                                 //Etiqueta para el logcat.
InputStream inwifi = socketwifi.getInputStream();                   //Lee los datos de la variable.
OutputStream outwifi = new FileOutputStream(ruta2);                 //Guarda la información en la ruta2.
intoout(inwifi, outwifi);                                           //Pasa de input a outputstream.
socketwifi.close();                                                 //Cierra el socket WIFI.
outwifi.close();                                                    //Cierra el outwifi.
```

```
inwifi.close();                                                          //Cierra el inwifi.
abriimg = true;                                                          //Valor de true a la variable.
} catch (Exception e) {
e.printStackTrace();                                                     //Mensaje de error.
}
}
socketwifi.close();                                                      //Cierra el socket WIFI
Log.d("WiFi", "Socket WiFi cerrado");                                    //Etiqueta para el logcat.
} catch (Exception e) {
Log.e("ClientActivity", "Error: WiFi", e);                               //Etiqueta para el logcat.
}
}
}
public void abrir(){                                                     //Método abrir.
if (abriimg){                                                            //Si abriimg es true.
abriimg=false;                                                           //Cambia a false.
Log.i("abrir imagen","comienzo");                                        //Etiqueta para el logcat.
if (abriimg1){                                                           //Si abriimg1 es true.
Log.i("abrir imagen","abrir1");                                          //Etiqueta para el logcat.
abriimg1 = false;                                                        //Cambia a false.
abriimg2 = true;                                                         //Cambia a true.
recibirimg = true;
Bitmap bMap = BitmapFactory.decodeFile(ruta1);                           //Carga el archivo de ruta1.
int w = bMap.getWidth();                                                 //Obtiene medida de ancho.
int h = bMap.getHeight();                                                //Obtiene medida de alto.
Matrix mtx = new Matrix();                                               //Crea una nueva matriz.
mtx.postRotate(90);                                                      //Rota la matriz 90 grados.
Bitmap rotatedBMP = Bitmap.createBitmap(bMap, 0, 0, w, h, mtx, true);    //Asigna valores a la matriz del gráfico.
BitmapDrawable bmd = new BitmapDrawable(rotatedBMP);                     //Muestra el archivo antes cargado.
imgabrirfoto.setImageDrawable(bmd);                                      //Abre la imagen.
recibirimg = true;                                                       //Valor de true a variable.
} else if (abriimg2){                                                    //Si abriimg2 es true.
Log.i("abrir imagen","abrir2");                                          //Etiqueta para el logcat.
abriimg1 = true;                                                         //Cambia a true.
abriimg2 = false;                                                        //Cambia a false.
recibirimg = true;                                                       //Valor de true a la variable.
Bitmap bMap = BitmapFactory.decodeFile(ruta2);                           //Carga el archivo de ruta2.
int w = bMap.getWidth();                                                 //Obtiene medida de ancho.
```

```
int h = bMap.getHeight();                                                    //Obtiene medida de alto.
Matrix mtx = new Matrix();                                                   //Crea una nueva matriz.
mtx.postRotate(90);                                                          //Rota la matriz 90 grados.
Bitmap rotatedBMP = Bitmap.createBitmap(bMap, 0, 0, w, h, mtx, true);        //Asigna valores a la matriz del gráfico.
BitmapDrawable bmd = new BitmapDrawable(rotatedBMP);                         //Muestra el archivo antes cargado.
imgabrirfoto.setImageDrawable(bmd);                                          //Abre la imagen.
}}
}
static void intoout(InputStream inwifi, OutputStream outwifi)                //Método intoout el cul convierte el
throws IOException {                                                           inputstream en outputstream.
byte[] buf = new byte[8192];                                                 //Crea un arreglo de 8192.
int len = 0;                                                                 //Variable inicializada en 0.
while ((len = inwifi.read(buf)) != -1) {                                     //Mientras no se corte la cadena.
outwifi.write(buf, 0, len);                                                  //Comienza a escribir desde len.
}
}
public void pantallainicial(){                                               //Método pantallainicial.
imgestadoconexion.setVisibility(View.VISIBLE);                               //Hace visible la imagen.
imgabrirfoto.setVisibility(View.GONE);                                       //Hace invisible la imagen.
txtcomentarios.setVisibility(View.VISIBLE);                                  //Hace visible el texto.
txtvalortemperaturapotencia.setVisibility(View.GONE);                        //Hace invisible el texto.
txtvalorcargabateria.setVisibility(View.GONE);                               //Hace invisible el texto.
txtvalortemperaturaambiente.setVisibility(View.GONE);                        //Hace invisible el texto.
btnconectarwifi.setVisibility(View.GONE);                                    //Hace invisible el botón.
btnconectarbt.setVisibility(View.VISIBLE);                                   //Hace visible el botón.
btnestadodelcarro.setVisibility(View.GONE);                                  //Hace invisible el botón.
btnregresardelestadodelcarro.setVisibility(View.GONE);                       //Hace invisible el botón.
txttemperaturaambiente.setVisibility(View.GONE);                             //Hace invisible el texto.
txtcargabateria.setVisibility(View.GONE);                                    //Hace invisible el texto.
txttemperaturapotencia.setVisibility(View.GONE);                             //Hace invisible el texto.
txtvelocidadcarro.setVisibility(View.GONE);                                  //Hace invisible el texto.
txtvalorvelocidadcarro.setVisibility(View.GONE);                             //Hace invisible el texto.
txtdistanciacarro.setVisibility(View.GONE);                                  //Hace invisible el texto.
txtvalordistanciacarro.setVisibility(View.GONE);                             //Hace invisible el texto.
txtrevolucionescarro.setVisibility(View.GONE);                               //Hace invisible el texto.
txtvalorrevolucionescarro.setVisibility(View.GONE);                          //Hace invisible el texto.
txtcomentarios.setText("Bienvenido a la aplicación.                          //Establece el texto.
Verificar que el Bluetooth " + "esté preparado para la conexión.
```

