UNIVERSIDAD CARLOS III DE MADRID

PROYECTO FIN DE CARRERA

Diseño e implementación de un vehículo a escala controlado por voz

Autor: Miguel López Aguilar
Tutor: Javier Fernández Muñoz
Titulación: Ingeniería informática

27/09/2015

Índice de contenidos

Índice de tablas

Índice de ilustraciones

1 Introducción

En este apartado servirá como introducción al documento y en él se trata la motivación que nos ha llevado a la realización del proyecto, así como los objetivos que con este se quieren conseguir. Para facilitar la compresión del documento se dedica un apartado para resumir la estructura a seguir.

1.1 Motivación

La tecnología del control de voz está en constante desarrollo y es una de las que más atrae a los usuarios, por lo tanto a día de hoy es una de las que más se implementan. Es una tecnología que se suele usar para realizar acciones de control como por ejemplo el control de una televisión, controlar la consola principal de un vehículo…

Actualmente se están creando diversas aplicaciones para Smartphone con la utilidad de utilizar su conectividad inalámbrica para controlar mediante voz otros elementos que de por si no tenían o no estaban pensados para desarrollar dicha actividad.

En este caso lo que se va a realizar es el diseño e implementación de un vehículo a escala controlado mediante voz, para ello se modificara un vehículo radiocontrol para controlarlo mediante un Smartphone con Android a través del Bluetooth. En este proyecto se controlará el vehículo de dos maneras distintas: Manual y Voz.

El control Manual se basa en controlar el vehículo mediante dos joysticks aprovechando la pantalla táctil del Smartphone, el joystick derecho controla el movimiento (delante-atrás) mientras el joystick derecho controla la dirección (izquierda-derecha).

El control de voz se basa en controlar el vehículo mediante comandos de voz, aprovechando el reconocimiento de voz ofrecido por Google, los comandos reconocidos son los siguientes: Adelante, atrás, para, izquierda, derecha, recto.

En esta memoria se detallan los pasos seguidos para el correcto diseño e implementación del vehículo, para que sirva de guía para futuros prototipos.

1.2 Objetivo

El principal objetivo del proyecto es diseñar e implementar un vehículo a escala, para controlarlo con un Smartphone con sistema operativo Android a través del Bluetooth. Para ello es necesario tener conocimientos generales sobre las placas electrónicas de arduino y programar sobre ellas, así como programación en Android.

El proyecto constará de dos partes:

- Parte hardware que consistirá en el diseño e implementación de un circuito eléctrico para que el arduino pueda recibir las indicaciones vía Bluetooth y tras ello sea capaz de controlar de controlar el vehículo.

- Parte software que consistirá en el diseño e implementación de dos aplicaciones para Android, así como programar la parte lógica de la placa Arduino.

Tras indicar cuál es el principal objetivo del proyecto, se pasará a dividirlos en objetivos a cumplir para cumplir el objetivo principal:

- ❖ Diseñar e implementar una aplicación para Android que nos permita conectarnos al vehículo y controlarlo a través de Bluetooth de manera manual.

- ❖ Diseñar e implementar una aplicación para Android que nos permita conectarnos al vehículo y controlarlo a través de Bluetooth mediante comandos de voz.

- ❖ Diseñar e implementar un Hardware que sea válido para controlar el vehículo y recibir instrucciones a través de Bluetooth.

- ❖ Diseñar e implementar un código que irá cargado en la placa Arduino para darle a esta una funcionalidad útil deseada.

Para complementar estos objetivos se proponen los siguientes sub-objetivos:

- ❖ Diseñar un sistema un sistema anti-choque para evitar que el vehículo colisione frontalmente.

- ❖ Diseñar unas interfaces muy intuitivas para permitir que un usuario inexperto sea capaz de controlar el vehículo.

Para concluir este apartado es necesario indicar que para cumplir el objetivo principal del proyecto es necesario diseñar e implementar todos los objetivos, así mismo se intentará complementar completando los sub-objetivo.

1.3 Estructura del documento

Para facilitar la lectura del proyecto se describe la estructura del documento y se realizará una pequeña descripción de las diferentes partes que lo componen.

1. **Introducción**: Parte introductoria en la que se explica la motivación, los objetivos y la estructura del documento.

2. **Estado del Arte**: Parte en la que se describe la situación actual y las diferentes alternativas que existen.

3. **Análisis del sistema**: Parte donde se indican las necesidades del usuario y se definen los requisitos y los casos de uso.

4. **Diseño e implementación**: Parte donde se describe el diseño y como este se ha implementado para alcanzar una solución factible.

5. **Pruebas y evaluación**: Parte donde se describe una planificación de las pruebas que se van a ejecutar para verificar el correcto funcionamiento de las aplicaciones.

6. **Planificación y presupuesto**: Parte en la que se describe la planificación que se va a seguir para el desarrollo del proyecto y una predicción del presupuesto que este tendrá.

7. **Conclusiones y trabajos futuros**: Parte en la cual se realizan las conclusiones del proyecto y la conclusión personal. Para concluir se incluirán las líneas que el proyecto podrá desarrollar para trabajos futuros o investigaciones futuras.

8. **Bibliografía**: Parte donde se encuentra las referencias utilizadas a lo largo del proyecto.

1.4 Definiciones y Acrónimos

En este apartado se realiza las definiciones y acrónimos. La mayoría de las definiciones fueron extraídas de la Wikipedia [30].

1.4.1 Definiciones

Arduino: Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

Application Package File(APK): Un archivo con extensión .apk (Application Package File) es un paquete para el sistema operativo Android. Este formato es una variante del formato JAR de Java y se usa para distribuir e instalar componentes empaquetados para la plataforma Android para Smartphone y tablets.

Android: Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tablets.

Baudios: El baudio es una unidad de medida utilizada en telecomunicaciones, que representa el número de símbolos por segundo en un medio de transmisión digital.

Bluetooth: Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz.

Chip: Un circuito integrado (CI), también conocido como chip o microchip, es una estructura de pequeñas dimensiones de material semiconductor, de algunos milímetros cuadrados de área, sobre la que se fabrican circuitos electrónicos generalmente mediante fotolitografía y que está protegida dentro de un encapsulado de plástico o de cerámica.

Emulador: Un emulador es un software que permite ejecutar programas o videojuegos en una plataforma (sea una arquitectura de hardware o un sistema operativo) diferente de aquella para la cual fueron escritos originalmente.

Eclipse: Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.

Entorno de programación: Un entorno de programación es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

Java Development Kit: Se trata de un conjunto de herramientas (programas y librerías) que permiten desarrollar (compilar, ejecutar, generar documentación, etc.) programas en lenguaje Java.

Hardware: Se conoce como hardware a todas las partes físicas de un sistema informático; sus componentes son: eléctricos, electrónicos. Electromecánicos y mecánicos.

Intent: Un intento es una descripción abstracta de una operación a realizar.

Interface de usuario: La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo.

Puente H: Un Puente en H es un circuito electrónico que permite a un motor eléctrico DC girar en ambos sentidos, avance y retroceso.

Software: Se conoce como software al equipo lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas.

Smartphone: El teléfono inteligente (en inglés: Smartphone) es un tipo teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades, semejante a la de una minicomputadora, y con una mayor conectividad que un teléfono móvil convencional.

Socket: El Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.

Sistema operativo: Un sistema operativo es un programa o conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación.

Software development kit: Un kit de desarrollo de software o SDK (siglas en inglés de software development kit) es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador o desarrollador de software crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, computadoras, videoconsolas, sistemas operativos, etcétera.

ZIP: ZIP es un formato de compresión sin pérdida, muy utilizado para la compresión de datos como documentos, imágenes o programas.

1.4.2 Acrónimos

APK: Application Package File.

ADT: Abstract Data Type.

DC: Corriente continua.

E/S: Entrada y salida.

IDE: Integrated Development Environment

JDK: Java Development Kit.

KB: Kilobytes.

KHz: Kilohercio.

mA: miliamperios.

MHz: megahercio.

R/C: Radiocontrol.

SDK: Software development kit.

UUID: Universally Unique ID

µs: micro segundo.

2 Estado del arte

2.1 Sistemas Hardware-Software controlados por voz

En este apartado se muestran algunos ejemplos de sistemas controlados mediante voz, para así poder realizar un análisis y mostrar las diferentes ventajas de una interfaz por voz. Así mismo se muestra la situación actual del mercado tanto en Hardware como en software, para crear un producto lo más innovador posible y adecuado a las necesidades del usuario.

2.1.1 Sistema de control por voz en los vehículos

En la actualidad la seguridad en los vehículos es fundamental. A día de hoy los vehículos están en constante desarrollo y por lo tanto cada vez incorporan más funciones difíciles de controlar si se está conduciendo, para evitar estas distracciones una solución muy sencilla, controlar el vehículo mediante comandos de voz ya que así el conductor no tendría que apartar la vista de la carretera lo que mejoraría la seguridad.

Ilustración 1: Sistema de control por voz en los vehículos

2.1.2 Televisiones controladas por voz y gestos

El nuevo salto tecnológico en televisiones son los Smart TV y su nuevo objetivo es ser usada sin el mando a distancia tradicional, sino que basa su control en instrucciones por voz y gestos. Como ventaja de esta interfaz indicar que para encenderla solo tendremos que comentar un "Hola tele encender", y a partir de que esté encendida reconocerá diferentes comandos como: "subir volumen", "bajar canal", "Ir a Menú", "Programación"… Sin que tengas que buscar el mando a distancia, ni preocuparte de si este tiene pilas.

Ilustración 2: Televisiones controladas por voz y gestos

2.1.3 Casa domótica controlada por voz

Actualmente las casas inteligentes se están convirtiendo en realidad, en este caso lo que nos interesa son las controladas mediante la interfaz de voz. Estas casas con esta interfaz son perfectas para personas con discapacidad sobre todo con deficiencias motrices, ya que podrán tener el control de la casa sin realizar ningún movimiento. El sistema es sencillo el usuario solo tendrá que dar una serie de instrucciones fáciles de comprender cómo: "Subir persiana comedor", "encender luz del pasillo", "temperatura del salón a 25 grados"… y así tendrá el control de la mayoría de elementos electrónicos de la casa. También se puede controlar la casa aun estando fuera de ella mediante el teléfono móvil, esto también es de gran utilidad ya que como ejemplo podrás encender el robot de cocina a distancia para que cuando llegues la comida esté lista y recién hecha. Un ejemplo de esta tecnología es el sistema *Ubi*.

Ilustración 3: Casa domótica controlada por voz

2.2 Herramientas para el desarrollo hardware

En este apartado se realizará una descripción general de los diferentes materiales hardware utilizado, así como su uso en este proyecto con el fin de facilitar su compresión y las diferentes alternativas que puede tener cada uno de ellos.

2.2.1 Arduino mega

El arduino mega se utilizó para recibir las instrucciones del Smartphone y transformarlas en el movimiento del vehículo. Información obtenida de la página oficial [4].

“El Mega Arduino es una placa electrónica basada en el . Cuenta con 54 pines digitales de entrada / salida (de los cuales 14 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (hardware puertos serie), un 16 MHz oscilador de cristal, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador; simplemente conectarlo a un ordenador con un cable USB o el poder con un adaptador de CA o la batería a CC para empezar. “

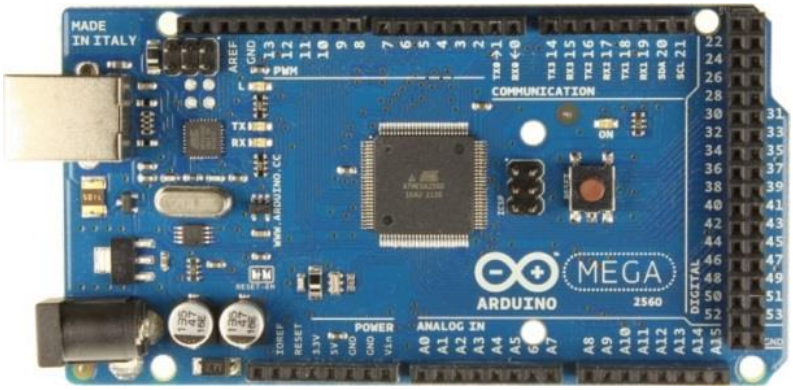


Ilustración 4: Arduino Mega

Especificaciones	
Micro controladores	ATmega1280
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12 voltios
Voltaje de entrada (límites)	6-20 voltios
Digital pines I/O	54 (de las cuales 15 proporcionan salida PWM)
Pines de entrada analógica	16
Corriente DC por E/S Pin	40 mA
Corriente DC de 3.3V Pin	50 mA
Memoria flash	128 KB de los cuales 4 KB utilizado por el gestor de arranque
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

Tabla 1: Especificaciones Arduino

2.2.2 Módulo Bluetooth

El módulo Bluetooth (HC-06) es el utilizado para la comunicación entre el arduino y el Smartphone.

Ilustración 5: Módulo Bluetooth

Especificaciones	
Tensión de alimentación	3.3 voltios – 6 voltios
Velocidad de transferencia de datos	1200-1382400 baudios
Velocidad de transferencia de datos por defecto	9600 baudios
El alcance de la comunicación en la línea de visión	30 metros
Clave por defecto	1234
Versión del Bluetooth	V2.0+ EDR
Nombre por defecto	HC-06

Tabla 2: Especificaciones Módulo Bluetooth

2.2.3 Chip L293B

Se utiliza para la realización de un puente H con transistores, para poder controlar los motores de corriente continua. Con este puente H conseguiremos que los motores eléctricos de corriente continua giren en ambos sentidos, consiguiendo así el giro de izquierda a derecha y la marcha adelante y marcha atrás.

Ilustración 6: Chip L293B

2.2.4 LCD Keypad Shield

LCD Keypad Shield es un módulo para arduino, con el cual le mostraremos al usuario en el modo en el que se encuentra el vehículo e información del mismo.

Ilustración 7: LCD Keypad Shield

Tabla

Especificaciones	
Voltaje de alimentación	5v DC
5 botones de funcionamiento	Select, left, up, down y rigth
Botón RST	Reiniciar el programa arduino
Ping digital 10 de arduino	Control de luz de fondo

Tabla 3: Especificaciones LCD Keypad Shield

2.2.5 Módulo HC-SR04

El módulo HC-SR04 es un sensor de distancia de proximidad, es utilizado para evitar que el vehículo impacte de frente contra obstáculos. Para ello utiliza la tecnología de los ultrasonidos y calcula la distancia con el tiempo que tarda en rebotar el sonido con los obstáculos.

Ilustración 8: Módulo HC-SR04

Especificaciones	
Voltaje de alimentación	5 voltios DC
Corriente:	15 mA
Angulo de cobertura	< 15º
Rango de medición	2 centímetros – 4 metros
Frecuencia de operación	40 KHZ
4 pines	VCC, Trig, Echo, GND

Tabla 4: Especificaciones Módulo HC-SR04

2.2.6 Vehículo controlado por R/C a escala

El vehículo controlado por R/C a escala, será un juguete (preferiblemente que no funcione o no tenga ningún uso) que se desmontará para quitarle los circuitos internos dejando solo los dos motores de corriente continua (Ilustración 10: Motor Corriente).

Ilustración 9: Vehículo controlado por R/C a escala

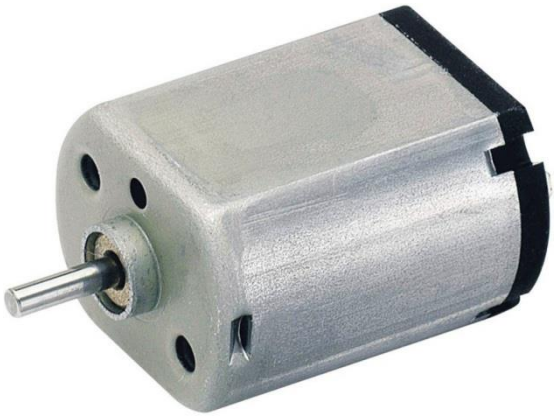


Ilustración 10: Motor Corriente continua

2.2.7 Alimentador de arduino

El alimentador de arduino es utilizado para alimentar a la placa de arduino y que esta funcione sin tener que estar conectada con el cable de E/S a un ordenador.

Ilustración 11: Alimentador de arduino

2.2.8 Cables

Los cables son utilizados para las conexiones entre los componentes, para que pueda existir una comunicación entre ellos.

Ilustración 12: Cables

2.2.9 Pila 9V

La pila de 9V (Voltios) se utilizará para alimentar la placa de arduino mega y que así esta funcione y no tenga que estar conectada al ordenador.

Ilustración 13: Pila 9V

2.2.10 Placa de pruebas

La placa de pruebas [19] es una placa en la cuales se conectan los componentes, para más tarde cablearlos, esta placa nos evita realizar soldaduras y así podemos modificar el producto de una manera rápida y sencilla.

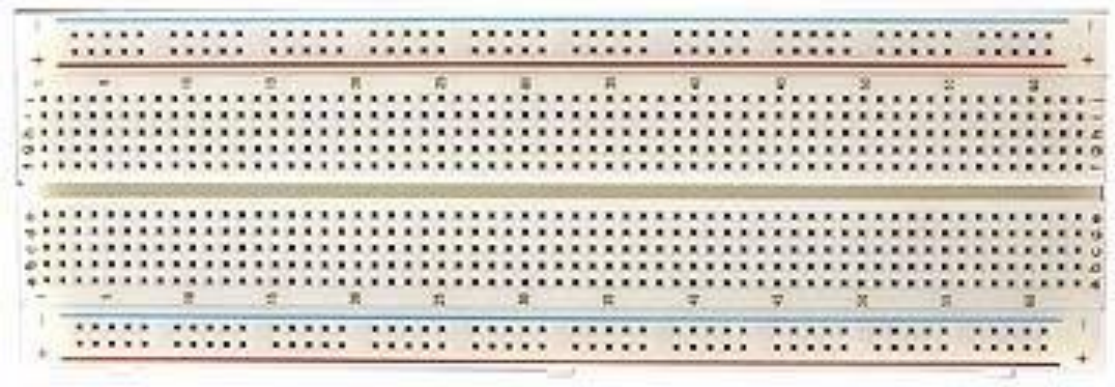


Ilustración 14: Placa de pruebas

2.2.11 Smartphone con sistema operativo Android

Para la instalación de las aplicaciones que controlan el vehículo se ha utilizado un Orange MonteCarlo con la versión del sistema operativo Android Gingerbread 2.3.5 [17].

Ilustración 15: Smartphone con sistema operativo Android

2.3 Herramientas para el desarrollo Software

En este apartado se realiza una descripción general de las diferentes herramientas de programación utilizadas y las diferentes alternativas que puede tener cada uno de ellos, así como su uso en este proyecto con el fin de facilitar su compresión

2.3.1 Entorno de programación para arduino

El entorno de programación para arduino es usado para escribir el código y cargarlo en placa de arduino mega (programar la placa).