```
Cuando esté listo apriete el botón" + "de conexión Bluetooth");
}

public void pantallafotos(){                                        //Método pantallafotos.
imgestadoconexion.setVisibility(View.VISIBLE);                      //Hace visible la imagen.
imgabrirfoto.setVisibility(View.VISIBLE);                           //Hace visible la imagen.
txtcomentarios.setVisibility(View.GONE);                            //Hace invisible el texto.
txtvalortemperaturapotencia.setVisibility(View.GONE);               //Hace invisible el texto.
txtvalorcargabateria.setVisibility(View.GONE);                      //Hace invisible el texto.
txtvalortemperaturaambiente.setVisibility(View.GONE);               //Hace invisible el texto.
btnconectarwifi.setVisibility(View.GONE);                           //Hace invisible el botón.
btnconectarbt.setVisibility(View.GONE);                             //Hace invisible el botón.
btnestadodelcarro.setVisibility(View.VISIBLE);                      //Hace visible el botón.
btnregresardelestadodelcarro.setVisibility(View.GONE);              //Hace invisible el botón.
txttemperaturaambiente.setVisibility(View.GONE);                    //Hace invisible el texto.
txtcargabateria.setVisibility(View.GONE);                           //Hace invisible el texto.
txttemperaturapotencia.setVisibility(View.GONE);                    //Hace invisible el texto.
txtvelocidadcarro.setVisibility(View.GONE);                         //Hace invisible el texto.
txtvalorvelocidadcarro.setVisibility(View.GONE);                    //Hace invisible el texto.
txtdistanciacarro.setVisibility(View.GONE);                         //Hace invisible el texto.
txtvalordistanciacarro.setVisibility(View.GONE);                    //Hace invisible el texto.
txtrevolucionescarro.setVisibility(View.GONE);                      //Hace invisible el texto.
txtvalorrevolucionescarro.setVisibility(View.GONE);                 //Hace invisible el texto.
}
public void pantallaestado(){                                       //Método pantallaestado.
imgestadoconexion.setVisibility(View.VISIBLE);                      //Hace visible la imagen.
imgabrirfoto.setVisibility(View.GONE);                              //Hace invisible la imagen.
txtcomentarios.setVisibility(View.GONE);                            //Hace invisible el texto.
txtvalortemperaturapotencia.setVisibility(View.VISIBLE);            //Hace visible el texto.
txtvalorcargabateria.setVisibility(View.VISIBLE);                   //Hace visible el texto.
txtvalortemperaturaambiente.setVisibility(View.VISIBLE);            //Hace visible el texto.
btnconectarwifi.setVisibility(View.GONE);                           //Hace invisible el botón.
btnconectarbt.setVisibility(View.GONE);                             //Hace invisible el botón.
btnestadodelcarro.setVisibility(View.GONE);                         //Hace invisible el botón.
btnregresardelestadodelcarro.setVisibility(View.VISIBLE);           //Hace visible el botón.
txttemperaturaambiente.setVisibility(View.VISIBLE);                 //Hace visible el texto.
txtcargabateria.setVisibility(View.VISIBLE);                        //Hace visible el texto.
txttemperaturapotencia.setVisibility(View.VISIBLE);                 //Hace visible el texto.
```

```
txtvelocidadcarro.setVisibility(View.VISIBLE);                    //Hace visible el texto.
txtvalorvelocidadcarro.setVisibility(View.VISIBLE);               //Hace visible el texto.
txtdistanciacarro.setVisibility(View.VISIBLE);                    //Hace visible el texto.
txtvalordistanciacarro.setVisibility(View.VISIBLE);               //Hace visible el texto.
txtrevolucionescarro.setVisibility(View.VISIBLE);                 //Hace visible el texto.
txtvalorrevolucionescarro.setVisibility(View.VISIBLE);            //Hace visible el texto.
}

public void onClick(View v) {                                     //Método onClick.
if(v==btnestadodelcarro) {                                        //Si v se asocia con btnestadodelcarro.
boolreceivebt = true;                                             //Se hace true.
boolparomotor = true;                                             //Se hace true.
pantallaestado();                                                 //Llama al método pantallaestado.
} else if(v==btnregresardelestadodelcarro) {                      //v = btnregresardelestadodelcarro.
boolsendbt = true;                                                //Se hace true.
boolparomotor = false;                                            //Se hace false
pantallafotos();                                                  //Llama al método pantallafotos.
}else if(v==btnconectarbt) {                                      //Si v se asocial con btnconectarbt.
boolconnectbt = true;                                             //Se hace true.
} else if(v==btnconectarwifi) {                                   //Si v se asocia con btnconectarwifi.
pantallafotos();                                                  //Llama al método pantallafotos
recibirimg = true;                                                //Se hace true.
}}}
```

Manifest de la aplicación que envía la imagen:

```
<?xml version="1.0" encoding="utf-8"?>                                  //Validación de XML.
<manifest xmlns:android="http://schemas.android.com/apk/res/android"    //Define el nombre del espacio XML.
package="com.tomaenvia2"                                                //Nombre del paquete donde se guarda
                                                                          información de la organización de las
                                                                          clases de java y del proyecto.

android:versionCode="1"                                                 //Versión de la aplicación.
android:versionName="1.0" >                                             //Nombre de la versión de la aplicación.
<uses-sdk android:minSdkVersion="8" />                                  //Versión mínima de SDK que se requiere.
<uses-permission android:name="android.permission.INTERNET" />          //Permisos para utilizar el Internet.
<uses-permission android:name="android.permission.CAMERA"/>             //Permisos para utilizar la cámara.
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/> //Permisos para escribir en dispositivos
                                                                          de almacenamiento externo.
<application
android:icon="@drawable/ic_launcher"                                    //Ocupa el logo ic_launcher almacenado en
                                                                          la carpeta drawable.
android:label="@string/app_name"                                        //Ocupa la etiqueta app_name almacenada
                                                                          en la carpeta string.
android:debuggable="true">                                              //Permite que la compilación del código
                                                                          se haga directamente en el teléfono.
<activity
android:label="@string/app_name"                                        //Ocupa la etiqueta app_name almacenada
                                                                          en la carpeta string.
android:name=".tomaenvia2" >                                            //Establece el nombre de la aplicación.
<intent-filter >
<action android:name="android.intent.action.MAIN" />                    //Ocupa recursos del MAIN.
<category android:name="android.intent.category.LAUNCHER" />            //Ocupa recursos del LAUNCHER.
</intent-filter>
</activity>
</application>
</manifest>
```

Main layout de la aplicación que envía la imagen:

```
<?xml version="1.0" encoding="utf-8"?>                          //Validación de XML.
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"     //Define el nombre del espacio XML.
android:layout_width="match_parent"                             //Se ajusta al tamaño del contenedor.
android:layout_height="match_parent"                            //Se ajusta al tamaño del contenedor.
android:orientation="vertical" >                                //Define la orientación.

<TextView                                                       //PARA EL TEXTO.
android:id="@+id/txtcomentarios"                                //Identificador.
android:layout_width="wrap_content"                             //Se ajusta al tamaño del contenido.
android:layout_height="wrap_content"                            //Se ajusta al tamaño del contenido.
android:text="comentarios" />                                   //Texto que aparecerá.

<SurfaceView                                                    //PARA LOS GRÁFICOS.
android:id="@+id/camara"                                        //Identificador.
android:layout_width="match_parent"                             //Se ajusta al tamaño del contenedor.
android:layout_height="300dp" >                                 //Dimensiones del alto del gráfico.
</SurfaceView>

<Button                                                         //PARA EL BOTÓN.
android:id="@+id/btnterminarsecuencia"                          //Identificador.
android:layout_width="match_parent"                             //Se ajusta al tamaño del contenedor.
android:layout_height="50dp"                                    //Dimensiones del alto del botón.
android:text="Salir" />                                         //Texto que aparecerá.

<Button                                                         //PARA EL BOTÓN.
android:id="@+id/btniniciarsecuencia"                           //Identificador.
android:layout_width="match_parent"                             //Se ajusta al tamaño del contenedor.
android:layout_height="50dp"                                    //Dimensiones del alto del botón.
android:text="Conectar y enviar" >                              //Texto que aparecerá.
</Button>

</LinearLayout>                                                 //Despliega la vista en una misma
                                                                  Dirección (lineal).
```

Clase principal de la aplicación que envía la imagen:

//Código para el carro/////////////////////////////////////

```
package com.tomaenvia2;                                                    //Contiene información de la aplicación.

import android.app.Activity;                                               //Librería Android.
import android.os.Bundle;                                                  //Librería Android.
import java.io.FileNotFoundException;                                      //Librería Java.
import java.io.FileOutputStream;                                           //Librería Java.
import java.io.IOException;                                                //Librería Java.
import android.content.pm.ActivityInfo;                                    //Librería Android.
import android.hardware.Camera;                                            //Librería Android.
import android.hardware.Camera.PictureCallback;                            //Librería Android.
import android.hardware.Camera.ShutterCallback;                            //Librería Android.
import android.os.Handler;                                                 //Librería Android.
import android.util.Log;                                                   //Librería Android.
import android.view.SurfaceHolder;                                         //Librería Android.
import android.view.SurfaceView;                                           //Librería Android.
import android.view.View;                                                  //Librería Android.
import android.view.View.OnClickListener;                                  //Librería Android.
import android.view.Window;                                                //Librería Android.
import android.view.WindowManager;                                         //Librería Android.
import android.widget.Button;                                              //Librería Android.
import android.widget.TextView;                                            //Librería Android.
import java.io.FileInputStream;                                            //Librería Java.
import java.io.InputStream;                                                //Librería Java.
import java.io.OutputStream;                                               //Librería Java.
import java.net.ServerSocket;                                              //Librería Java.
import java.net.Socket;                                                    //Librería Java.


public class tomaenvia2 extends Activity implements                       //Clase tomaenvia2 que hereda los
OnClickListener, SurfaceHolder.Callback {                                    atributos de la clase Activity
                                                                             para usar una interfaz y a su vez
                                                                             llama a los métodos OnClickListener y
                                                                             SurfaceHolder.Callback.


private Camera camera;                                                     //Declaración de variable private.
```