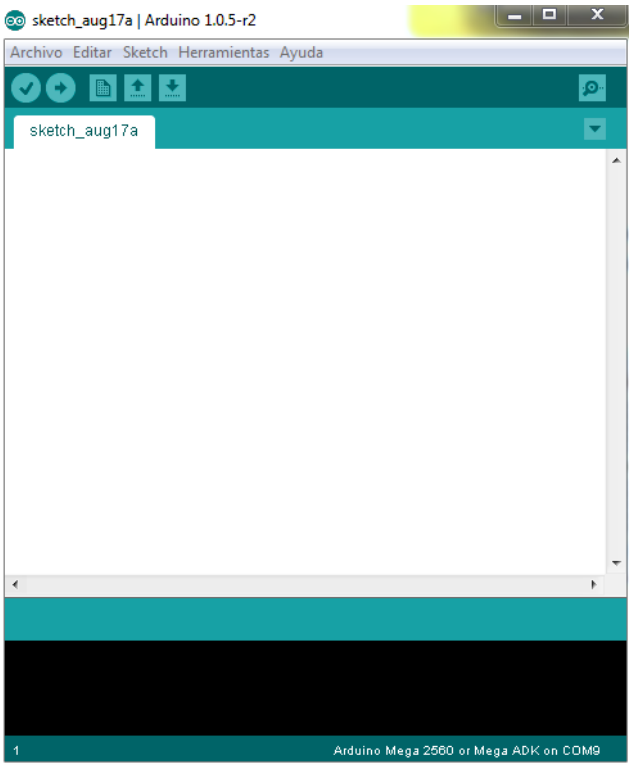


Ilustración 16: Entorno de programación para arduino

El entorno se descargara de la página oficial de arduino [6].

En este caso se descargó la versión más reciente (ARDUINO 1.6.5), dando igual cual descargar si el instalador o en su defecto el archivo ZIP.

Ilustración 17: Software arduino

En el caso de que no tengas conocimiento en arduino puedes consultar el manual de referencia que se encuentra en su página oficial [5].

2.3.2 Alternativas al entorno de programación para arduino

En este apartado se describirán las alternativas al entorno de programación para arduino, en este caso son unas alternativas con una interfaz gráfica extraídas de la siguiente web [50].

2.3.2.1 Minibloq

Minibloq funciona traduciendo los bloques gráficos a código Arduino permitiendo además programar la placa.

Funcionalidades:

- Funciona para todas las placas de Arduino.
- Minibloq funciona con Windows, y la versión para Linux está en versión beta.
- Permite visualizar las entradas y configurar las Entradas y Salidas.
- Como inconveniente podemos señalar que los bloques no resultan muy intuitivos si vienes de usar *Scratch*.

2.3.2.2 Modkit

Modkit está todavía en fase de desarrollo con una versión *Alpha*. Tiene una versión libre y otra de pago.

Funcionalidades:

- Funciona en las tres plataformas.
- Traduce el código realizado mediante bloques y programa la placa.
- Detecta automáticamente la versión de la placa.
- Entorno de programación atractivo con bloques inspirados en *Scratch* fácilmente identificables.

2.3.2.3 Ardublock

Ardublock traduce directamente los bloques en código en el IDE de Arduino actuando como un plugin externo, para lo cual debemos realizar algunas configuraciones en el IDE de Arduino.

Funcionalidades:

- Funciona con los 3 sistemas operativos Windows, Linux y Mac.
- El entorno gráfico quizás no es tan atractivo como el de *Scratch*

2.3.2.4 Blocklyduino

Blockly es un entorno gráfico de programación online desarrollado por Google para aprender a programar, incluye juegos y diversas aplicaciones.

Uno de los proyectos externos es *Blocklyduino* que transforma los bloques de programación en código Arduino que luego tendríamos que cargar en la placa con el IDE de Arduino.

Aunque está todavía en fase de desarrollo, los programas básicos que he probado los traducía correctamente.

2.3.2.5 S4A

S4A es una modificación del popular *Scratch* 1.4 creado por el MIT, para trabajar con Arduino, la adaptación ha sido creada por *citilab*.

La principal ventaja es que si ya hemos trabajado previamente con *Scratch* nos resultará muy cómodo adaptarnos a las nuevas funcionalidades para controlar Arduino

Conserva las funcionalidades de *Scratch* con lo que nos permite relacionar el mundo real y el virtual, por ejemplo añadiendo mandos a nuestros videojuegos

Funciona en todas las plataformas Windows, Linux y MAC

Como inconveniente cabe señalar que Arduino tiene que estar conectado al PC para funcionar ya que es el programa de S4A el que recibe el valor de los sensores y decide las actuaciones mientras en Arduino opera un programa de comunicación

Otro inconveniente es que los pines vienen preconfigurados y a nivel usuario no se pueden cambiar

2.3.2.6 Snap4Arduino

Snap es una herramienta basada en *Scratch*, con la particularidad de que puedes construir tus propios bloques.

Snap4Arduino es una versión de Snap desarrollada por *citilab* para trabajar con Arduino, está todavía en versión *Alpha*.

Funcionalidades:

- Funciona tanto con Linux, como con Windows y MAC
- Permite configurar los pines de salida
- Soporta todo tipo de versiones de las placas de Arduino, incluido la Mega

2.3.3 Entorno de programación para Android: Eclipse

Ilustración 18: Eclipse Luna

Para la programación en Android se utilizó el software de programación eclipse en su versión luna. Este fue el entorno elegido para desarrollar las aplicaciones para Android ya que es el entorno utilizado durante la carrera y sobre el que más conocimiento base tenemos. Actualmente es un entorno peor para programar en Android que Android Studio como demuestra la siguiente comparativa:

Características	Android Studio	Eclipse ADT
Sistema de construcción	Gradle	ANT
Construcción y gestión de proyectos basado en Maven (herramienta de software para la gestión y construcción de proyectos Java, similar a Apache ANT, pero su modelo es más simple ya que está basado en XML)	Si	No (es necesario instalar un plugin auxiliar)
Construir variantes y generación de múltiples APK (muy útil para Android Wear)	Si	No

Refactorización y completado avanzado de código Android	Si	No
Diseño del editor gráfico	Si	Si
Firma APK y gestión de almacén de claves	Si	Si
Soporte para NDK (Native Development Kit: herramientas para implementar código nativo escrito en C y C++)	Próximas versiones	Si
Soporte para Google Cloud Platform	Si	No
Vista en tiempo real de renderizado de layouts	Si	No
Nuevos módulos en proyecto	Si	No
Editor de navegación	Si	No
Generador de assets	Si	No
Datos de ejemplo en diseño de layout (sin renderizar en tiempo de ejecución)	Si	No
Visualización de recursos desde editor de código	Si (a la izquierda de la línea de asignación del recurso)	No

Ilustración 19: Comparativa Android Studio VS Eclipse

2.3.4 Alternativas al entorno de programación para Android

En este apartado se describirán las alternativas al entorno de programación para Android, en este caso la mejor alternativa es Android Studio que es una versión más moderna y reemplaza al SDK de eclipse y el plugin ADT. Una de las ventajas que tiene sobre este es una instalación más sencilla, además de ser de Google y por lo tanto basado en el IDE de *Intellij* IDEA se presenta como la alternativa e IDE oficial de Android, nos ofrece un mejor editor de diseño que Eclipse e incluye la integración con *Google Cloud Messaging* para poder ver los cambios en el acto con diferentes resoluciones.

Las ventajas más resaltables son las siguientes:

1. Mejor reutilización de código y recursos.
2. Facilita la distribución del código.
3. Herramientas *Lint*.
4. Mejor gestión de las dependencias.
5. Permite compilar desde la línea de comandos.
6. Facilita la creación de versiones de la aplicación.
7. Reestructuración de código y soluciones más rápidas.
8. Herramientas específicas, mejor rendimiento, usabilidad y compatibilidad.
9. Función para firmar aplicaciones.
10. Más y mejores plantillas base para crear aplicaciones.
11. Mejoras en el editor gráfico.
12. Soporte para *Google Cloud Platform*.
13. Compilador *Graddle*: *Gradle* es la herramienta que automatiza la construcción de las aplicaciones, ya tiene montadas las tareas para las mayoría de los proyectos por defecto, usa *Groovy* como lenguaje, tiene soporte para ANT, soporte multi-proyectos y maneja compilaciones incrementales, lo cual nos ahorrará mucho tiempo en el desarrollo de proyectos.

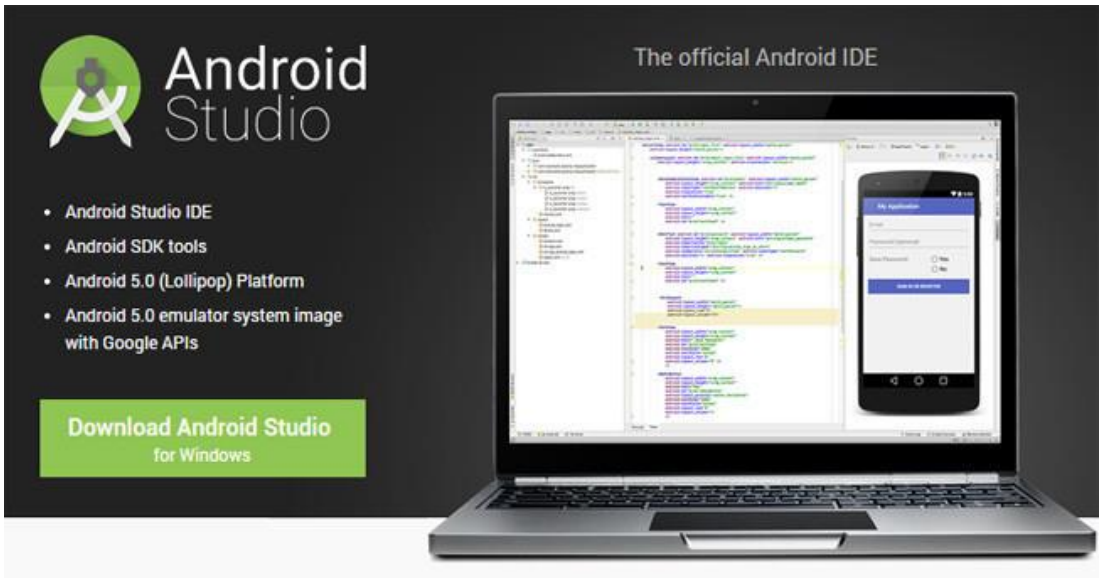


Ilustración 20: Alternativas al entorno de programación para Android

3 Análisis del sistema

En este apartado se realiza un Análisis del sistema. Para este análisis se realiza una definición de los requisitos del sistema para para las aplicaciones de control manual, control por voz y de la aplicación que se encuentra cargada en la placa de arduino.

3.1 Requisitos de usuario

Los requisitos de usuario, indican qué funciones puede o tiene que hacer la aplicación para cumplir los propósitos requeridos para el sistema.

A continuación, se detallan los requisitos de usuario para las aplicaciones de control manual, control por voz y de la aplicación para Arduino.

Para definir los requisitos de usuario que tendrá nuestro sistema, usaremos la siguiente tabla:

Identificador	RUV/RUM/RUA-XX		
Título	Titulo identificativo		
Necesidad	(3)Esencial/(2)Deseable/ (1)Opcional	**Prioridad**	(3)Alta/(2)Media/(1)Baja
Estabilidad		**Verificabilidad**	(3)Alta/(2)Media/(1)Baja
Descripción	Descripción del requisito de usuario.		

Tabla 5: Estructura de tabla de requisitos de usuario

Identificador: Cada requisito estará identificado por RUV-XX o RUM-XX o RUA-XX. "RUV" será para los requisitos de usuario para el control mediante voz, "RUM" será para para el control manual, mientras que "RUA" serán los requisitos de usuario para la aplicación de Arduino. Además "XX" identificará el número del requisito en función de su tipo, siendo "01" el primero y "99" el último.

Título: Titulo breve que identifique inequívocamente el propósito del requisito.

Necesidad: La necesidad irá representada por un número, siendo 1 el más bajo, que representará un requisito opcional; y 3 el más alto, que pertenecerá a un requisito esencial para el usuario, siendo imprescindible su inclusión en el sistema.

Prioridad: Este campo contendrá el nivel de prioridad del requisito de cara a su implementación posterior en el sistema. Puede tomar tres valores posibles: siendo 1 el nivel prioritario más bajo, 2 el intermedio y 3 el nivel más alto.

Verificabilidad: Se identificará con un número del 1 al 3. Los requisitos poco verificables tendrán un 1, mientras que los fácilmente verificables un 3.

Estabilidad: Indicará el periodo de vida del requisito en la aplicación. El objetivo será que cuanto más estable sea el requisito mejor.

Descripción: Contendrá una explicación del requisito, que refleja una cualidad o funcionalidad que el usuario desea o puede desear en algún momento.

3.1.1 Requisitos para el control por voz

En este apartado se especificaran los requisitos para el control mediante voz.

Identificador	RUV-01		
Título	Habilitar Conexión inalámbrica		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Se debe poder habilitar una conexión inalámbrica.		

Tabla 6: Requisitos de Usuario RUV-01

Identificador	RUV-02		
Título	El Smartphone podrá conectarse con el vehículo		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Se debe poder realizar una conexión inalámbrica con el vehículo.		

Tabla 7: Requisitos de Usuario RUV-02

Identificador	RUV-03		
Título	Reconocer comandos de voz		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Se debe poder mover el coche (adelante, atrás y girar a izquierda y derecha) con comandos de voz.		

Tabla 8: Requisitos de Usuario RUV-03

Identificador	RUV-04		
Título	Funcionamiento de la Aplicación		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	La aplicación tiene que funcionar y visualizarse correctamente ciertas versiones de Android.		

Tabla 9: Requisitos de Usuario RUV-04

Identificador	RUV-05		
Título	Idioma		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El idioma de la aplicación será español.		

Tabla 10: Requisitos de Usuario RUV-05

Identificador	RUV-06		
Título	Diseño Visual		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El diseño visual tiene que ser fácil e intuitivo por lo que un usuario inexperto debe poder usar la aplicación sin problemas.		

Tabla 11: Requisitos de Usuario RUV-06

Identificador	RUV-07		
Título	Tiempo de conexión		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El tiempo de conexión al vehículo no debe superar los 30 segundos.		

Tabla 12: Requisitos de Usuario RUV-07

Identificador	RUV-08		
Título	Salida de la aplicación		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario de la aplicación no podrá salir de la aplicación de manera inesperada.		

Tabla 13: Requisitos de Usuario RUV-08

3.1.2 Requisitos para el control manual

En este apartado se especificaran los requisitos para el control Manual.

Identificador	RUM-01		
Título	Habilitar Conexión inalámbrica		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Se debe poder habilitar una conexión inalámbrica.		

Tabla 14: Requisitos de Usuario RUM-01

Identificador	RUM-02		
Título	El Smartphone podrá conectarse con el vehículo		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Se debe poder realizar una conexión inalámbrica con el vehículo.		

Tabla 15: Requisitos de Usuario RUM-02

Identificador	RUM-03		
Título	Reconocer movimiento tocando la pantalla		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Se debe poder mover el coche (adelante, atrás y girar a izquierda y derecha) tocando la pantalla.		

Tabla 16: Requisitos de Usuario RUM-03

Identificador	RUM-04		
Título	Funcionamiento de la Aplicación		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	La aplicación tiene que funcionar y visualizarse correctamente para versiones de Android superiores e iguales a Android Gingerbread 2.3.5 [17].		

Tabla 17: Requisitos de Usuario RUM-04

Identificador	RUM-05		
Título	Idioma		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El idioma de la aplicación será español.		

Tabla 18: Requisitos de Usuario RUM-05

Identificador	RUV-06		
Título	Diseño Visual		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El diseño visual tiene que ser fácil e intuitivo por lo que un usuario inexperto debe poder usar la aplicación sin problemas.		

Tabla 19: Requisitos de Usuario RUM-06

Identificador	RUM-07		
Título	Tiempo de conexión		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El tiempo de conexión al vehículo no debe superar los 30 segundos.		

Tabla 20: Requisitos de Usuario RUM-07

Identificador	RUM-08		
Título	Salida de la aplicación		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario de la aplicación no podrá salir de la aplicación de manera inesperada.		

Tabla 21: Requisitos de Usuario RUM-08

3.1.3 Requisitos de la aplicación para Arduino

En este apartado se especificaran los requisitos de la aplicación que será cargada en la placa Arduino.

Identificador	RUA-01		
Título	Informar al usuario		
Necesidad	1	**Prioridad**	2
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Deberá informar al usuario del modo en el que se encuentra		

Tabla 22: Requisitos de Usuario RUA-01

Identificador	RUA-02		
Título	Evitará el choque frontal		
Necesidad	1	**Prioridad**	2
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Deberá Evitar el choque frontal		

Tabla 23: Requisitos de Usuario RUA-02

Identificador	RUA-03		
Título	Funcionará con las aplicaciones		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Tendrá que funcionar correctamente con las dos aplicaciones del Smartphone.		

Tabla 24: Requisitos de Usuario RUA-03

Identificador	RUA-04		
Título	Tiempo Respuesta		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El tiempo de respuesta para ejecutar las instrucciones tiene que ser inferior a 0.5 segundos.		

Tabla 25: Requisitos de Usuario RUA-04

Identificador	RUA-05		
Título	Uso de la batería		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	La batería debe de durar al menos 5 minutos.		

Tabla 26: Requisitos de Usuario RUA-05

3.2 Casos de uso

En este apartado se va a exponer la especificación de los casos de uso del sistema para definir su diseño, es decir, cómo diferentes actores actúan con el sistema.

En este caso vamos a tener tres posibles escenarios:

- ✓ Escenario 1: Aplicación de control manual.
- ✓ Escenario 2: Aplicación de control por voz.
- ✓ Escenario 3: Aplicación cargada en Arduino.

Para la descripción de los casos de uso se utiliza la siguiente plantilla:

Caso de uso	
Identificador	CDU-XX-M/V/A
Título	
Actores	
Descripción	
Precondición	
Postcondiciones	
Condiciones de fallo	

Tabla 27: Plantillas caso de uso

Dónde:

- ✓ Identificador: Cada caso de uso estará identificado por las siglas CDU (Caso de Uso). "XX" identificará el número del caso de uso, siendo "01"el primero y "99" el último.
  "M" identifica a los de la aplicación Manual, "V" identifica a los de la aplicación por voz y "A" identifica a los de la aplicación que ira cargada en Arduino.
- ✓ Título: Breve encabezado que identifique inequívocamente el propósito del caso de uso.
- ✓ Actores: Indicará el tipo o tipos de usuario que pueden intervenir en el caso de uso.
- ✓ Descripción: Contendrá una explicación del caso de uso, que refleja una cualidad o funcionalidad que el usuario desea o puede desear en algún momento.
- ✓ Precondiciones: Condiciones que se deben cumplir para llevar a cabo la operación.
- ✓ Postcondiciones: Estado del sistema tras realizar la operación.
- ✓ Condiciones de fallo: Posibles errores que pueden ocurrir durante la ejecución de la operación.

3.2.1 Escenario 1: Aplicación de control manual.

A continuación se mostrará el caso del uso para la aplicación del control manual.

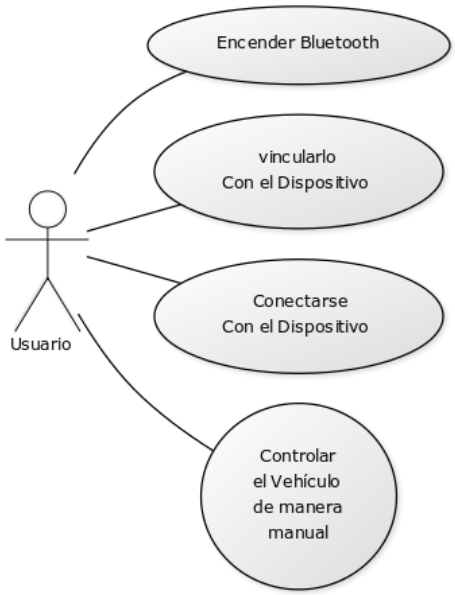


Ilustración 21: Casos de uso Aplicación control manual

A continuación pasamos a detallar cada uno de los casos de uso:

Caso de uso	
Identificador	CDU-01-M
Título	Encender Bluetooth
Actores	Usuario
Descripción	El usuario tendrá que ser capaz de encender el Bluetooth en el caso de que este esté apagado, ya que el dispositivo lo necesita para comunicarse.
Precondición	Que el Bluetooth este apagado en el Smartphone.
Postcondiciones	El Bluetooth quedara activado en el Smartphone.
Condiciones de fallo	Que no se dispongan de los permisos suficientes para Activar el Bluetooth.

Tabla 28: Caso de uso CDU-01-M

Caso de uso	
Identificador	CDU-02-M
Título	Vincularlo Con el Dispositivo
Actores	Usuario
Descripción	Para poder comunicarse con el vehículo es necesario que el Smartphone y el dispositivo Bluetooth del vehículo estén vinculados.
Precondición	Que el Smartphone y el Bluetooth del vehículo no estén vinculados.
Postcondiciones	El Smartphone y el Bluetooth del vehículo quedaran vinculados.
Condiciones de fallo	Que se introduzca mal la contraseña.

Tabla 29: Caso de uso CDU-02-M

Caso de uso	
Identificador	CDU-03-M
Título	Conectarse Con el Dispositivo
Actores	Usuario
Descripción	Para poder comunicarse con el vehículo es necesario que el Smartphone y el dispositivo Bluetooth estén conectados mediante un socket.
Precondición	Que el Smartphone y el Bluetooth del vehículo estén vinculados.
Postcondiciones	El Smartphone y el Bluetooth del vehículo quedaran conectados mediante un socket.
Condiciones de fallo	Que se produzca un error al establecer la conexión.

Tabla 30: Caso de uso CDU-03-M

Caso de uso	
Identificador	CDU-04-M
Título	Controlar el Vehículo
Actores	Usuario
Descripción	Al usuario se le tiene que ofrecer una interfaz para el control manual, para que sea capaz de controlar el vehículo mediante instrucciones.
Precondición	Que el Smartphone y el vehículo estén conectados.
Postcondiciones	Se le permitirá al usuario realizar las siguientes instrucciones de control del vehículo: Mover hacia delante, Mover hacia atrás, Girar hacia izquierda, Girar hacia derecha, Parar y recto.
Condiciones de fallo	Perdidas de datos por ser un sistema inalámbrico y saturación del socket.

Tabla 31: Caso de uso CDU-04-M

3.2.2 Escenario 2: Aplicación de control mediante comandos de voz.

A continuación se mostrará el caso del uso para la aplicación del mediante comandos de voz.

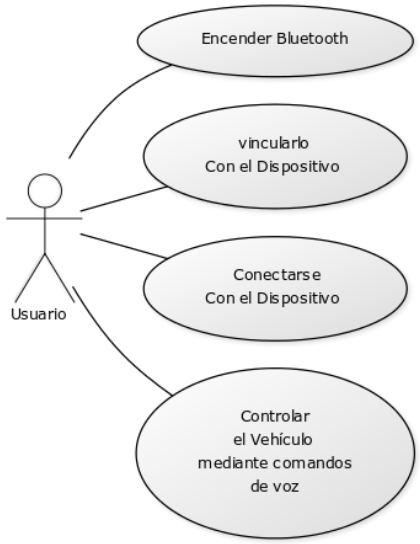


Ilustración 22: Casos de uso Aplicación control mediante comandos de voz

A continuación pasamos a detallar cada uno de los casos de uso:

Caso de uso	
Identificador	CDU-01-V
Título	Encender Bluetooth
Actores	Usuario
Descripción	El usuario tendrá que ser capaz de encender el Bluetooth en el caso de que este esté apagado, ya que el dispositivo lo necesita para comunicarse.
Precondición	Que el Bluetooth este apagado en el Smartphone.
Postcondiciones	El Bluetooth quedara activado en el Smartphone.
Condiciones de fallo	Que no se dispongan de los permisos suficientes para Activar el Bluetooth.

Tabla 32: Caso de uso CDU-01-V

Caso de uso	
Identificador	CDU-02-V
Título	Vincularlo Con el Dispositivo
Actores	Usuario
Descripción	Para poder comunicarse con el vehículo es necesario que el Smartphone y el dispositivo Bluetooth del vehículo estén vinculados.
Precondición	Que el Smartphone y el Bluetooth del vehículo no estén vinculados.
Postcondiciones	El Smartphone y el Bluetooth del vehículo quedaran vinculados.
Condiciones de fallo	Que se introduzca mal la contraseña.

Tabla 33: Caso de uso CDU-02-V

Caso de uso	
Identificador	CDU-03-V
Título	Conectarse Con el Dispositivo
Actores	Usuario
Descripción	Para poder comunicarse con el vehículo es necesario que el Smartphone y el dispositivo Bluetooth estén conectados mediante un socket.
Precondición	Que el Smartphone y el Bluetooth del vehículo estén vinculados.
Postcondiciones	El Smartphone y el Bluetooth del vehículo quedaran conectados mediante un socket.
Condiciones de fallo	Que se produzca un error al establecer la conexión.

Tabla 34: Caso de uso CDU-03-V

Caso de uso	
Identificador	CDU-04-V
Título	Controlar el vehículo mediante comando de voz
Actores	Usuario
Descripción	Al usuario se le tiene que ofrecer una interfaz para el control mediante voz, para que sea capaz de controlar el vehículo mediante instrucciones.
Precondición	Que el Smartphone y el vehículo estén conectados.
Postcondiciones	Se le permitirá al usuario realizar las siguientes instrucciones de control del vehículo: Mover hacia delante, Mover hacia atrás, Girar hacia izquierda, Girar hacia derecha, Parar y recto.
Condiciones de fallo	Perdidas de datos por ser una conexión inalámbrica y saturación del socket.

Tabla 35: Caso de uso CDU-04-V

3.2.3 Escenario 3: Aplicación cargada en Arduino.

A continuación se mostrará el caso del uso para la aplicación cargada en Arduino.

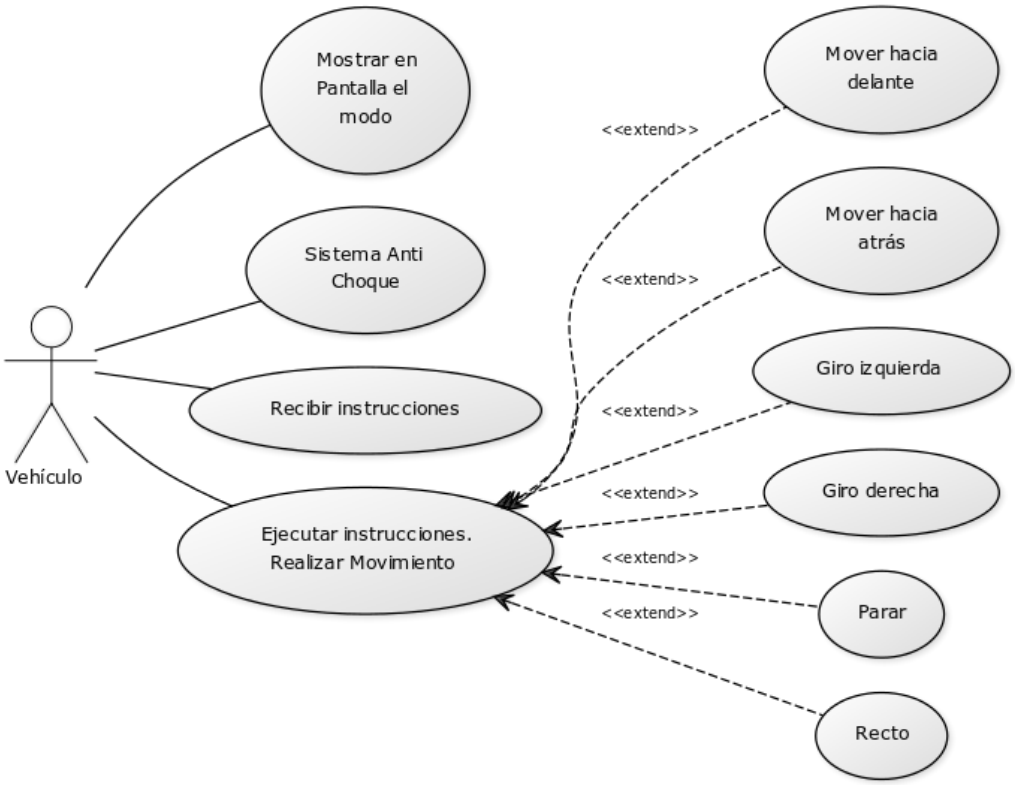


Ilustración 23: Casos de uso Aplicación cargada en Arduino

A continuación pasamos a detallar cada uno de los casos de uso:

Caso de uso	
Identificador	CDU-01-A
Título	Mostrar en Pantalla el modo
Actores	Vehículo
Descripción	El vehículo tiene que mostrar al usuario el modo en el que se encuentra.
Precondición	Que el Smartphone y el vehículo estén conectados. Y que se esté ejecutando una aplicación.
Postcondiciones	La pantalla cambiara de texto dependiendo del modo en el que se encuentre: Si Modo, Manual o voz.

Condiciones de fallo	Rotura de la pantalla o que lo ponga mal escrito.

Tabla 36: Caso de uso CDU-01-A

Caso de uso	
Identificador	CDU-02-A
Título	Sistema Anti-choque
Actores	Vehículo
Descripción	El vehículo dispondrá de un sistema anti-choque para evitar que el usuario colisione de frente.
Precondición	No se aplica
Postcondiciones	El vehículo retrocederá en un breve instante de tiempo para simular una frenada.
Condiciones de fallo	Que el frenado no sea suficiente y el vehículo impacte frontalmente o que no se calcule correctamente la distancia al objetivo.

Tabla 37: Caso de uso CDU-02-A

Caso de uso	
Identificador	CDU-03-A
Título	Recibir instrucciones
Actores	Vehículo
Descripción	El vehículo tendrá que estar todo el tiempo intentando recibir instrucciones de control enviadas por el Smartphone.
Precondición	Que el Smartphone y el vehículo estén conectados
Postcondiciones	El vehículo recogerá las instrucciones para poder ejecutarlas.
Condiciones de fallo	Perdidas de datos por ser un sistema inalámbrico y saturación del socket.

Tabla 38: Caso de uso CDU-03-A

Caso de uso	
Identificador	CDU-04-A
Título	Ejecutar instrucciones. Realizar Movimiento
Actores	Vehículo
Descripción	El vehículo tendrá ejecutar las instrucciones recibidas para transformarla en el movimiento del vehículo.
Precondición	Que el Smartphone y el vehículo estén conectados
Postcondiciones	Transformar las instrucciones en movimiento.
Condiciones de fallo	Que se realice un movimiento no indicado.

Tabla 39: Caso de uso CDU-04-A

3.3 Requisitos software del sistema

A continuación, se detallan los requisitos software del sistema para las aplicaciones de control manual, control por voz y de la aplicación para Arduino.

Para definir los requisitos del sistema que tendrá nuestro sistema, usaremos la siguiente tabla:

Identificador	RSV/RSM/RUA-XX		
Título	Titulo identificativo		
Necesidad	(3)Esencial/(2)Deseable/ (1)Opcional	**Prioridad**	(3)Alta/(2)Media/(1)Baja
Estabilidad		**Verificabilidad**	(3)Alta/(2)Media/(1)Baja
Descripción	Descripción del requisito de usuario.		
Dependencias RU			

Tabla 40: Estructura de tabla de requisitos del sistema

Identificador: Cada requisito estará identificado por RSV-XX o RSM-XX o RSA-XX. "RSV" será para los requisitos del sistema para el control mediante voz, "RSM" será para para el control manual, mientras que "RSA" serán los requisitos del sistema para la aplicación de Arduino. Además "XX" identificará el número del requisito en función de su tipo, siendo "01" el primero y "99" el último.

Título: Título breve que identifique inequívocamente el propósito del requisito.

Necesidad: La necesidad irá representada por un número, siendo 1 el más bajo, que representará un requisito opcional; y 3 el más alto, que pertenece a un requisito esencial para el usuario, siendo imprescindible su inclusión en el sistema.

Prioridad: Este campo contendrá el nivel de prioridad del requisito de cara a su implementación posterior en el sistema. Puede tomar tres valores posibles: siendo 1 el nivel prioritario más bajo, 2 el intermedio y 3 el nivel más alto.

Verificabilidad: Se identificará con un número del 1 al 3. Los requisitos poco verificables tendrán un 1, mientras que los fácilmente verificables un 3.

Estabilidad: Indicará el periodo de vida del requisito en la aplicación. El objetivo será que cuanto más estable sea el requisito mejor.

Descripción: Contendrá una explicación del requisito, que reflejará una cualidad o funcionalidad que el usuario desea o puede desear en algún momento.

Dependencias RU: Indica qué requisito de usuario se evalúa o implementa con dicho requisito software.

3.3.1 Requisitos para el control por voz

En este apartado se especificaran los requisitos para el control mediante voz.

Identificador	RSV-01		
Título	Activación del Bluetooth		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario podrá activar el Bluetooth desde la aplicación. Se le mostrar al usuario un mensaje en el que se le indique que el Bluetooth está desactivado y que tiene que activarlo.		
Dependencias RU	RUV-01		

Tabla 41: Requisitos del Sistema RSV-01

Identificador	RSV-02		
Título	Solicitud de permisos de Bluetooth		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario espera a que el Bluetooth este activado.		
Dependencias RU	RUV-01		

Tabla 42: Requisitos del Sistema RSV-02

Identificador	RSV-03		
Título	Solicitud de vínculo de Bluetooth.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario vinculará el Smartphone con el vehículo a través de la contraseña.		
Dependencias RU	RUV-02		

Tabla 43: Requisitos del Sistema RSV-03

Identificador	RSV-04		
Título	Conectar con el vehículo.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario espera a que se conecte con el dispositivo.		
Dependencias RU	RUV-02		

Tabla 44: Requisitos del Sistema RSV-04

Identificador	RSV-05		
Título	Conectar con éxito.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Le informa al usuario que se ha conectado al dispositivo con éxito.		
Dependencias RU	RUV-02		

Tabla 45: Requisitos del Sistema RSV-05

Identificador	RSV-06		
Título	Reintentar conectar.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	En el caso de que no se consiga conectar con el dispositivo indicarle al usuario de que si quiere volver a reintentar conectarse.		
Dependencias RU	RUV-02		

Tabla 46: Requisitos del Sistema RSV-06

Identificador	RSV-07		
Título	Control por voz.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario tendrá que controlar el vehículo mediante comandos de voz. Para ello tendrá un botón el cual será “No” cuando no esté recibiendo instrucciones y “Si” cuando esté listo para recibir instrucciones. Al púlsalo aparecerá una barra de intensidad del tono que está recogiendo, así conseguirá darle al usuario una sensación de grabación de audio.		
Dependencias RU	RUV-03		

Tabla 47: Requisitos del Sistema RSV-07

Identificador	RSV-08		
Título	Funcionamiento de la Aplicación		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	La aplicación tiene que funcionar y visualizarse correctamente para versiones de Android superiores e iguales a Android Gingerbread 2.3.5 [17].		
Dependencias RU	RUV-04		

Tabla 48: Requisitos del Sistema RSV-08

Identificador	RUV-09		
Título	Idioma		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El idioma de todos los textos de la aplicación será español, utilizando en ocasiones el idioma por defecto del Smartphone.		
Dependencias RU	RUV-05		

Tabla 49: Requisitos del Sistema RSV-09

Identificador	RUV-10		
Título	Diseño de la interfaz		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El diseño de la interfaz de usuario tiene que ser fácil e intuitivo por lo que un usuario inexperto debe poder usar la aplicación sin problemas.		
Dependencias RU	RUV-06		

Tabla 50: Requisitos del Sistema RSV-10

Identificador	RUV-11		
Título	Tiempo de conexión		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El tiempo de conexión al vehículo no debe superar los 30 segundos.		
Dependencias RU	RUV-07		

Tabla 51: Requisitos del Sistema RSV-11

Identificador	RUV-12		
Título	Salida de la aplicación		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario de la aplicación no podrá salir de la aplicación de manera inesperada, solo podrá salir por los mensajes que se le muestran por pantalla.		
Dependencias RU	RUV-08		

Tabla 52: Requisitos del Sistema RSV-12

Identificador	RSV-13		
Título	Lenguaje de programación		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El lenguaje de programación será JAVA.		
Dependencias RU	-		

Tabla 53: Requisitos del Sistema RSV-13

3.3.2 Requisitos para el control manual

En este apartado se especificaran los requisitos para el control Manual.

Identificador	RSM-01		
Título	Activación del Bluetooth		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario podrá activar el Bluetooth desde la aplicación. Se le mostrar al usuario un mensaje en el que se le indique que el Bluetooth está desactivado y que tiene que activarlo.		
Dependencias RU	RUM-01		

Tabla 54: Requisitos del Sistema RSM-01

Identificador	RSM-02		
Título	Solicitud de permisos de Bluetooth		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario espera a que el Bluetooth este activado.		
Dependencias RU	RUM-01		

Tabla 55: Requisitos del Sistema RSM-02

Identificador	RSM-03		
Título	Solicitud de vínculo de Bluetooth.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario vinculará el Smartphone con el vehículo a través de la contraseña.		
Dependencias RU	RUM-02		

Tabla 56: Requisitos del Sistema RSM-03

Identificador	RSM-04		
Título	Conectar con el vehículo.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario espera a que se conecte con el dispositivo.		
Dependencias RU	RUM-02		

Tabla 57: Requisitos del Sistema RSM-04

Identificador	RSM-05		
Título	Conectar con éxito.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	Le informa al usuario que se ha conectado al dispositivo con éxito.		
Dependencias RU	RUM-02		

Tabla 58: Requisitos del Sistema RSM-05

Identificador	RSM-06		
Título	Reintentar conectar.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	En el caso de que no se consiga conectar con el dispositivo indicarle al usuario de que si quiere volver a reintentar conectarse.		
Dependencias RU	RUM-02		

Tabla 59: Requisitos del Sistema RSM-06

Identificador	RSM-07		
Título	Control Manual.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario tendrá que controlar el vehículo mediante dos joysticks aprovechando la pantalla táctil del Smartphone, el joystick derecho controla el movimiento (delante-atrás) mientras el joystick derecho controla la dirección (izquierda-derecha).		
Dependencias RU	RUM-03		

Tabla 60: Requisitos del Sistema RSM-07

Identificador	RSM-08		
Título	Funcionamiento de la Aplicación		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	La aplicación tiene que funcionar y visualizarse correctamente para versiones de Android superiores e iguales a Android Gingerbread 2.3.5 [17].		
Dependencias RU	RUM-04		

Tabla 61: Requisitos del Sistema RSM-08

Identificador	RUM-09		
Título	Idioma		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El idioma de todos los textos de la aplicación será español, utilizando en ocasiones el idioma por defecto del Smartphone.		
Dependencias RU	RUM-05		

Tabla 62: Requisitos del Sistema RSM-09

Identificador	RUM-10		
Título	Diseño de la interfaz		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El diseño de la interfaz de usuario tiene que ser fácil e intuitivo por lo que un usuario inexperto debe poder usar la aplicación sin problemas.		
Dependencias RU	RUM-06		

Tabla 63: Requisitos del Sistema RSM-10

Identificador	RUM-11		
Título	Tiempo de conexión		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El tiempo de conexión al vehículo no debe superar los 30 segundos.		
Dependencias RU	RUM-07		

Tabla 64: Requisitos del Sistema RSM-11

Identificador	RUM-12		
Título	Salida de la aplicación		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario de la aplicación no podrá salir de la aplicación de manera inesperada, solo podrá salir por los mensajes que se le muestran por pantalla.		
Dependencias RU	RUM-08		

Tabla 65: Requisitos del Sistema RSM-12

Identificador	RSM-13		
Título	Lenguaje de programación		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El lenguaje de programación será JAVA.		
Dependencias RU	-		

Tabla 66: Requisitos del Sistema RSM-13

3.3.3 Requisitos de la aplicación para Arduino

En este apartado se especificaran los requisitos de la aplicación que será cargada en la placa Arduino.