```
private TextView txtcomentarios;                                    //Declaración de variable private.
private Button btniniciarsecuencia;                                 //Declaración de variable private.
private Button btnpararsecuencia;                                   //Declaración de variable private.
private SurfaceView surface;                                        //Declaración de variable private.
private SurfaceHolder holder;                                       //Declaración de variable private.
public FileOutputStream outStreamCamara = null;                     //Declaración de variable public.
private Handler fotoHandler = null;                                 //Declaración de variable private.
private Runnable fotoRunner = null;                                 //Declaración de variable private.
private int tiempo_entre_fotos = 900;                               //Declaración de variable private.
private String ruta1 = "/mnt/sdcard/fotocel/test1.jpg";             //Ruta para guardar la imagen.
private String ruta2 = "/mnt/sdcard/fotocel/test2.jpg";             //Ruta para guardar la imagen.
public boolean tomarfoto = false;                                   //Declaración de variable public.
public boolean conectayenvia = false;                               //Declaración de variable public.
public boolean foto1 = true;                                        //Declaración de variable public.
public boolean envia1 = true;                                       //Declaración de variable public.
public boolean foto2 = false;                                       //Declaración de variable public.
public boolean envia2 = false;                                      //Declaración de variable public.
public static final int SERVERPORT = 8080;                          //Declaración del puerto a utilizar.
private ServerSocket serverSocket;                                  //Declaración de variable private.
private Handler conectarhandler = null;                             //Declaración de variable private.
private Runnable conectarunner = null;                              //Declaración de variable private.

@Override                                                           //Comienza la súper clase.
public void onCreate(Bundle savedInstanceState) {                   //Empieza con el último estado.
super.onCreate(savedInstanceState);                                 //Empieza con el último estado.
this.requestWindowFeature(Window.FEATURE_NO_TITLE);                 //Elimina el título de la aplicación.
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,    //Elimina la barra de estado del
WindowManager.LayoutParams.FLAG_FULLSCREEN);                          teléfono móvil.
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);  //Orientación de la aplicación.
setContentView(R.layout.main);                                      //Asigna el archivo main como interfaz.

surface = (SurfaceView)findViewById(R.id.camara);                   //Asigna identificador.
txtcomentarios = (TextView) findViewById(R.id.txtcomentarios);      //Asigna identificador.
txtcomentarios.setText("Apretar el botón si el otro equipo está     //Asigna texto.
conectado a este");
holder = surface.getHolder();                                       //Variable para manipular el surface.
holder.addCallback(this);                                           //Interfaz para el holder.
holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);            //Define el tipo de superficie a usar.
```

```
btniniciarsecuencia = (Button) findViewById(R.id.btniniciarsecuencia);       //Asigna identificador.
btnpararsecuencia = (Button) findViewById(R.id.btnterminarsecuencia);        //Asigna identificador.
btniniciarsecuencia.setOnClickListener(this);                                //Switch y devolución para el botón.
btnpararsecuencia.setOnClickListener(this);                                  //Switch y devolución para el botón.
btniniciarsecuencia.setVisibility(View.VISIBLE);                             //Hace visible el botón.
btnpararsecuencia.setVisibility(View.GONE);                                  //Hace invisible el botón.
Log.d("ON CREATE","*****ON CREATE*****");                                    //Etiqueta para el logcat.
fotoHandler = new Handler();                                                 //Nueva variable handler.
fotoRunner = new Runnable() {                                                //Nueva variable runnable.
public void run() {tomarfoto();                                              //Ejecuta el método tomarfoto
fotoHandler.postDelayed(this, tiempo_entre_fotos);                             en los tiempo establecidos.
}
};
conectarhandler = new Handler();                                             //Nueva variable handler.
conectarunner = new Runnable() {                                             //Nueva variable runnable.
public void run() {conectayenvia();                                          //Ejecuta el método conectayenvia y
conectarhandler.post(this);                                                    envía la información inmediatamente.
}
};
Log.d("ON CREATE","Se han creado los threads");                              //Etiqueta para el logcat.
}
@Override                                                                    //Comienza la súper clase.
protected void onResume() {                                                  //Método onResume.
super.onResume();                                                            //Súper clase.
new Thread (fotoRunner).start();                                             //Declara y empieza el thread.
new Thread (conectarunner).start();                                          //Declara y empieza el thread.
}
@Override                                                                    //Comienza la súper clase.
protected void onStop() {                                                    //Método onStop.
super.onStop();                                                              //Súper clase.
Log.d("ON STOP","****ON STOP*****");                                         //Etiqueta para el logcat.
try {
serverSocket.close();                                                        //Cierra el socket.
} catch (IOException e) {
e.printStackTrace();                                                         //Etiqueta de error.
}
}
```