Identificador	RSA-01		
Título	Mostrar en Pantalla.		
Necesidad	1	**Prioridad**	2
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario podrá ver por pantalla el modo en el que se encuentra el vehículo.		
Dependencias RU	RUA-01		

Tabla 67: Requisitos del Sistema RSA-01

Identificador	RSA-02		
Título	Sistema Anti-choque.		
Necesidad	2	**Prioridad**	2
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario dispondrá de un sistema anti-choque para evitar que el vehículo colisione.		
Dependencias RU	RUA-02		

Tabla 68: Requisitos del Sistema RSA-02

Identificador	RSA-03		
Título	Recibir instrucciones.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario podrá enviar instrucciones al vehículo desde el Smartphone y este tendrá que recibirlas para ejecutarlas.		
Dependencias RU	RUA-03		

Tabla 69: Requisitos del Sistema RSA-03

Identificador	RSA-04		
Título	Cambio de Modo.		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El usuario podrá cambiar de aplicación y esto tendrá que repercutir en un cambio de modo, ya que el sistema de control dependerá del modo en el que se encuentre el vehículo.		
Dependencias RU	RUA-03		

Tabla 70: Requisitos del Sistema RSA-04

Identificador	RSA-05		
Título	Tiempo Respuesta		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	El tiempo de respuesta para ejecutar las instrucciones tiene que ser inferior a 0.5 segundos.		
Dependencias RU	RUA-04		

Tabla 71: Requisitos del Sistema RSA-05

Identificador	RSA-06		
Título	Uso de la batería		
Necesidad	3	**Prioridad**	3
Estabilidad	Toda la vida del sistema	**Verificabilidad**	3
Descripción	La batería debe de durar al menos 5 minutos.		
Dependencias RU	RUA-05		

Tabla 72: Requisitos del Sistema RSA-06

3.4 Tabla de trazabilidad de requisitos de usuario /sistema

Para realizar el análisis y la validación de los requisitos de software expuestos se comprueba la relación entre requisitos de usuario y requisitos software:

3.4.1 Tabla de trazabilidad para el control por voz

En este apartado se especifica la matriz de trazabilidad el control mediante voz.

	REQUISITOS DE USUARIO							
REQUISITOS DE SISTEMA	RUV-01	RUV-02	RUV-03	RUV-04	RUV-05	RUV-06	RUV-07	RUV-08
RSV-01								
RSV-02								
RSV-03								
RSV-04								
RSV-05								
RSV-06								
RSV-07								
RSV-08								
RSV-09								
RSV-10								
RSV-11								
RSV-12								
RSV-13								

Tabla 73: Trazabilidad 1

3.4.2 Tabla de trazabilidad para el control manual

En este apartado se especifica la matriz de trazabilidad para el control manual.

	REQUISITOS DE USUARIO							
REQUISITOS DE SISTEMA	RUM-01	RUM-02	RUM-03	RUM-04	RUM-05	RUM-06	RUM-07	RUM-08
RSM-01								
RSM-02								
RSM-03								
RSM-04								
RSM-05								
RSM-06								
RSM-07								
RSM-08								
RSM-09								
RSM-10								
RSM-11								
RSM-12								
RSM-13								

Tabla 74: Trazabilidad 2

3.4.3 Tabla de trazabilidad para la aplicación para Arduino

En este apartado se especifica la matriz de trazabilidad de la aplicación que será cargada en la placa Arduino.

	REQUISITOS DE USUARIO				
REQUISITOS DE SISTEMA	RUA-01	RUA-02	RUA-03	RUA-04	RUA-05
RSA-01					
RSA-02					
RSA-03					
RSA-04					
RSA-05					
RSA-06					

Tabla 75: Trazabilidad 3

3.4.4 Matriz de trazabilidad de todos los requisitos

En este apartado se especifica la matriz de trazabilidad para todos los requisitos.

		RUV-01	RUV-02	RUV-03	RUV-04	RUV-05	RUV-06	RUV-07	RUV-08	RUM-01	RUM-02	RUM-03	RUM-04	RUM-05	RUM-06	RUM-07	RUM-08	RUA-01	RUA-02	RUA-03	RUA-04	RUA-05
REQUISITOS DE SISTEMA	RSV-01	■																				
	RSV-02	■																				
	RSV-03		■																			
	RSV-04		■																			
	RSV-05		■																			
	RSV-06		■																			
	RSV-07			■																		
	RSV-08				■																	
	RSV-09					■																
	RSV-10						■															
	RSV-11							■														
	RSV-12								■													
	RSV-13																					
	RSM-01									■												
	RSM-02									■												
	RSM-03										■											
	RSM-04										■											
	RSM-05										■											
	RSM-06										■											
	RSM-07											■										
	RSM-08												■									
	RSM-09													■								
	RSM-10														■							
	RSM-11															■						
	RSM-12																■					
	RSM-13																					
	RSA-01																	■				
	RSA-02																		■			
	RSA-03																			■		
	RSA-04																			■		
	RSA-05																				■	
	RSA-06																					■

Tabla 76: Tabla de trazabilidad

Como se puede apreciar todos los requisitos de usuario están cubiertos por uno o más requisitos del sistema, por lo que significa que todos requisitos quedarán implementados al final del proyecto.

4 Diseño e implementación de un vehículo a escala controlado por voz

En este apartado se mostrar los pasos seguidos para el diseño e implementación del vehículo a escala controlado por voz.

Antes de comenzar se realiza la instalación de los entornos de programación tanto para Android como para Arduino para lo cual seguí los pasos que están en Anexo 0: Instalación del entorno.

4.1 Diseño e implementación del hardware

En este apartado se describirán los pasos seguidos para para diseñar e implementar el hardware.

4.1.1 Diseño del hardware

Para el diseño del hardware se utilízala la herramienta software Fritzing, para ello accederemos a su web oficial [20], lo descargamos e instalamos.

El primer paso es diseñar un puente H, para controlar que los motores giran en ambos sentidos, para ello utilizaremos el chip L293B del cual tras consultar su manual (Anexo I: Manual del Chip L293B) se determinó que las conexiones serían las siguientes:

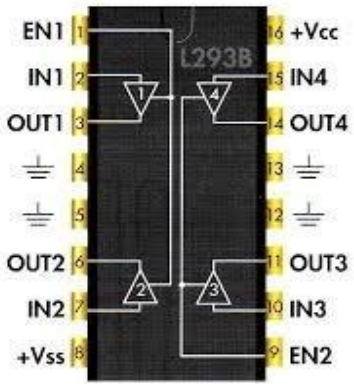


Ilustración 24: Diseño del hardware L293B

EN1: Habilitar y deshabilitar el motor 1, será conectada al pin 32 de la placa arduino mega.

IN1 e IN2: Controlara el sentido del motor 1, y se conectarán al pin 22 y 23 de la placa arduino mega.

OUT1 y OUT2: Serán las conexiones del motor1.

+Vss: Conexión del polo positivo de la batería que alimenta a los motores.

+Vcc: Conexión de alimentación lógica, en este caso será de 5 V de la placa arduino mega.

IN4 e IN3: Controlara el sentido del motor 2, y se conectarán al pin 24 y 25 de la placa arduino mega

OUT3 y OUT4: Serán las conexiones del motor2.

EN2: Habilita y deshabilita el motor 2, será conectada al pin 33 de la placa arduino mega.

Toma tierra: Conexión a GND de la placa arduino mega y al polo negativo de la barrería.

Tras conocer el funcionamiento del chipL293B y de cómo se conectara para su correcto funcionamiento se pasa al diseño del puente H, quedando de la siguiente manera:

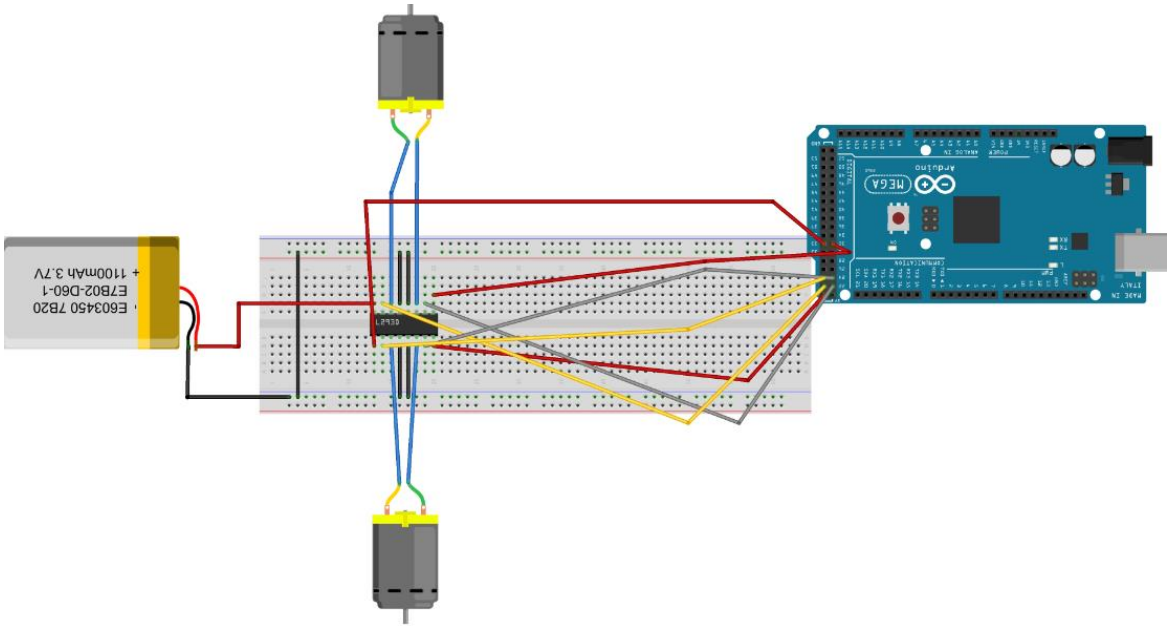


Ilustración 25: Diseño del hardware puente H

Una vez terminamos el puente H se procederá al diseño del sistema anti-choque, para ello se utiliza el módulo HC-SR04, para saber su funcionamiento consultaremos el manual (Anexo II: Manual del módulo ultrasónico HC-SR04) y se determinó que las conexiones serían las siguientes:

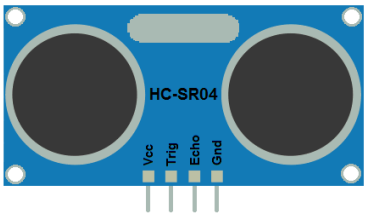


Ilustración 26: Diseño del hardware HC-SR04

Vcc: Conexión de alimentación lógica, en este caso será de 5 V de la placa arduino mega.

GND: Conexión a GND de la placa arduino mega.

Trig: Se le suministra un pulso de 10 µs, estará conectada al pin 44 de la placa arduino mega.

Echo: Espera el pulso de retorno, estará conectada al pin 45 de la placa arduino mega.

Tras realizar la conexión, tenemos que obtener la fórmula de la distancia, para ello en el manual se nos indica que la Distancia= (duración/2) (la duración se divide a la mitad por ser ida y vuelta), pero en nuestro caso al medirla en centímetros sabemos que Velocidad del sonido= 1/29 cm/µs por lo tanto la fórmula de la distancia será la siguiente:

- ✓ **Distancia= (duración/2)/29**

Tras conocer el funcionamiento del módulo ultrasónico HC- SR04 se pasa al diseño del circuito, el cual será el siguiente:

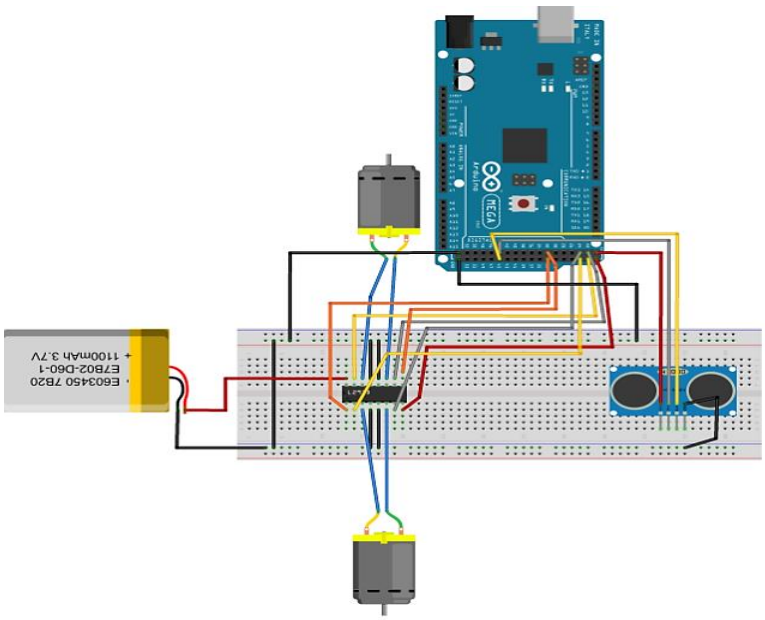


Ilustración 27: Diseño del hardware sistema anti-choque

Una vez diseñado el puente H y el sistema anti-choque se pasa a diseñar el sistema de comunicación utilizando la tecnología Bluetooth, para ello utilizaremos el módulo HC-06, el cual tras consultar su manual (Anexo III: Manual del módulo Bluetooth HC-06) se determinó que las conexiones serían las siguientes:

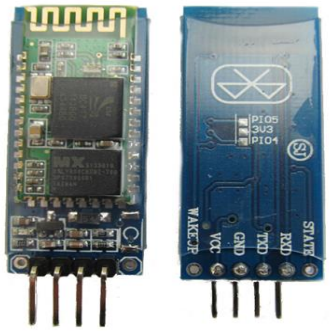


Ilustración 28: Diseño del hardware HC-06

Vcc: Alimentación del módulo entre 3,6V y 6V, en nuestro caso será 5V de la placa arduino mega.

GND: Conexión a GND de la palca arduino mega.

TXD: Transmisión de datos, se conectara al ping 18 de la palca arduino mega.

RXD: Recepción de datos, se conectara al ping 18 de la palca arduino mega.

Tras conocer el funcionamiento del módulo HC-06 se pasa al diseño del circuito, el cual será el siguiente:

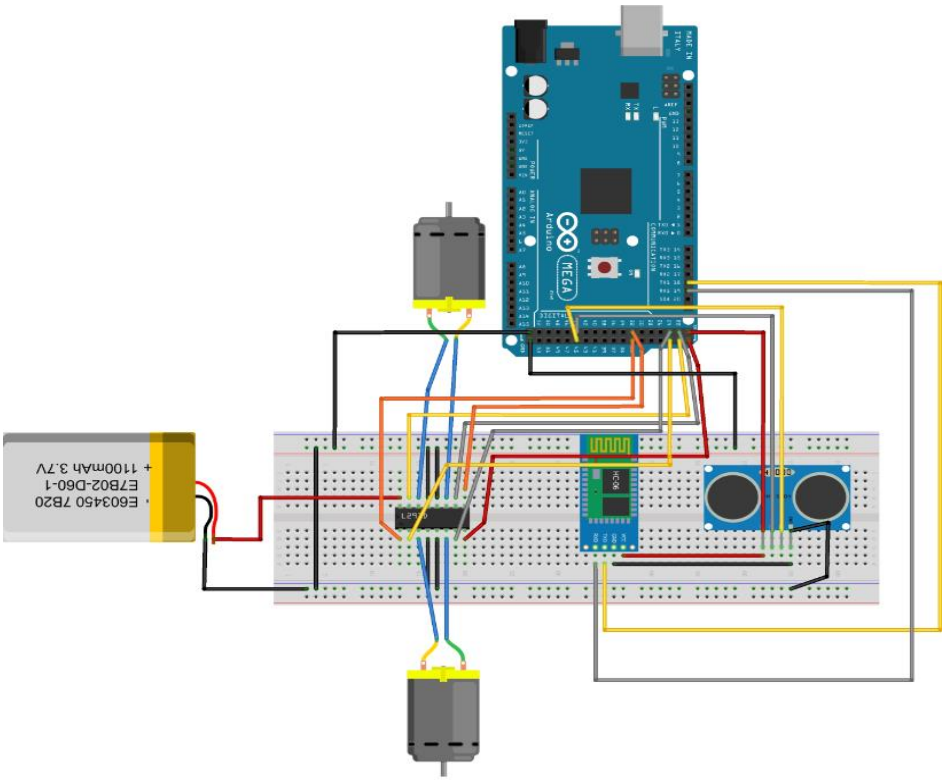
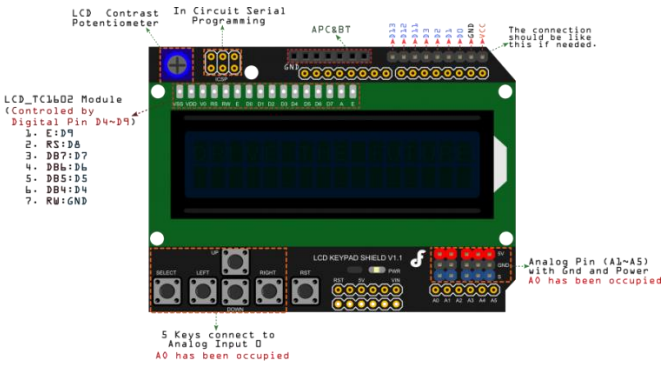


Ilustración 29: Diseño del hardware sistema de comunicación

Una vez tenemos todo el diseño básico del hardware lo completamos añadiendo una pantalla de información, para ello utilizaremos el módulo LCD Keypad Shield del cual se consultó su manual (Anexo IV: Manual del módulo LCD Keypad Shield) y se observó que simplemente había que colocarlo encima de la placa de arduino mega, la siguiente imagen nos da una mejor idea de su funcionamiento.