```
public void surfaceCreated(SurfaceHolder holder) {                          //Método surfaceCreated.
Log.d("surfacecreated","comienzo");                                          //Etiqueta para el logcat.
try {
camera = Camera.open();                                                      //Abre la cámara.
camera.setPreviewDisplay(holder);                                            //Inicia la pantalla de preview.
camera.setDisplayOrientation(90);                                            //Indica la orientación de la pantalla.
camera.startPreview();                                                       //Inicia el preview.
Camera.Parameters cameraParameters = camera.getParameters();                 //Obtiene los parámetros de la cámara.
cameraParameters.setPreviewSize(176, 144);                                   //Tamaño del preview.
cameraParameters.setPictureSize(176,144);                                    //Tamaño de las fotos.
cameraParameters.setRotation(90);                                            //Rota 90 grados.
camera.setParameters(cameraParameters);                                      //Aplica los tamaños antes establecidos.
} catch (IOException e) {
Log.d("CAMERA", e.getMessage());                                             //Etiqueta para el logcat.
}
}
public void surfaceDestroyed(SurfaceHolder holder) {                         //Método surfaceDestroyed.
Log.d("surfacedestroyed","comienzo");                                        //Etiqueta para el logcat.
camera.stopPreview();                                                        //Detiene el preview.
camera.release();                                                            //Libera la cámara.
}
public void surfaceChanged(SurfaceHolder holder, int format, int width,      //Método surfaceChanged. Se utilizan las
int height) {                                                                  variables y valores antes establecidos.
Log.d("surfacechanged","comienzo");                                          //Etiqueta para el logcat.
}
public void tomarfoto () {                                                   //Método tomarfoto.
if (tomarfoto){                                                              //Si tomar foto es true.
tomarfoto=false;                                                             //Lo cambia a false
Log.d("tomarfoto","comienzo");                                               //Etiqueta para el logcat.
camera.takePicture(shutterCallback, rawCallback, jpegCallback);              //Toma la fotografía.
}
}


ShutterCallback shutterCallback = new ShutterCallback() {                    //Nueva variable shutterCallback.
public void onShutter() {                                                    //Método onShutter.
Log.d("onshutter","comienzo");                                               //Etiqueta para el logcat.
}
```

```
};

PictureCallback rawCallback = new PictureCallback() {                    //Nueva variable rawCallback
public void onPictureTaken(byte[] data, Camera camera) {                 //Método onPictureTaken.
Log.d("rawcallback","comienzo");                                         //Etiqueta para el logcat.
}
};

PictureCallback jpegCallback = new PictureCallback() {                   //Nueva variable jpegCallback.
public void onPictureTaken(byte[] data, Camera camera) {                 //Método onPictureTaken.
Log.d("jpegcallback","comienzo");                                        //Etiqueta para el logcat.
if (foto1){                                                              //Si foto1 es true.
Log.i("jpegcallback","foto1");                                           //Etiqueta para el logcat.
foto1 = false;                                                           //Lo cambia a false.
foto2 = true;                                                            //Lo cambia a true.
try {
outStreamCamara = new FileOutputStream(ruta1);                           //Envía la foto a la ruta1
outStreamCamara.write(data);                                             //Escribe el archivo.
conectayenvia = true;                                                    //Valor true a la variable.
outStreamCamara.close();                                                 //Cierra el envío de información.
} catch (FileNotFoundException e) {
Log.d("CAMERA", e.getMessage());                                         //Etiqueta para el logcat.
} catch (IOException e) {
Log.d("CAMERA", e.getMessage());                                         //Etiqueta para el logcat.
}
} else if (foto2){                                                       //Si foto2 es true.
Log.i("picturecallback","foto2");                                        //Etiqueta para el logcat.
foto1 = true;                                                            //Lo cambia a true.
foto2 = false;                                                           //Lo cambia a false
try {
outStreamCamara = new FileOutputStream(ruta2);                           //Envía la foto a la ruta2.
outStreamCamara.write(data);                                             //Escribe el archivo.
conectayenvia = true;                                                    //Valor de true a la variable.
outStreamCamara.close();                                                 //Cierra el envío de información.
} catch (FileNotFoundException e) {
Log.d("CAMERA", e.getMessage());                                         //Etiqueta para el logcat.
} catch (IOException e) {
Log.d("CAMERA", e.getMessage());                                         //Etiqueta para el logcat.
```

```
}
}
camera.stopPreview();                                   //Detiene el preview.
camera.startPreview();                                  //Lo vuelve a iniciar.
}
};

public void conectayenvia () {                          //Método conectayenvia.
if (conectayenvia){                                     //Si conectayenvia es true.
txtcomentarios.setText("Secuencia iniciada. Apretar el botón   //Se establece el texto a mostrar en
para detener y salir");                                   la pantalla de la aplicación.
Log.d("conectayenvia","comienzo");                      //Etiqueta para el logcat.
try {
serverSocket = new ServerSocket(SERVERPORT);            //Se establece el Puerto a utilizar.
Log.d("conectayenvia","antesdeacept");                  //Etiqueta para el logcat.
conectayenvia = false;                                  //Lo cambia a false.
Socket client = serverSocket.accept();                  //Se acepta la conexión.
Log.d("conectayenvia","despuesacept");                  //Etiqueta para el logcat.
Log.d("on create"," se dio el accept");                 //Etiqueta para el logcat.
if (envia1){                                            //Si envia1 es true.
Log.i("conectayenvia","envia1");                        //Etiqueta para el logcat.
envia1 = false;                                         //Lo cambia a false.
envia2 = true;                                          //Lo cambia a true.
try {
InputStream in = new FileInputStream(ruta1);            //Nuevo archivo en la ruta1
OutputStream out = client.getOutputStream();            //Obtiene información de OutputStream.
parametrosenviaryrecibir(in, out);                      //Se asignan los parámetros.
tomarfoto = true;                                       //Valor de true a la variable.
client.close();                                         //Se cierra la conexión con el client.
serverSocket.close();                                   //Cierra el socket.
out.close();                                            //Se cierra la salida.
in.close();                                             //Se cierra la entrada.
} catch (Exception e) {
Log.e("ClientActivity", "S: Error", e);                 //Etiqueta para el logcat.
}
} else if (envia2){                                     //Si envia2 es true.
Log.i("conectayenvia","envia2");                        //Etiqueta para el logcat.
envia1 = true;                                          //Cambia a true.
```