Instruction for D4 To D10 and Analog Pin 0

Pin	Function	Instruction
Digital 4(D4)	D4~D7 are used as DB4~DB7	Four high order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the LCD.
Digital 5(D5)		
Digital 6(D6)		
Digital 7(D7)		
Digital 8(D8)	RS	Choose Data or Signal Display
Digital 9(D9)	Enable	Starts data read/write
Digital 10(D10)	LCD Backlight Control	
Analog 0(A0)	Button select	Select, up, right, down and left

Ilustración 30: Diseño del hardware LCD Keypad Shield

Tras conocer el funcionamiento del módulo LCD Keypad Shield se pasara al diseño del circuito final, el cual será el siguiente:

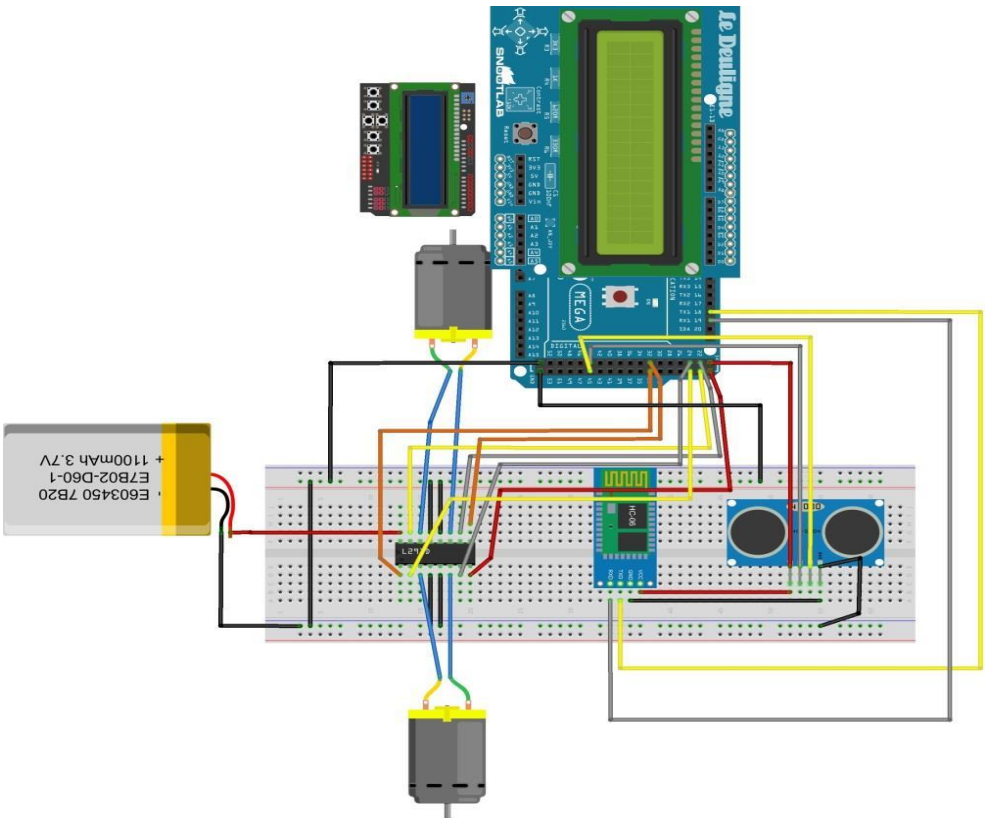


Ilustración 31: Diseño del hardware circuito final

Otra forma de ver el circuito sería de manera esquemática:

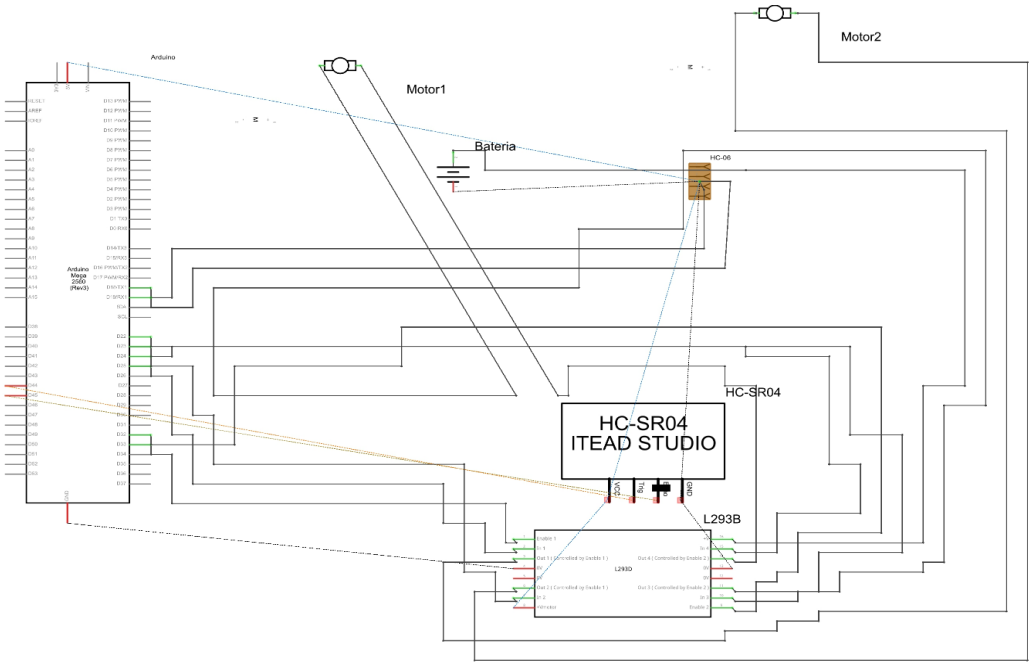


Ilustración 32: Diseño del hardware circuito final esquemático

4.1.2 Implementación del Hardware

Una vez realizado el diseño del circuito se realizara su implementación hardware, para ello lo primero que se realizó fue la compra de los componentes necesarios para el montaje y una vez los tenemos se procederá a su montaje. Lo primero que tenemos que realizar es desmontar el coche y quitarle el circuito de control que este traía pero dejando los motores quedando el vehículo en el chasis de la siguiente manera:

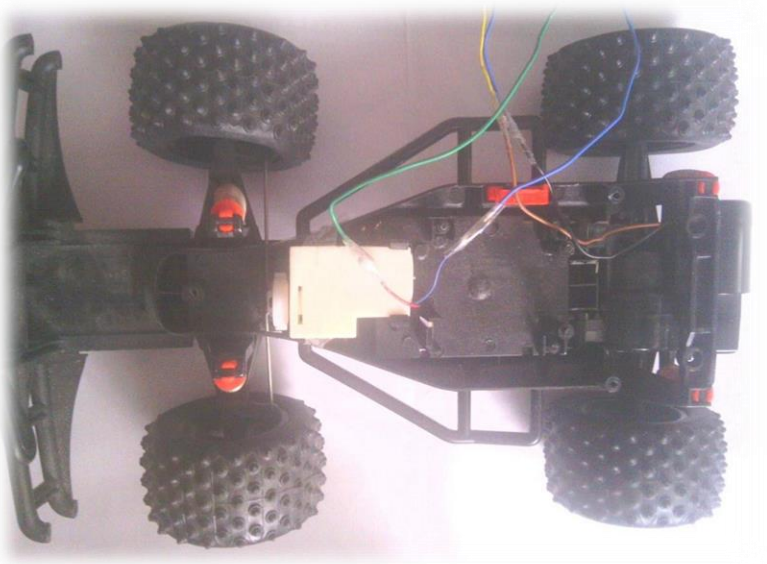


Ilustración 33: Vehículo en el chasis

Una vez tenemos el vehículo en el chasis, utilizaremos la placa de pruebas para montar el circuito según lo diseñado en el apartado anterior, quedando la implementación del hardware final de la siguiente manera (al ser necesario mucho cable para realizar las conexiones es difícil distinguir las conexiones en el circuito):

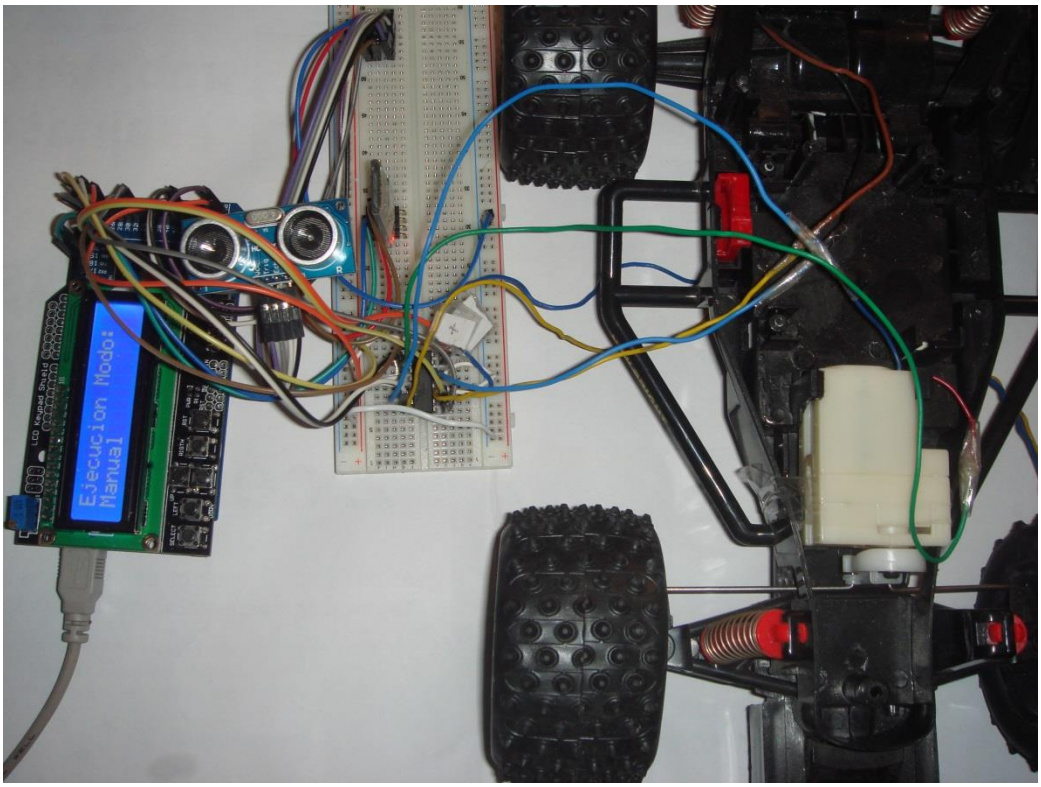


Ilustración 34: Implementación del Hardware final 1

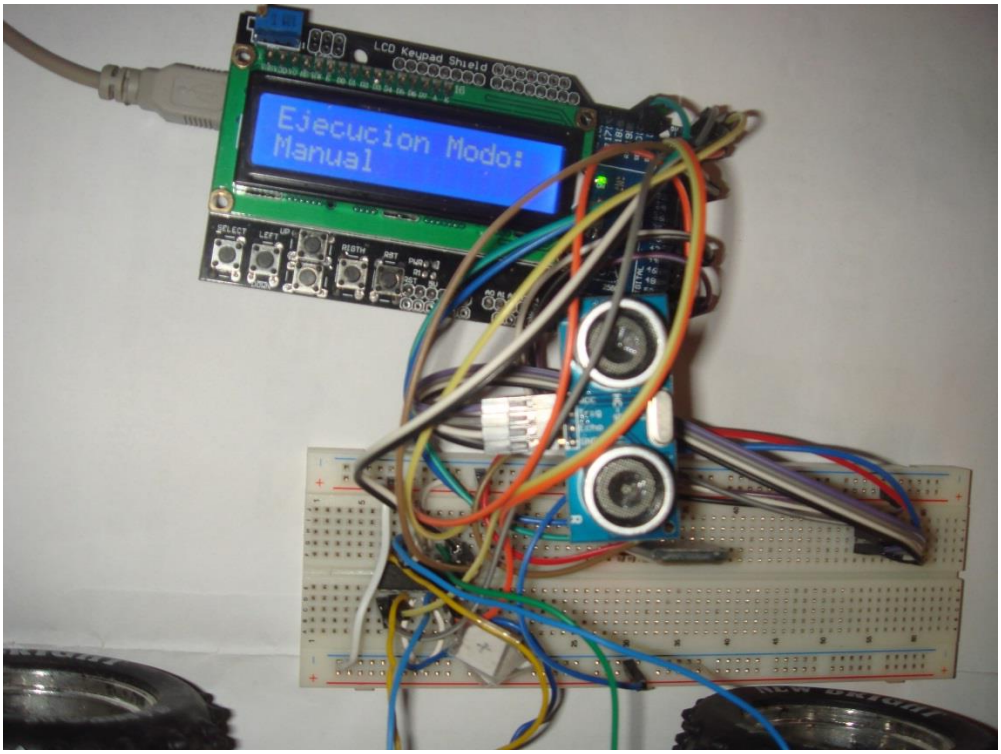


Ilustración 35: Implementación del Hardware final 2

4.2 Diseño e implementación del Software

Una vez tenemos diseñado e implementado el hardware se diseñó e implementó el software, siendo este la parte más difícil del proyecto. Empezaremos programando el Arduino, para más tarde programar en Android.

Antes de cualquier diseño se diseñará el sistema de comunicación entre el Smartphone y el vehículo. La comunicación será a través del envió de 1 carácter por parte del Smartphone para indicar las instrucciones a vehículo, a continuación se mostrarán los caracteres y su significado:

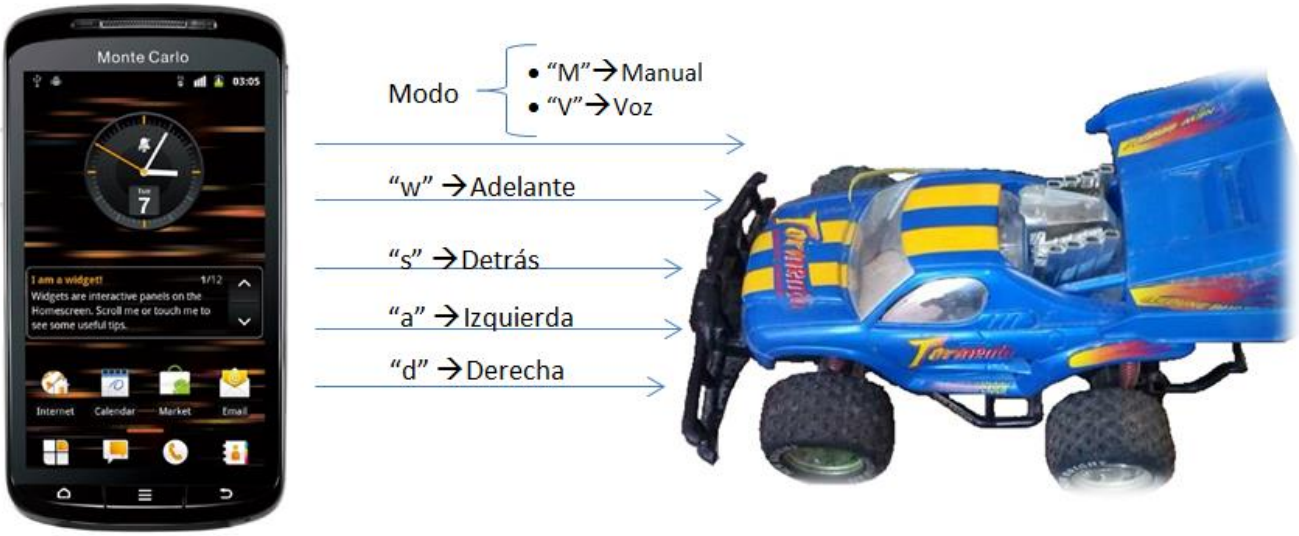


Ilustración 36: Sistema de Comunicación

4.2.1 Diseño e implementación del Software de Arduino

En este apartado se describe todo lo realizado para realizar el diseño e implementación del software para la placa arduino.

Antes de empezar a programar se testeo que los componentes funcionaban bien por separado y se realizó múltiples pruebas de funcionamiento, por lo que se le dedico bastante tiempo para entender bien el funcionamiento de los componentes y cómo programarlos correctamente.

Mientras se testean los componentes se descubrió que se podía cambiar la contraseña y el nombre del módulo Bluetooth HC-06, tras consultar la siguiente página web [15] en la cual se encuentra el tutorial.

Para ello se le cargó a la tarjeta Arduino el siguiente código:

cambiardatos

```
void setup()
{
  Serial1.begin(9600); //Velocidad del puerto del módulo Bluetooth
  Serial.begin(9600); //Abrimos la comunicación serie con el PC y establecemos velocidad
}

void loop()
{
  if(Serial1.available())
  {
    Serial.write(Serial1.read());
  }

  if(Serial.available())
  {
     Serial1.write(Serial.read());
  }
}
```

Ilustración 37: Código cambio de datos Bluetooth

Más tarde se abrió la consola para comunicarse con el arduino para ejecutar los siguientes comandos:

Los comandos AT que se pueden enviar en este módulo son los siguientes:

Comando AT	Descripción	Respuesta
AT	Test de comunicación.	Responde con un **OK**
AT+VERSION	Retorna la versión del Modulo	**OKlinvorV1.8**
AT+BAUDx	Configura la velocidad de trasmisión del modulo según el valor de "x"1 = 1200 bps 2 = 2400 bps 3 = 4800 bps 4 = 9600 bps (por defecto) 5 = 19200 bps 6 = 38400 bps 7 = 57600 bps 8 = 115200 bps 9 = 230400 bps A = 460800 bps B = 921600 bps C = 1382400 bps	AT+BAUD4 Configura la velocidad a 9600 baud rate Responde con **OK9600**
AT+NAMEx	Configura el nombre con el que se visualizara el modulo, soporta hasta 20 caracteres	AT+NAMEDIYMakers Configura el nombre del modulo a DIYMakers Responde con **OKsetname**
AT+PINxxxx	Configura el Pin de acceso al modulo (password).1234 por defecto.	AT+PIN1122 Configura el pin a 1122 Responde con **OKsetPIN**

Ilustración 38: Comandos AT

En nuestro caso solo ejecutaremos los siguientes comandos:

- AT: Para que nos responda con Ok y sepamos que está todo bien conectado.
- AT+NAMECOCHE R/C: Le cambiaremos el nombre a COCHE R/C.
- AT+PIN1992: Le cambiaremos la contraseña de conexión a 1992.

Antes de empezar a programar se diseñó la estructura principal del código mediante un diagrama de flujo:

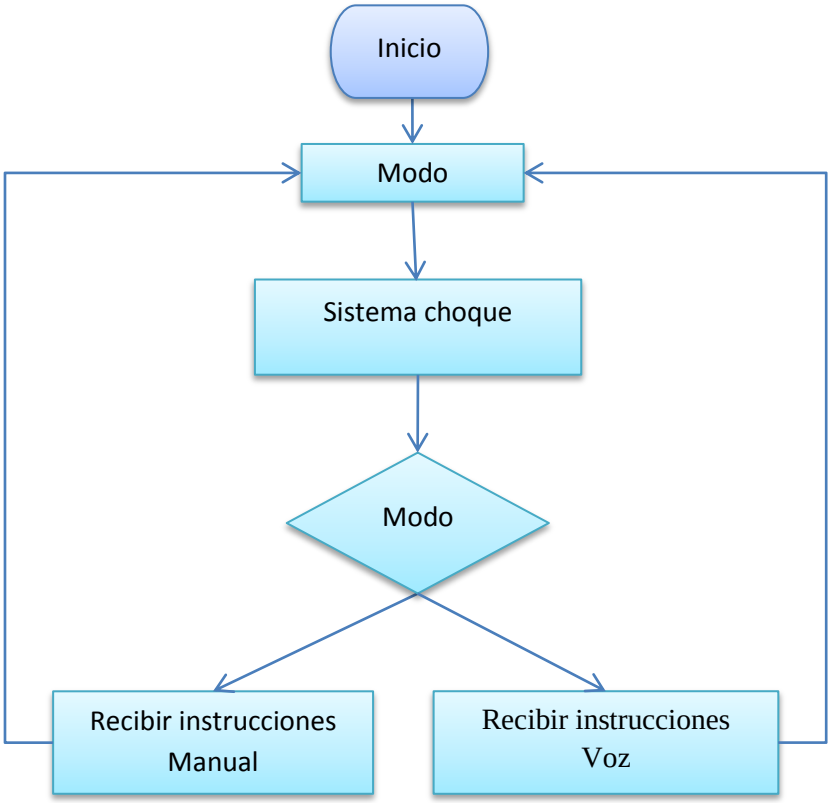


Ilustración 39: Diseño del diagrama de Flujo Arduino

En él se puede observar que hay dos modos uno manual y otro por control de voz. El modo se encargará de indicar si la aplicación que se está ejecutando es manual o mediante voz, ya que las instrucciones recibidas se ejecutarán de manera distinta dependiendo del modo.

Una vez sabemos el modo éste se indica por la pantalla LCD que tiene el vehículo, y se modifica la variable global para poder responder más tarde a la pregunta de en qué modo se encuentra para recibir las instrucciones y así poder responder ante ellas de manera satisfactoria.

4.2.2 Diseño e implementación del Software de Android

En este apartado se describirá todo lo realizado para realizar el diseño e implementación del software para el Smartphone con Android.

Para ello lo primero es el diseño de la estructura principal del código, para lo cual se diseña el siguiente diagrama de flujo:

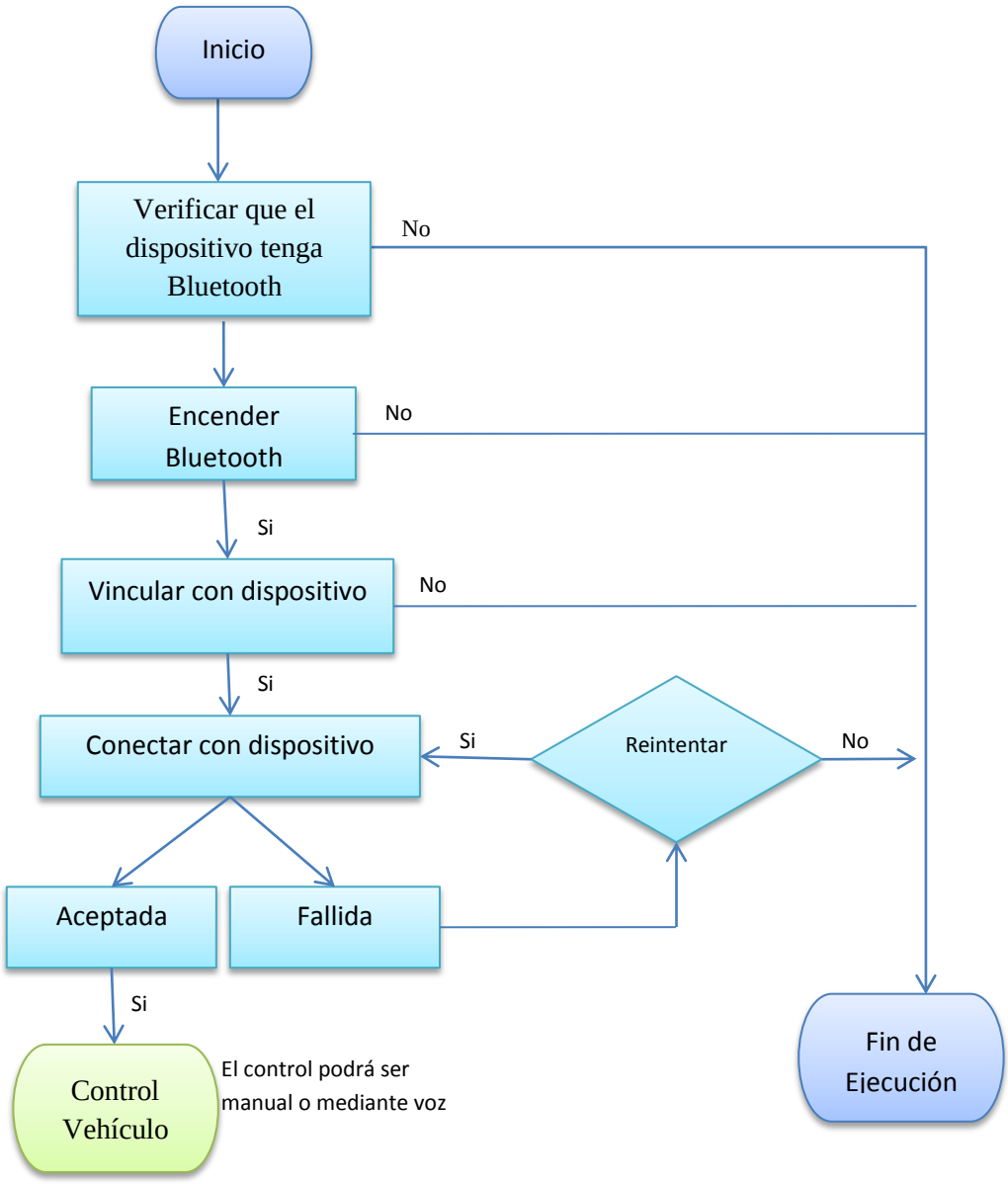


Ilustración 40: Diseño del diagrama de Flujo Android

Como se aprecia en el diagrama lo primero que se va a realizar y que tendrá en común tanto para el control de voz o el control manual será la conexión Bluetooth, para ello lo primero que será necesario encender el Bluetooth para lo cual nos aparecerá el siguiente mensaje:

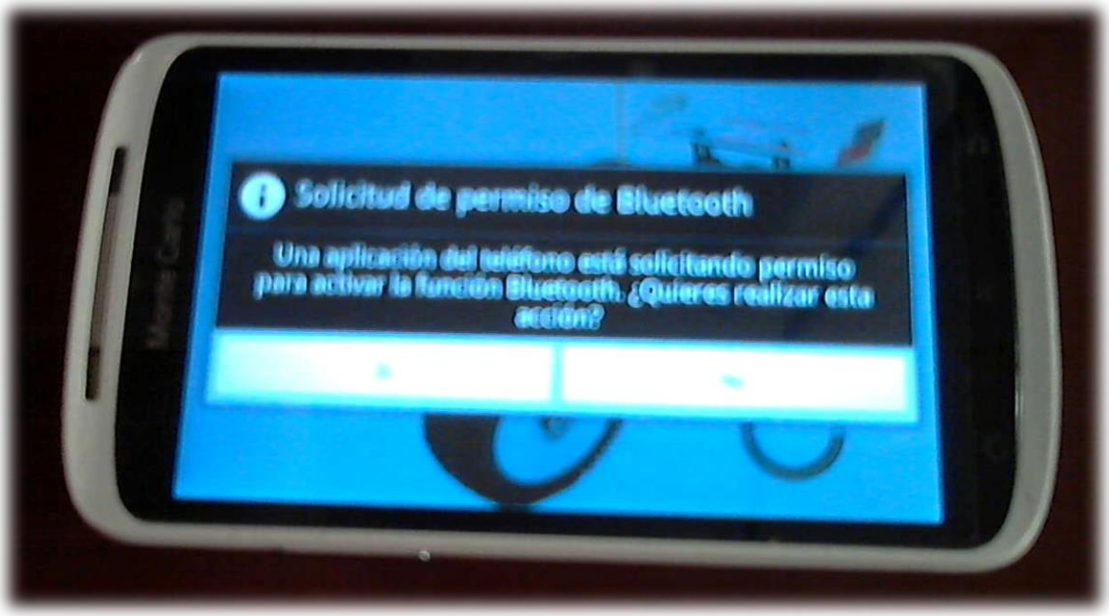


Ilustración 41: Aplicación activar Bluetooth

Al encender el Bluetooth nos saldrá más tarde el siguiente mensaje informándonos que el Bluetooth se está activando:

Ilustración 42: Aplicación activando Bluetooth

Para más tarde mostrar que se activó correctamente:

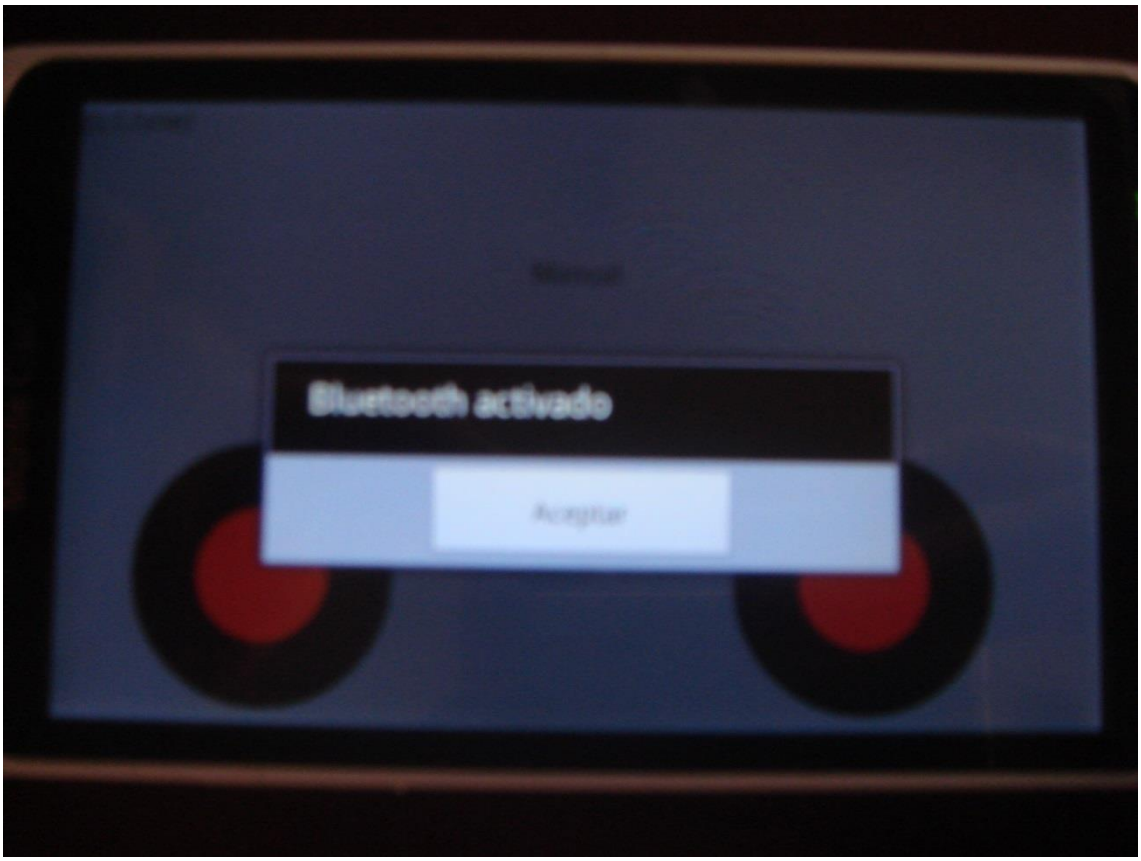


Ilustración 43: Aplicación Bluetooth activado

En el caso de que no lo desees activar aparecerá la siguiente pantalla y una vez aceptado se cerrara la aplicación.

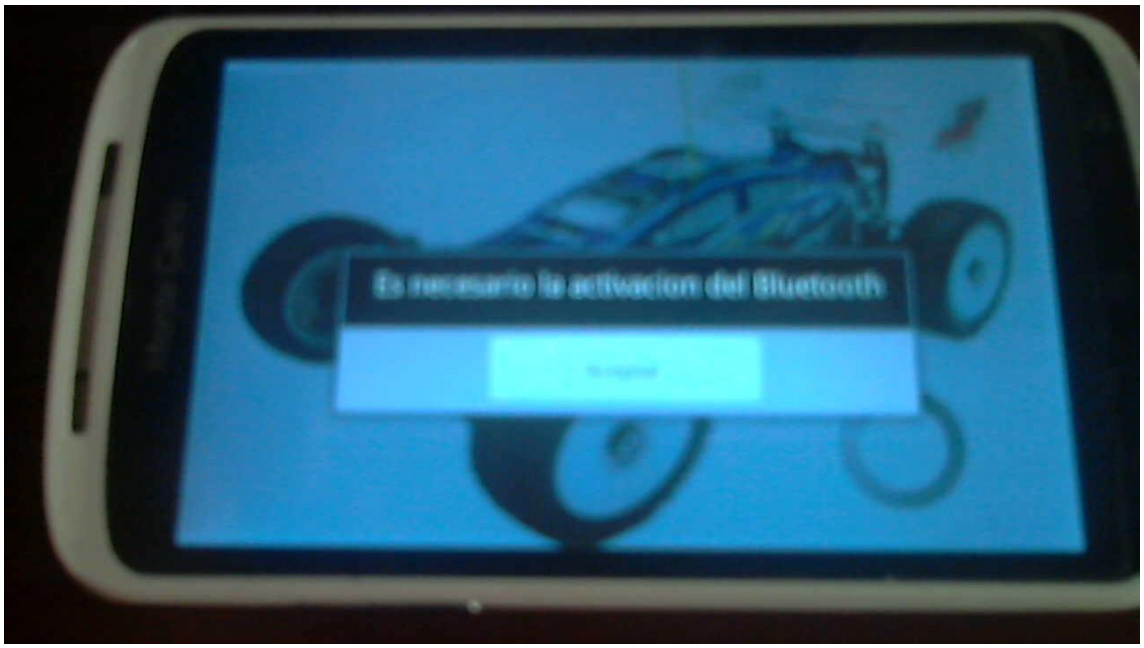


Ilustración 44: Aplicación no se activó el Bluetooth

Tras encender el Bluetooth se buscará el vehículo y una vez encontrado se pedirá que se vincule para poder conectarse a él, para realizar esta operación aparecerá la siguiente pantalla (PIN: 1992):

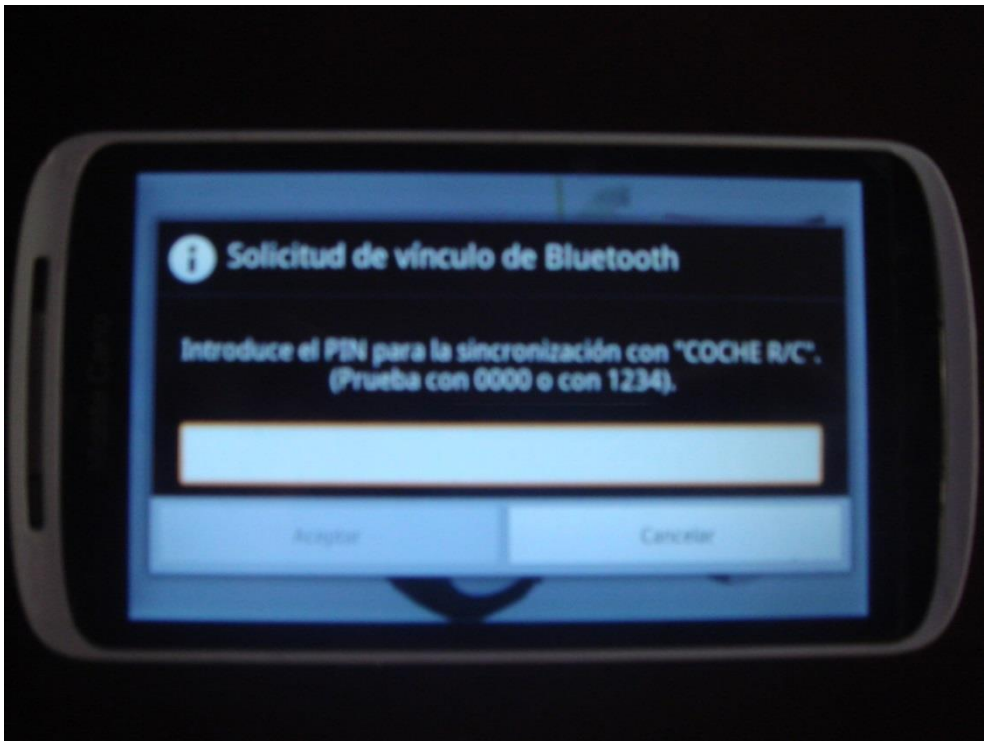


Ilustración 45: Aplicación vincular con el dispositivo

Una vez tengamos el vehículo vinculado tendremos que esperar a que se intente conectar, para ello aparecerá la siguiente pantalla:

Ilustración 46: Aplicación intentando conectar

Luego tendremos dos posibilidades:

- Que falle y entonces aparecerá el siguiente mensaje:

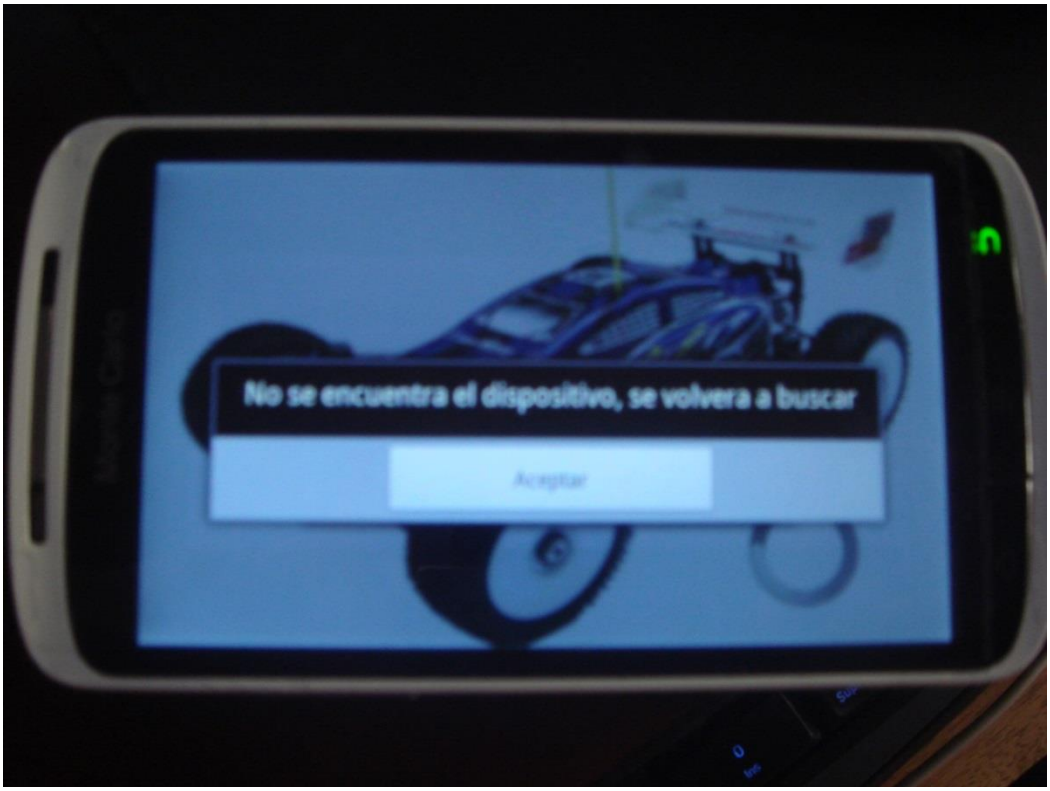


Ilustración 47: Aplicación error al conectar al dispositivo

- Que consiga conectarse entonces pasará a la pantalla de control del vehículo. En este caso tenemos dos posibilidades que sea de control manual o que sea de control mediante voz. Estas dos posibilidades se describirán detalladamente en los dos próximos puntos (4.2.2.1 y 4.2.2.2).