```
envia2 = false;                                                    //Cambia a false.
try {
InputStream in = new FileInputStream(ruta2);                       //Nuevo archivo en la ruta2
OutputStream out = client.getOutputStream();                       //Obtiene información de OutputStream.
parametrosenviaryrecibir(in, out);                                 //Asigna los parámetros.
tomarfoto = true;                                                  //Valor de true a la variable.
client.close();                                                    //Cierra conexión con client.
serverSocket.close();                                              //Cierra el socket.
out.close();                                                       //Cierra salida.
in.close();                                                        //Cierra entrada.
} catch (Exception e) {
Log.e("ClientActivity", "S: Error", e);                            //Etiqueta para el logcat.
}
}
} catch (Exception e) {
Log.d("SOCKET","valio madres");                                    //Etiqueta para el logcat.
e.printStackTrace();                                               //Etiqueta de error.
}
}
}
@Override                                                          //Comienza la súper clase.
protected void onDestroy() {                                       //Método onDestroy.
super.onDestroy();                                                 //Súper clase.
Log.d("ON DESTROY","*****ON DESTROY*****");                        //Etiqueta para el logcat.
}
@Override                                                          //Comienza la súper clase.
protected void onPause() {                                         //Método onPause.
Log.d("ON PAUSE","****ON PAUSE*****");                             //Etiqueta para el logcat.
super.onPause();                                                   //Súper clase.
}

static void parametrosenviaryrecibir(InputStream in, OutputStream out)       //Método parametrosenviaryrecibir.
throws IOException {
byte[] buf = new byte[8192];                                       //Arreglo para la imagen.
int len = 0;                                                       //Variable para iniciar en 0.
while ((len = in.read(buf)) != -1) {                               //Mientras no se corte la cadena.
out.write(buf, 0, len);                                            //Empieza a escribir desde len
}
```

```
}
public void onClick(View v) {                                          //Método onClick.
if (v==btniniciarsecuencia){                                           //Si v se asocia con btniniciarsecuencia.
tomarfoto = true;                                                      //Valor de true a la variable.
btniniciarsecuencia.setVisibility(View.GONE);                          //Hace invisible el botón.
btnpararsecuencia.setVisibility(View.VISIBLE);                         //Hace visible el botón.
txtcomentarios.setText("Ahora oprime el botón en el otro dispositivo");  //Asigna texto.
} else if (v== btnpararsecuencia){                                     //Si v se asocia con btnpararsecuencia.
tomarfoto = false;                                                     //Valor de false a la variable.
conectayenvia = false;                                                 //Valor de false a la variable.
conectarhandler.removeCallbacks(conectarunner);                        //Remueve valores del handler.
fotoHandler.removeCallbacks(fotoRunner);                               //Remueve valores del handler.
Log.d("Pausa","libero");                                               //Etiqueta para el logcat.
try {
outStreamCamara.close();                                               //Cierra el envío de la cámara.
} catch (IOException e) {
e.printStackTrace();                                                   //Etiqueta de error.
}
try {
serverSocket.close();                                                  //Cierra el socket.
} catch (IOException e) {
e.printStackTrace();                                                   //Etiqueta de error.
}
finish();                                                              //Termina el método.
return;                                                                //Regresa.
}}}
```

REFERENCIAS

PÁGINAS WEB

Android. (s.f.). Recuperado el 11 de Julio de 2011, de Wikipedia: http://es.wikipedia.org/wiki/Android

González, A. N. (8 de Febrero de 2008). *¿Qué es Android?* Recuperado el 11 de Julio de 2011, de Xataka Android: http://www.xatakandroid.com/sistema-operativo/que-es-android

Telefonía Móvil. (s.f.). Recuperado el 11 de Julio de 2011, de Wikipedia: http://es.wikipedia.org/wiki/Telefon%C3%ADa_m%C3%B3vil#Tel.C3.A9fono_m.C3.B3vil_o_celular

Herrera, G. (4 de Julio de 2011). *La Batalla entre Android y Apple.* Recuperado el 13 de Julio de 2011, de Info7 MX: http://www.info7.com.mx/editorial.php?id=1782

Radiofrecuencia. (s.f.). Recuperado el 23 de Julio de 2011, de Wikipedia: http://es.wikipedia.org/wiki/Radiofrecuencia

Buetooth. (s.f.). Recuperado el 30 de Octubre de 2011, de Wikipedia: http://es.wikipedia.org/wiki/Bluetooth#Versiones

Punto de acceso inalámbrico. (s.f.). Recuperado el 30 de Octubre de 2011, de Wikipedia: http://es.wikipedia.org/wiki/Punto_de_acceso_inal%C3%A1mbrico

Wi-Fi. (s.f.). Recuperado el 30 de Octubre de 2011, de Wikipedia: http://es.wikipedia.org/wiki/Wi-Fi

Marshall Brain, T. V. (s.f.). *How WiFi Works.* Recuperado el 30 de Octubre de 2011, de How stuff works: http://computer.howstuffworks.com/wireless-network1.htm

Aquino, N. E. (s.f.). *Redes y Comunicación Inalámbrica*. Recuperado el 31 de Octubre de 2011, de Ilustrados: http://www.ilustrados.com/tema/8666/Redes-Comunicacion-Inalambrica.html

JOSEMI. (18 de Agosto de 2006). *¿Conexiones con cable o inalámbricas? - Ventajas e inconvenientes*. Recuperado el 31 de Octubre de 2011, de Configurar Equipos: http://www.configurarequipos.com/doc351.html

Carrier Sense Multiple Access with Collision Avoidance. (s.f.). Recuperado el 31 de Octubre de 2011, de Wikipedia: http://es.wikipedia.org/wiki/Carrier_sense_multiple_access_with_collision_avoidance

IEEE 802.11. (s.f.). Recuperado el 31 de Octubre de 2011, de Wikipedia: http://es.wikipedia.org/wiki/IEEE_802.11#802.11b

Bluetooth vs. WiFi. (s.f.). Recuperado el 2 de Noviembre de 2011, de Diffen: http://www.diffen.com/difference/Bluetooth_vs_Wi-Fi

The difference between Bluetooth and WiFi. (5 de Enero de 2001). Recuperado el 2 de Noviembre de 2011, de Toshiba: http://nl.computers.toshiba-europe.com/innovation/download_whitepaper.jsp?z=25&WHITEPAPER_ID=TISBBLUEWIFI

WiFi vs. Bluetooth. (s.f.). Recuperado el 2 de Noviembre de 2011, de Universidad de Castilla-La Mancha: http://www.info-ab.uclm.es/labelec/solar/Comunicacion/Wifi/Wifi_vs_Bluetooh.htm

Bluetooth Controlled Car BBZ-201-A0. (s.f.). Recuperado el 3 de Noviembre de 2011, de BeeWi Simply Wireless: http://www.bee-wi.com/bluetooth-controlled-car-bbz201-beewi,us,4,BBZ201-A0.cfm

A Technological First. (s.f.). Recuperado el 7 de Noviembre de 2011, de Parrot Ar.Drone: http://ardrone.parrot.com/parrot-ar-drone/usa/technologies

About Parrot. (s.f.). Recuperado el 7 de Noviembre de 2011, de Parrot: http://www.parrot.com/usa/aboutparrot

Sony Ericsson Xperia X10 mini- A Fondo. (s.f.). Recuperado el 8 de Noviembre de 2011, de tuexperto.com: http://www.tuexperto.com/2010/02/14/sony-ericsson-xperia-x10-mini-%E2%80%93-a-fondo/

Samsung i5500 Galaxy 5 GT-I5500L (Android 2.1). (s.f.). Recuperado el 8 de Noviembre de 2011, de Wayerless: http://www.mobilecloseup.com/foro/showthread.php?t=558681

Descripción de la Open Handset Alliance (s.f.). Recuperado el 9 de noviembre del 2011 de Open Handset Alliance: http://www.openhandsetalliance.com/

Android (s.f.). Recuperado el 9 de noviembre del 2011 de Open Handset Alliance: http://www.openhandsetalliance.com/android_overview.html

Android (s.f.). Recuperado el 9 de noviembre del 2011 de Android developers: http://developer.android.com/index.html

Herramientas para desarrollar programas de Android (s.f.). Recuperado el 9 de noviembre del 2011 de Android developers: http://developer.android.com/guide/developing/tools/index.html

Fundamentos de las aplicaciones de Android (s.f.). Recuperado el 9 de noviembre del 2011 de Android developers: http://developer.android.com/guide/topics/fundamentals.html