Para que esta interfaz funcione correctamente tendrá que tener una implementación software, la cual se describe lo principal a continuación:

Lo primero que se realiza es comprobar que el dispositivo tenga Bluetooth:

```
if(Adaptador == null){
	aviso_error( "Fallo Bluetooth, el dispositivo no tiene Bluetooth");
	return -1;
}
```

Tras ello se comprueba que el Bluetooth este encendido:

```
if(!Adaptador.isEnabled()){
Intent VentanaBluetooth = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
startActivityForResult(VentanaBluetooth, REQUEST_ENABLE_BT);
}
```

En el caso de que ya estuviese activado o tras activarlo se pasara a conectarse al dispositivo para ello será necesario la creación de socket para más tarde prepáralo para la conexión, para ello utilizamos la clase *BluetoothDevice* [49] y la clase *Socket* [12] y el resultado es el siguiente código:

```
String address = "20:14:08:27:02:82";
final UUID MY_UUID =  UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
BluetoothDevice device = Adaptador.getRemoteDevice(address);
socket.btSocket = device.createRfcommSocketToServiceRecord(MY_UUID);
socket.btSocket.connect();
socket.outStream = socket.btSocket.getOutputStream();
socket.outStream.flush();
```

En este caso utilizaremos *getRemoteDevice* para conectarse al dispositivo *HC-06* con la MAC: 20:14:08:27:02:82 y con createRfcommSocketToServiceRecord crearemos un socket de comunicación Bluetooth.

Lógicamente la clase dispone de más elementos para el funcionamiento pero ya se describieron los elementos fundamentales para conectarse mediante Bluetooth.

Para que lo anterior funcione correctamente se utilizaran los siguientes permisos:

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

4.2.2.1 Diseño e implementación del Software de Android Manual

En este apartado se describe el funcionamiento de la aplicación para el control del vehículo manualmente, para ello se diseña la siguiente interfaz de usuario:

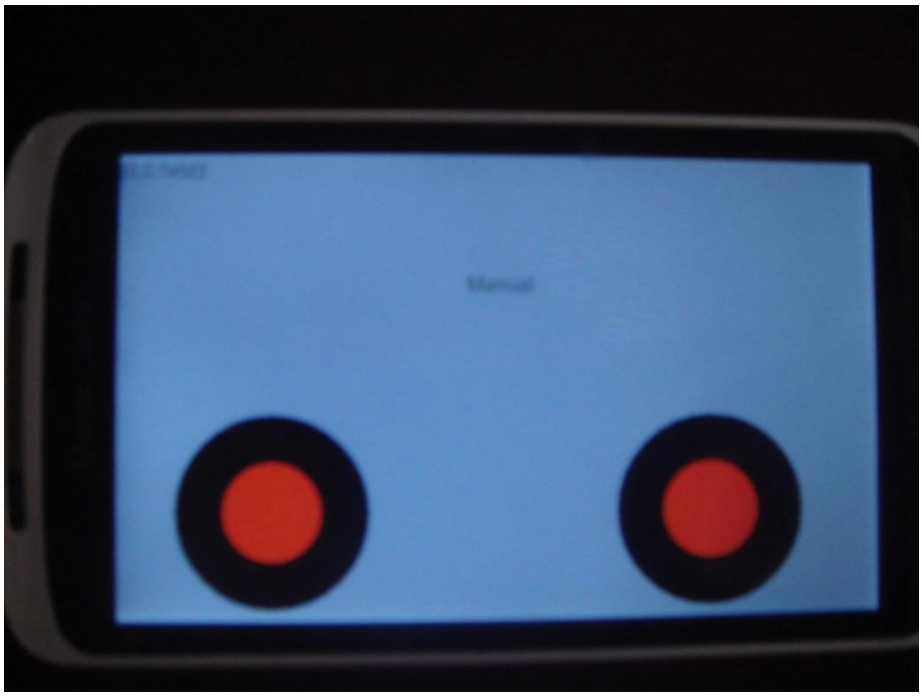


Ilustración 48: Aplicación interface Manual 1

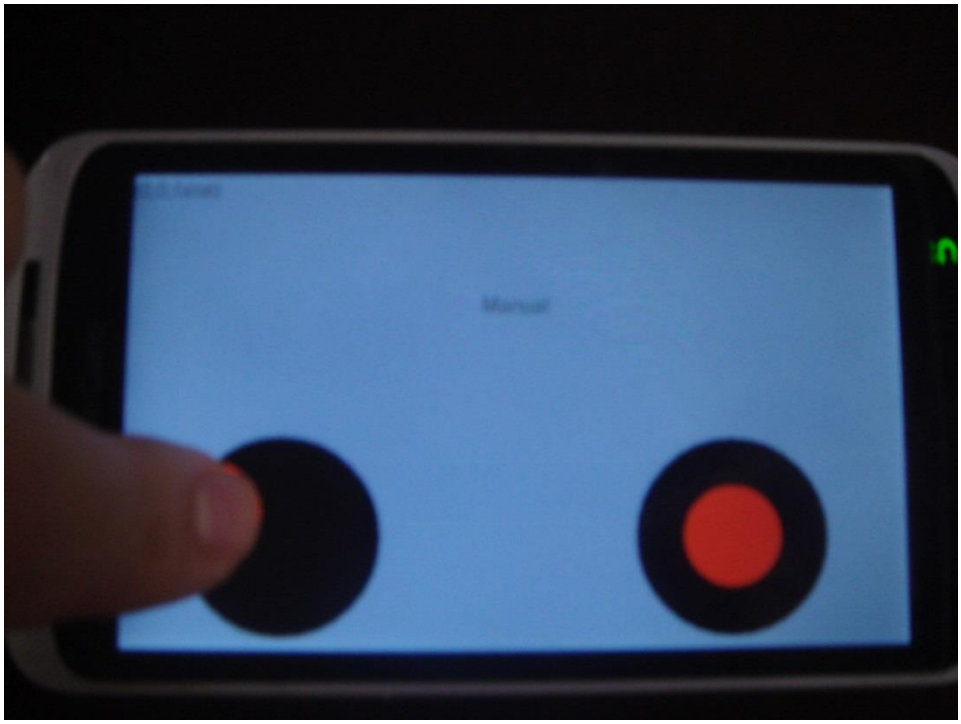


Ilustración 49: Aplicación interface Manual 2

Descripción del funcionamiento de la interface:

Como se puede apreciar la interface cuenta con dos joysticks, con los cuales el usuario podrá interactuar, con el joystick izquierdo controlaras que el vehículo se mueva hacia delante o hacia atrás, mientras que con el izquierdo se controlara la dirección del vehículo.

Descripción de la implementación software:

Para que esta interfaz funcione correctamente tendrá que tener una implementación software, la cual se describe lo principal a continuación:

En este caso tendremos un thread ya que la interfaz es construida en tiempo de ejecución mediante código. Para construir la interfaz en tiempo de ejecución la clase encargada será: *CrearJoystickBoton*, con una extensión de la clase de java *SurfaceView* [14] de la cual utilizaremos el siguiente método para la creación del hilo y su iniciación:

```
public void surfaceCreated(SurfaceHolder arg0) {
	thread = new MySurfaceThread(getHolder(), this);
	thread.setRunning(true);
	thread.start();
}
```

Tras la iniciación del hilo se utilizará el método *onDraw* para construir manualmente la interfaz, a continuación se muestra unas secciones de código que nos permiten hacernos una idea del funcionamiento general del método:

```
public void onDraw(Canvas canvas) {
//se realiza las configuraciones oportunas como por ejemplo pintar el fondo de
gris (los puntos suspensivos significa secciones de código omitidas)
…
canvas.drawColor(Color.GRAY);
…
//Declaración del círculo interno
Paint pcirculointerno = new Paint();
pcirculointerno.setColor(Color.RED);
pcirculointerno.setStyle(Paint.Style.FILL);
…
//dibujamos el circulo interno en la posición en la que este el dedo con radio de
//50
canvas.drawCircle(dedo1_x, dedo1_y, 50, pcirculointerno);
…
}
```

Para recoger la posición de los dedos se utilizó el método *onTouchEvent*, a continuación se muestra unas secciones de código que nos permiten hacernos una idea del funcionamiento general del método:

```
public boolean onTouchEvent(MotionEvent event) {
…
dedo1_x = (int) event.getX(0);
dedo1_y = (int) event.getY(0);
… }
```

Para terminar correctamente la ejecución del hilo se utiliza el siguiente método:

```
public void surfaceDestroyed(SurfaceHolder arg0) {
	boolean retry = true;
	thread.setRunning(false);
	while (retry) {
		try {
			thread.join();
			retry = false;
		} catch (InterruptedException e) {
		}
	}
}
```

4.2.2.2 Diseño e implementación del Software de Android Voz

En este apartado se describe el funcionamiento de la aplicación para el control del vehículo mediante voz, para ello se diseña la siguiente interfaz de usuario:

Ilustración 50: Aplicación interface Voz 1

Ilustración 51: Aplicación interface Voz 2

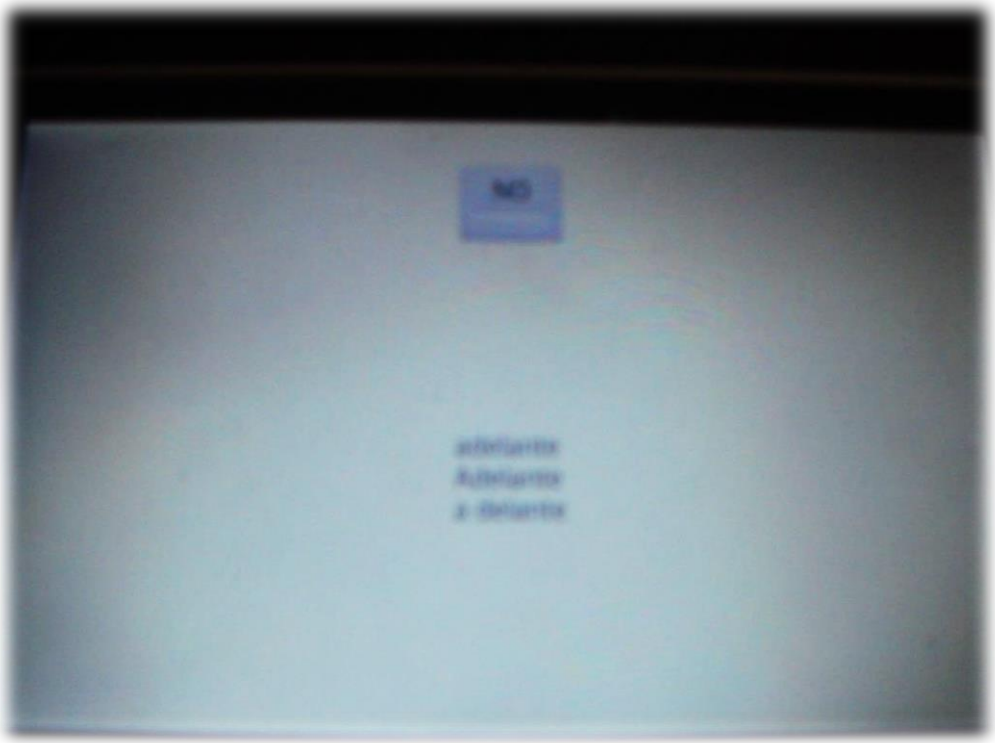


Ilustración 52: Aplicación interface Voz 3

Descripción del funcionamiento de la interface:

Como se puede apreciar la interfaz para el control mediante voz solo cuenta con un botón, el cual el usuario pulsara para dar instrucciones. Una vez pulsado se pasará a la Ilustración 51: Aplicación interface Voz 2, en la cual aparecerá una barra de progresión que indica el nivel de la voz que se está recogiendo. Una vez se recoja la voz esta pasará a ser traducida a texto y se mostrará en pantalla las posibles traducciones (Ilustración 52: Aplicación interface Voz 3) y se le mandará al vehículo la instrucción que sea reconocida.

Descripción de la implementación software:

Para la implementación software se consultó la siguiente página web [25] de la cual se extrajo parte del código y sobre todo la estructura fundamental.

Para la implementación del software se realiza todo en la clase *VoiceRecognitionActivity* en la cual se utilizaran distintas clases de *android.speech* [9] en este caso tiene una implementación de *RecognitionListener* [11] para utilizar su método publico *onResults* en el cual se recogerán los resultados devueltos por el servidor de Google para el reconocimiento de voz.

También se utilizará la clase *SpeechRecognizer* [13], la cual proporcionara acceso al servicio de reconocimiento de voz, para que esta clase funcione se necesita conexión a internet ya que es probable que el audio sea transmitido a servidores remotos para llevar a cabo el reconocimiento de voz.

Para el comenzar la captura de audio se utilizará un *intent* en este caso *RecognizerIntent* [10], para indicarle al dispositivo móvil que tiene que realizar una operación de intentar recoger audio.

Una vez descrito las clases utilizadas y para que se utilizan se pasará a indicar parte del código y el funcionamiento que este tiene hacerse una idea de cómo se recoge el audio y como este se pasa a texto:

```
//declaración de variables globales speech y recognizerIntent

private SpeechRecognizer speech = null;
private Intent recognizerIntent;
```

Como se puede apreciar se configuran las variables *speech* y *recognizerIntent* para el reconocimiento de voz y se le aplican parámetros como la acción al *intent*, el lenguaje de traducción, el número máximo de resultados, etc.

```
//preparación de las variables para el reconocimiento de voz
speech = SpeechRecognizer.createSpeechRecognizer(this);
speech.setRecognitionListener(this);
recognizerIntent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
recognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_PREFERENCE,"en");
recognizerIntent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
this.getPackageName());
recognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_WEB_SEARCH);
recognizerIntent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 3);
```

Tras tener configurado las variables, se pasará a darle una acción al botón y se utilizará *startListening* y *stopListening* para empezar y terminar la grabación.

```
Boton.setOnCheckedChangeListener(new OnCheckedChangeListener() {
public void onCheckedChanged(CompoundButton buttonView,boolean isChecked) {
	if (isChecked) {
		BarraProgresion.setVisibility(View.VISIBLE);
		BarraProgresion.setIndeterminate(true);
		speech.startListening(recognizerIntent);
	} else {
		BarraProgresion.setIndeterminate(false);
		BarraProgresion.setVisibility(View.INVISIBLE);
		speech.stopListening();
	}
}
});
```

Una vez tengamos los resultados se pasara a reconocerlos para enviarles las instrucciones al vehículo, en este caso el método solo reconoce la acción de adelante para hacerlo más entendible.

```
public void onResults(Bundle results) {
	ArrayList<String> matches =
	results.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
	String text = "";
	///Mandar los mensajes al coche
	for (String result : matches){
		text += result + "\n";
		if(result.equals("adelante")){
			try {
				socket.envio("w");
			} catch (IOException e) {
				e.printStackTrace();
			}
		}
	}
	Instrucciones.setText(text);
}
```

5 Pruebas y evolución

El presente apartado tiene como objetivo principal de comprobar que nuestro sistema cumple con las expectativas puestas en él y que funciona de la manera planteada y correctamente.

Por ello se va a elaborar un plan de pruebas que consiga verificar si el sistema cumple todos los requisitos y expectativas, siempre de la manera adecuada para el cliente.

En este documento realizaremos la selección de pruebas fundamentales para asegurar el correcto funcionamiento del sistema, teniendo que reestructurar su diseño y funcionalidades, si no cumple con alguna de estas pruebas.

Las pruebas serán lo más detalladas posibles para poder verificar de la forma más exhaustiva posible, el correcto funcionamiento, mediante las directrices correctas.

5.1 Definición de Requisitos del entorno de Pruebas

El Entorno de pruebas estará definido por varias configuraciones en función de las pruebas que se vayan a realizar y del momento de desarrollo en el que se encuentra la aplicación.

Para las pruebas se utilizará un Smartphone Orange MonteCarlo con las siguientes especificaciones del hardware:

- Pantalla táctil de 4.3 pulgadas con resolución de 800 × 480 píxeles.
- Procesador de 800 MHz.
- Cámara de 5 megapíxeles. Sin cámara frontal.
- Memoria RAM de 512 MB.
- Conectividad HSDPA y WIFI.
- Ranura para tarjetas de memoria microSD.
- Batería de 1420 mAh.

También se utilizara el prototipo del vehículo.

5.2 Pruebas de funcionalidad

Estas pruebas tienen como objetivo verificar la funcionalidad de distintos elementos del sistema y si estos se implementaron de manera correcta o no.

Para la definición de las pruebas funcionales usaremos la siguiente tabla:

Nombre			PF_XX
Aplicación	Voz/Manual/Arduino	Resultado	Valido/ Invalido
Descripción			
Acciones			
Requisito			

Tabla 77: Estructura de tabla de Pruebas Funcionales

Dónde:

Nombre: Identificara al nombre de la prueba.

Aplicación: Indicara a cuál de las 3 aplicaciones ira destinada la prueba.

ID: Es el identificador de la prueba.

Resultado: Es el resultado aportado tras realizar la prueba.

Descripción: Descripción del objetivo de la prueba.

Acciones: Pasos a seguir para realizar la prueba.

Requisito: Indica al requisito del sistema al que pertenece la prueba.

Prueba Funcional 1			PF_01
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que el usuario pueda activar el Bluetooth a través de la aplicación.		
Acciones	1. Tener desactivado el Bluetooth. 2. Abrir la aplicación y esperar a que aparezca una pantalla de activación del Bluetooth.		
Requisito	RSV-01		

Tabla 78: Prueba Funcional PF_01

Prueba Funcional 2			PF_02
Aplicación	Voz	Resultado	Valido
Descripción	La aplicación tendrá los permisos necesarios para activar el Bluetooth.		
Acciones	Intentar activar el Bluetooth y que este se active con éxito.		
Requisito	RSV-02		

Tabla 79: Prueba Funcional PF_02

Prueba Funcional 3			PF_03
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que el usuario pueda vincular el Smartphone al vehículo a través de la aplicación.		
Acciones	1. Tener activado el Bluetooth. 2. No tener vinculados los dispositivos. 3. Desde la aplicación vincular los dispositivos a través de la contraseña.		
Requisito	RSV-03		

Tabla 80: Prueba Funcional PF_03

Prueba Funcional 4			PF_04
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que los dos dispositivos se conectan correctamente.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito.		
Requisito	RSV-04		

Tabla 81: Prueba Funcional PF_04

Prueba Funcional 5			PF_05
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que al usuario se le informa que se conectó con el dispositivo con éxito.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito.		
Requisito	RSV-05		

Tabla 82: Prueba Funcional PF_05

Prueba Funcional 6			PF_06
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que al usuario se le permite intentar reconectar cuando el resultado de la conexión es erróneo.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo erróneamente.		
Requisito	RSV-06		

Tabla 83: Prueba Funcional PF_06

Prueba Funcional 7			PF_07
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que el usuario podrá controlar el coche con todos los comandos de voz.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito. 4. Ejecutar comando “delante”. 5. Ejecutar comando “atrás”. 6. Ejecutar comando “parado”. 7. Ejecutar comando “izquierda”. 8. Ejecutar comando “derecha”. 9. Ejecutar comando “recto”.		
Requisito	RSV-07		

Tabla 84: Prueba Funcional PF_07

Prueba Funcional 8			PF_08
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que la aplicación funcionar y se visualizarse correctamente para versiones de Android superiores e iguales a Android Gingerbread 2.3.5 [17].		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito. 4. Observar que la interfaz es correcta		
Requisito	RSV-08		

Tabla 85: Prueba Funcional PF_08

Prueba Funcional 9			PF_09
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que el idioma de todos los textos de la aplicación será español, utilizando en ocasiones el idioma por defecto del Smartphone.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito. 4. Observar que la interfaz es correcta		
Requisito	RSV-09		

Tabla 86: Prueba Funcional PF_09

Prueba Funcional 10			PF_10
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que el diseño de la interfaz de usuario tiene que ser fácil e intuitivo por lo que un usuario inexperto debe poder usar la aplicación sin problemas.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito. 4. Observar que la interfaz es correcta		
Requisito	RSV-10		