Consulta de ejemplos de código para Android (s.f.). Recuperado el 10 de noviembre del 2011 de Google code: http://code.google.com/intl/es-MX/android/

Requisitos para instalar el SDK (s.f.). Recuperado el 10 de noviembre del 2011 de Android developers: http://developer.android.com/sdk/requirements.html

Comparación entre el WP7 Marketplace, la Apple App Store y el Android Market (17 de octubre del 2011). Recuperado el 10 de noviembre del 2011 de Open Handset Alliance: http://www.research2guidance.com/android-market-reaches-half-a-million-successful-submissions/

Introducción a la creación de aplicaciones de Android (s.f.). Recuperado el 10 de noviembre del 2011 de Android developers: http://developer.android.com/guide/developing/index.html

Comparación entre las distintas plataformas disponibles para dispositivos de telefonía celular (s.f.). Recuperado el 10 de noviembre del 2011 de Vision Mobile: http://www.visionmobile.com/blog/2011/06/developer-economics-2011-winners-and-losers-in-the-platform-race/

Revisión de la versión 4.0 de Android (s.f.). Recuperado el 12 de noviembre del 2011 de Android developers: http://developer.android.com/sdk/android-4.0-highlights.html

Asistente personal activado por voz Iris para Android (2 de noviembre del 2011). Recuperado el 12 de noviembre del 2011 de GMA news: http://www.gmanews.tv/story/237267/meet-iris-siris-android-counterpart

Detalles de la aplicación Iris (s.f.). Recuperado el 12 de noviembre del 2011 del Android Market: https://market.android.com/details?id=com.dexetra.iris

IMÁGENES

Fig 1-1 http://www.santiagokoval.com/2010/03/13/telefonos-inalambricos-y-tumores-cerebrales-malignos/

Fig 1-2 Ableson, Frank, “Unlocking Android: a developer’s guide”, Ed. Manning, 2008, p. 5.

Fig 1-3 http://www.openhandsetalliance.com/

Fig 1-4 http://www.visionmobile.com/blog/2011/06/developer-economics-2011-winners-and-losers-in-the-platform-race/

Fig 1-5 http://www.elandroidelibre.com/2011/10/android-4-0-ice-cream-sandwich-presentado-oficialmente-analisis-completo.html

Fig 1-6 Mayné, J. (2009). Estado Actual de las Comunicaciones por Radiofrecuencia. *Silica* .

Fig 1-7 Mayné, J. (2009). Estado Actual de las Comunicaciones por Radiofrecuencia. *Silica* .

Fig 1-8 http://www.bee-wi.com/bluetooth-controlled-car-bbz201-beewi,us,4,BBZ201-A0.cfm

Fig 1-9 http://www.bee-wi.com/bluetooth-controlled-car-bbz201-beewi,us,4,BBZ201-A0.cfm

Fig 1-10 https://market.android.com/details?id=com.baracoda.android.bluetoothcar.remotecontrol&feature=search_result

Fig 1-11 http://www.blue-drone.com/

Fig 1-12 http://tecnyo.com/parrot-ar-drone-el-nuevo-juguete-para-adultos/

Fig 3-1 Datasheet L293d

Fig 3-2 Datasheet KA78R05

Fig 3-3 Datasheet LM35DZ

Fig 3-4 Datasheet IR383_HT y PT1302BC2

Fig 3-5 Datasheet LM324

Fig 3-6 http://upload.wikimedia.org/wikipedia/commons/c/c3/Opampfollowing.png

Fig 3-7 http://www.robotshop.com/spark-fun-bluesmirf-1.html

Fig 3-8 http://arduino.cc/en/uploads/Hacking/Atmega168PinMap2.png

Fig 4-3 http://developer.android.com/reference/android/app/Activity.html

Fig 4-4 ABLESON Frank, "Unlocking Android: a developer's guide", Ed. Manning, 2008, p. 44

Fig 5-1 http://valdezate.blogspot.com/2011/12/configurando-una-ip-cam.html

http://www.zero13wireless.net/foro/showthread.php?5746-Analisis-WebCam-IP-Wireless-TopCom-Pro-2000

http://www.fastgadgets.net/?p=67

http://www.cameras-cctv.com/vivotek-ip7361-ip-camera

Fig 5-2 http://www.esato.com/board/viewtopic.php?topic=196955

Fig 5-3 http://www.radiodx.qsl.br/galaxy_5.htm

Fig 6-1 http://electrozone.com.mx/tblarticulos_list.php?goto=371

http://www.olimex.cl/present.php?page=tutorial-encapsulados

http://ayudaelectronica.com/codigo-de-resistencias-smd/

http://ayudaelectronica.com/capacitores-smd/

Fig 6-2 http://www.masoportunidades.com.ar/aviso/6241452-cristal-20-mhz-20000-khz-smd-montaje-superficial-30ppm-disponible-en-capital-federal

http://listado.mercadolibre.com.ar/Kit-IC-SMD-TL072-LM324

http://www.zonemicro.ca/zoneenglish/pivot/archive.php?c=Other_Components&w=&t=

http://desandroid.com/2011/03/28/bienvenido-a-desandroid/android-logo/

http://es.wikipedia.org/wiki/Archivo:Diagrama_android.png

http://www.famouslogos.us/bluetooth-logo/

http://es.wikipedia.org/wiki/Archivo:Logo_WiFi.svg

http://www.arduino.cc/playground/Learning/Tutorial01

Fig 6-3 CORNISH María Laura, "El ABC de los plásticos", Universidad Iberoamericana, México, D.F., 1997, p. 98

Fig 6-4 CORNISH María Laura, "El ABC de los plásticos", Universidad Iberoamericana, México, D.F., 1997, p. 98

ARTÍCULOS

A Developers's First Look At Android. (2008). *Linux For You* , 48-52.

Mayné, J. (2009). Estado Actual de las Comunicaciones por Radiofrecuencia. *Silica* .

Huaizhi Li, M. S. (2005). A Key Establishment Protocol for Bluetooth Scatternets. *University of Kentucky* .

Sairam, G. R. (2002). Bluetooth in Wireless Communication. *Topics in Broadband Access* .

Gorman, J. (s.f.). *Difference Between Bluetooth & WiFi Technology.* Recuperado el 2 de Noviembre de 2011, de eHow: http://www.ehow.com/facts_4761491_difference-between-bluetooth-wifi-technology.html

Martin, J. A. (2002). Mobile Computing Tips: Bluetooth vs. WiFi FAQ. *PC World* .

Libros

ABLESON Frank, "Android Guía para desarrolladores", Ediciones Anaya multimedia, España, 2010.

STEELE James, “The Android developer’s cookbook building applications with the Android SDK”, editorial Addison-Wesley, primer edición, Estados Unidos, 2010.

DIMARZIO Jerome, “Android a programmer’s guide”, Editorial McGraw Hill, Estados Unidos 2008.stma

ROGERS Rick, “Android application development”, editorial O’Reilly, Estados Unidos, 2009

SMITH Dave, “Android recipes a problem-solution approach”, editorial Apress, Estados Unidos, 2011

FELKER Donn, “Android application development for dummies”, editorial Wiley Publishing, 2011

MEIER Reto, “Professional android 2 application development”, Wiley Publishing, 2010.

CORNISH María Laura, “El ABC de los plásticos”, Universidad Iberoamericana, México, D.F., 1997.