Tabla 87: Prueba Funcional PF_10

Prueba Funcional 11			PF_11
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que el tiempo de conexión al vehículo no supera los 30 segundos.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito.		
Requisito	RSV-11		

Tabla 88: Prueba Funcional PF_11

Prueba Funcional 12			PF_12
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que el usuario de la aplicación no podrá salir de la aplicación de manera inesperada, solo podrá salir por los mensajes que se le muestran por pantalla.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito.		
Requisito	RSV-12		

Tabla 89: Prueba Funcional PF_12

Prueba Funcional 13			PF_13
Aplicación	Voz	Resultado	Valido
Descripción	Comprobar que el lenguaje donde se ha programado la aplicación.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito. 4. Observar que la interfaz es correcta		
Requisito	RSV-13		

Tabla 90: Prueba Funcional PF_13

Prueba Funcional 14			PF_14
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que el usuario pueda activar el Bluetooth a través de la aplicación.		
Acciones	3. Tener desactivado el Bluetooth. 4. Abrir la aplicación y esperar a que aparezca una pantalla de activación del Bluetooth.		
Requisito	RSM-01		

Tabla 91: Prueba Funcional PF_14

Prueba Funcional 15			PF_15
Aplicación	Manual	Resultado	Valido
Descripción	La aplicación tendrá los permisos necesarios para activar el Bluetooth.		
Acciones	Intentar activar el Bluetooth y que este se active con éxito.		
Requisito	RSM-02		

Tabla 92: Prueba Funcional PF_15

Prueba Funcional 16			PF_16
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que el usuario pueda vincular el Smartphone al vehículo a través de la aplicación.		
Acciones	4. Tener activado el Bluetooth. 5. No tener vinculados los dispositivos. 6. Desde la aplicación vincular los dispositivos a través de la contraseña.		
Requisito	RSM-03		

Tabla 93: Prueba Funcional PF_16

Prueba Funcional 17			PF_17
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que los dos dispositivos se conectan correctamente.		
Acciones	4. Tener activado el Bluetooth. 5. Tener vinculados los dispositivos. 6. Conectarse con el dispositivo con éxito.		
Requisito	RSM-04		

Tabla 94: Prueba Funcional PF_17

Prueba Funcional 18			PF_18
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que al usuario se le informa que se conectó con el dispositivo con éxito.		
Acciones	4. Tener activado el Bluetooth. 5. Tener vinculados los dispositivos. 6. Conectarse con el dispositivo con éxito.		
Requisito	RSM-05		

Tabla 95: Prueba Funcional PF_18

Prueba Funcional 19			PF_19
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que al usuario se le permite intentar reconectar cuando el resultado de la conexión es erróneo.		
Acciones	4. Tener activado el Bluetooth. 5. Tener vinculados los dispositivos. 6. Conectarse con el dispositivo erróneamente.		
Requisito	RSM-06		

Tabla 96: Prueba Funcional PF_19

Prueba Funcional 20			PF_20
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que el usuario podrá controlar el coche con la pantalla táctil del Smartphone.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito. 4. Tocar joystick hacia delante. 5. Tocar joystick hacia atrás. 6. No tocar el joystick izquierdo (parado). 7. Tocar joystick hacia izquierda. 8. Tocar joystick hacia derecha. 9. No tocar el joystick derecho (recto).		
Requisito	RSM-07		

Tabla 97: Prueba Funcional PF_20

Prueba Funcional 21			PF_21
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que la aplicación funcionar y se visualizarse correctamente para versiones de Android superiores e iguales a Android Gingerbread 2.3.5 [17].		
Acciones	5. Tener activado el Bluetooth. 6. Tener vinculados los dispositivos. 7. Conectarse con el dispositivo con éxito. 8. Observar que la interfaz es correcta		
Requisito	RSM-08		

Tabla 98: Prueba Funcional PF_21

Prueba Funcional 22			PF_22
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que el idioma de todos los textos de la aplicación será español, utilizando en ocasiones el idioma por defecto del Smartphone.		
Acciones	5. Tener activado el Bluetooth. 6. Tener vinculados los dispositivos. 7. Conectarse con el dispositivo con éxito. 8. Observar que la interfaz es correcta		
Requisito	RSM-09		

Tabla 99: Prueba Funcional PF_22

Prueba Funcional 23			PF_23
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que el diseño de la interfaz de usuario tiene que ser fácil e intuitivo por lo que un usuario inexperto debe poder usar la aplicación sin problemas.		
Acciones	5. Tener activado el Bluetooth. 6. Tener vinculados los dispositivos. 7. Conectarse con el dispositivo con éxito. 8. Observar que la interfaz es correcta		
Requisito	RSM-10		

Tabla 100: Prueba Funcional PF_23

Prueba Funcional 24			PF_24
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que el tiempo de conexión al vehículo no supera los 30 segundos.		
Acciones	4. Tener activado el Bluetooth. 5. Tener vinculados los dispositivos. 6. Conectarse con el dispositivo con éxito.		
Requisito	RSM-11		

Tabla 101: Prueba Funcional PF_24

Prueba Funcional 25			PF_25
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que el usuario de la aplicación no podrá salir de la aplicación de manera inesperada, solo podrá salir por los mensajes que se le muestran por pantalla.		
Acciones	4. Tener activado el Bluetooth. 5. Tener vinculados los dispositivos. 6. Conectarse con el dispositivo con éxito.		
Requisito	RSM-12		

Tabla 102: Prueba Funcional PF_25

Prueba Funcional 26			PF_26
Aplicación	Manual	Resultado	Valido
Descripción	Comprobar que el lenguaje donde se ha programado la aplicación.		
Acciones	5. Tener activado el Bluetooth. 6. Tener vinculados los dispositivos. 7. Conectarse con el dispositivo con éxito. 8. Observar que la interfaz es correcta		
Requisito	RSM-13		

Tabla 103: Prueba Funcional PF_26

Prueba Funcional 27			PF_27
Aplicación	Arduino	Resultado	Valido
Descripción	El usuario podrá ver por pantalla el modo en el que se encuentra el vehículo y así comprobar que el vehículo cambia de modo.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito con las dos aplicaciones para observar que tiene un modo distinto para cada una.		
Requisito	RSA-01, RSA-04		

Tabla 104: Prueba Funcional PF_27

Prueba Funcional 28			PF_28
Aplicación	Arduino	Resultado	Valido
Descripción	Comprobar que el sistema Anti-choque funciona bien y evita que el sistema colisione. Así mismo comprobar que recibe y ejecuta instrucciones.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito 4. Controlar el vehículo e intentar chocarlo contra un obstáculo.		
Requisito	RSA-02, RSA-03		

Tabla 105: Prueba Funcional PF_28

Prueba Funcional 29			PF_29
Aplicación	Arduino	Resultado	Valido
Descripción	Comprobar que el tiempo de respuesta para ejecutar las instrucciones tiene que ser inferior a 0.5 segundos		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito 4. Controlar el vehículo.		
Requisito	RSA-05		

Tabla 106: Prueba Funcional PF_28

Prueba Funcional 30			PF_30
Aplicación	Arduino	Resultado	Valido
Descripción	Comprobar la batería debe de durar al menos 5 minutos.		
Acciones	1. Tener activado el Bluetooth. 2. Tener vinculados los dispositivos. 3. Conectarse con el dispositivo con éxito 4. Controlar el vehículo durante 5 minutos o más.		
Requisito	RSA-06		

Tabla 107: Prueba Funcional PF_29

5.3 Pruebas de estrés

En este caso no se realizan pruebas de estrés, ya que el vehículo soporta una sola conexión de una de las dos aplicaciones al mismo tiempo.

5.4 Matriz de trazabilidad de funcionalidad

En este apartado se mostrar la matriz de trazabilidad de funcionalidad la cual tiene el objetivo de comprobar que los requisitos del sistema han quedado implementados y probados.

Se podrá apreciar como todas las filas de la matriz tiene una marca, por lo que el requisito se ha probado por una o varias pruebas, así mismo existirán pruebas que comprueben varios requisitos como las pruebas “PF_15” y “PF_16”

REQUISITOS DE SISTEMA	PF_01	PF_02	PF_03	PF_04	PF_05	PF_06	PF_07	PF_08	PF_09	PF_10	PF_11	PF_12	PF_13	PF_14	PF_15	PF_16	PF_17	PF_18	PF_19	PF_20	PF_21	PF_22	PF_23	PF_24	PF_25	PF_26	PF_27	PF_28	PF_29	PF_30
RSV-01	■																													
RSV-02		■																												
RSV-03			■																											
RSV-04				■																										
RSV-05					■																									
RSV-06						■																								
RSV-07							■																							
RSV-08								■																						
RSV-09									■																					
RSV-10										■																				
RSV-11											■																			
RSV-12												■																		
RSV-13													■																	
RSM-01														■																
RSM-02															■															
RSM-03																■														
RSM-04																	■													
RSM-05																		■												
RSM-06																			■											
RSM-07																				■										
RSM-08																					■									
RSM-09																						■								
RSM-10																							■							
RSM-11																								■						
RSM-12																									■					
RSM-13																										■				
RSA-01																											■			
RSA-02																												■		
RSA-03																												■		
RSA-04																											■			
RSA-05																													■	
RSA-06																														■

Tabla 108: Matriz de trazabilidad de funcionalidad

6 Planificación y presupuesto

En este apartado se detalla la planificación que seguirá el proyecto durante sus diferentes fases, así como el presupuesto para la realización del mismo. Para definir la planificación se utilizará un diagrama de Gantt, que es un diagrama gráfico para mostrar el tiempo de dedicación previsto para diferentes tareas o actividades. El presupuesto mostrará todos los gastos establecidos para el proyecto, para ofrecer una mayor transparencia aparecerán todos los costes por separado para aportar claridad al coste final.

6.1 Planificación

En este apartado se detalla la planificación inicial que se establece para el proyecto, pero dicha planificación es posible que se vea alterada por diversos motivos. Primero antes de realizar el diagrama de Gantt se realizará un cronograma de las actividades que consistirá En una tabla donde veremos las distintas actividades a realizar en el proyecto, las fechas previstas de inicio y de fin para cada una de ellas, así como sus días de duración.

Actividad	Fecha inicio	fecha fin	Duración
Propuesta	09/01/2015	09/01/2015	1
Calculo de costes	12/01/2015	15/01/2015	5
Estudio Viabilidad	19/01/2015	30/01/2015	10
Análisis	02/02/2015	20/02/2015	15
Diseño	23/02/2015	01/04/2015	28
Implementación	06/04/2015	31/07/2015	85
Pruebas	03/08/2015	07/08/2015	5
Documentación	10/08/2015	25/09/2015	35

Tabla 109: Planificación de actividades

6.1.1 Diagrama de Gantt

En este apartado se mostrará la planificación de las actividades anteriores mediante el diagrama de Gantt, para ello se utiliza la herramienta *Gantt proyect* descargada de la siguiente web [51].

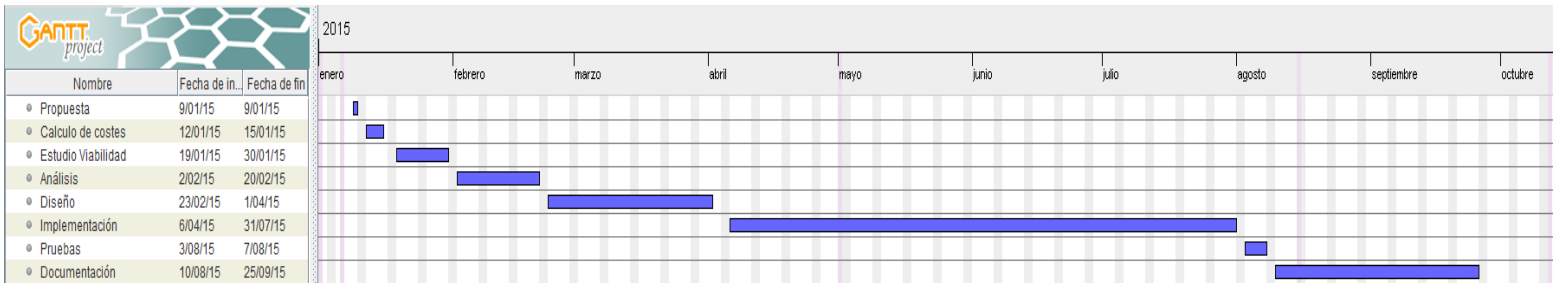


Ilustración 53: Diagrama de Gantt

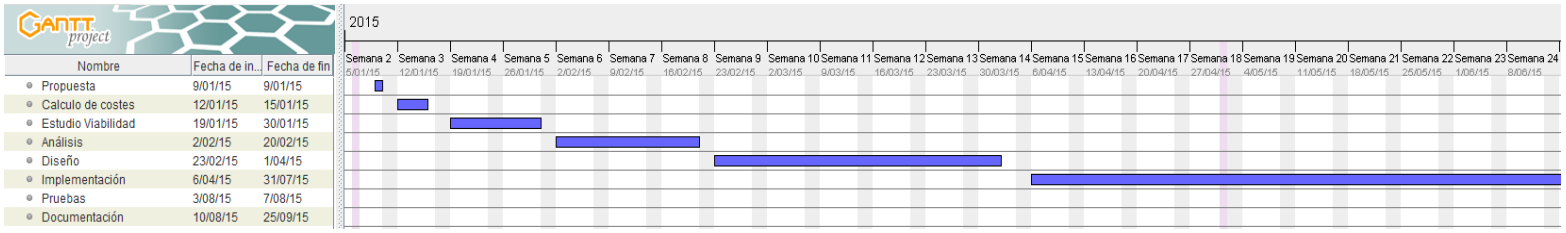


Ilustración 54: Diagrama de Gantt Semanas 1

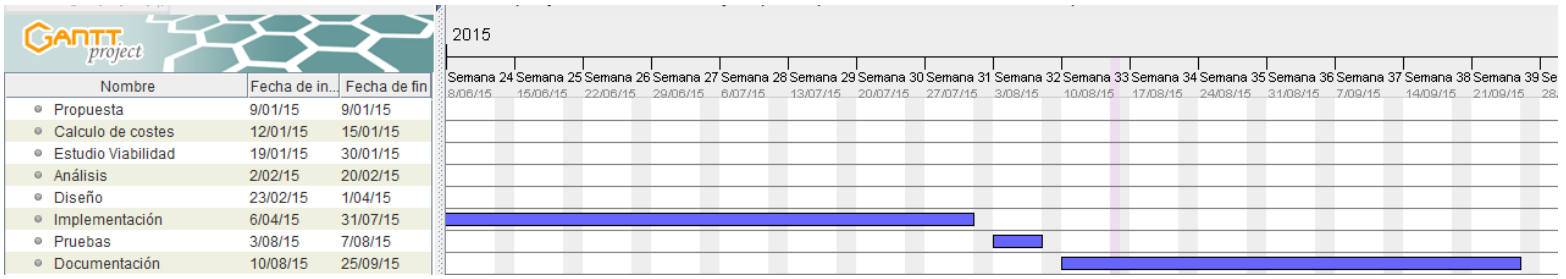


Ilustración 55: Diagrama de Gantt Semanas 2

Como se puede apreciar la fecha de inicio del proyecto es 9 de Enero del 2015 y la fecha de finalización del proyecto el 25 de septiembre del 2015.

Como podemos observar en el gráfico Gantt, nuestro proyecto empieza el día 9 de Enero del 2015, y termina el día 25 de septiembre del 2015, es decir, aproximadamente, el proyecto consta de 9 meses de duración.

6.2 Presupuesto

En este apartado se mostrará el presupuesto estimado para la realización del proyecto desglosado en diferentes apartados, todos los costes son sin I.V.A ya que este se añadirá cuando tengamos el coste total.

6.2.1 Resumen del tiempo dedicado

En este apartado se calcularan las horas invertidas en el proyecto, para calcular estas horas utilizaremos los datos aportados por el diagrama de Gantt.

A continuación se muestra en detalle las horas dedicadas para cada tarea y la suma total de ellas.

- Cálculo de costes: 5 días X 8 horas = 40 horas.
- Estudio Viabilidad: 10 días X 8 horas = 80 horas.
- Análisis: 15 días X 8 horas = 120 horas.
- Diseño: 28 días X 8 horas = 224 horas.
- Implementación: 85 días X 8 horas = 680 horas.
- Pruebas: 5 días X 8 horas = 40 horas.
- Documentación: 35 días X 8 horas = 280 horas.

Tras realizar los cálculos se estima que el tiempo total del proyecto es de 183 días de trabajo, o lo que es lo mismo 1464 horas.

6.2.2 Coste del personal

Se calculará el coste del personal dependiendo del rol desempeñado, ya que dependiendo del rol se dispondrá de un precio por hora diferente y trabajara en el proyecto una cantidad de días diferentes.

Nombre	Rol	Horas Trabajadas/Día	Días trabajados	Precio por Hora	Total
Miguel López Aguilar	Analista	8	94	17,50 €	13.160,00 €
Miguel López Aguilar	Diseñador	8	28	14,00 €	3.136,00 €
Miguel López Aguilar	Programador	8	90	12,00 €	8.640,00 €
				Total	24.936,00 €

Tabla 110: Coste del Personal

El coste del personal dedicado al proyecto para completar el mismo finalmente será de 24.936 €.

6.2.3 Costes de hardware

Para la realización del proyecto ha sido necesaria la adquisición de diferentes materiales hardware, para el cálculo de los costes hardware se ha utilizado la siguiente fórmula de amortización:

Amortización = (Dedicación/ (5 años*12 meses))*Uso en proyecto*Coste

Descripción	Coste(€)	% Uso dedicado al proyecto	Dedicación (meses)	Periodo de deprecación	Coste imputable (€)
Ordenador de Sobremesa	1.200,00 €	100	9	60	180,00 €
Smartphone	120,00 €	100	9	60	18,00 €
Arduino y Componentes	50,00 €	100	9	60	7,50 €
Cables	5,00 €	100	9	60	0,75 €
Vehículo	30,00 €	100	9	60	4,50 €
				Total	210,75 €

Tabla 111: Costes de Hardware

El coste total del hardware adquirido es de 210,75€.

6.2.4 Costes de Software

En este apartado se muestra el precio de las licencias de los componentes Software necesarios para este proyecto. Las licencias que no tienen coste alguno (gratuitas) no se han incluido en este presupuesto.

A continuación se detallan las licencias y el coste sin I.V.A.

Descripción	Coste (€)
Microsoft Office Homa & Student 2016	130,00 €
Licencia de Windows 7	80,00 €
Total	210,00 €

Tabla 112: Costes de Software

El coste total del Software adquirido es de 210,00 €.

6.2.5 Presupuesto Final

En este apartado se detalla el presupuesto final, que será la suma de los costes totales anteriormente calculados para más tarde añadirle el I.V.A. Para calcular el presupuesto total se han añadido los siguientes costes indirectos: se ha considerado que el riesgo es del 10% y que el beneficio que se quiere obtener al desarrollarlo es del 35%.

Descripción de costes	Coste Total (€)
Costes de personal	24.963,00 €
Costes de hardware	210,75 €
Costes de software	210,00 €
Subtotal	25.383,75 €
Riesgo (10%)	2.538,38 €
Beneficio (35%)	8.884,31 €
Total sin IVA	36.806,44 €
TOTAL (21% IVA incluido)	44.535,79 €

El coste total del proyecto sin I.V.A es de 36.806,44 € y aplicando el 21% de I.V.A el precio final del proyecto asciende a **44.535,79 €.**

7 Conclusiones y trabajos futuros

Después de realizar el proyecto es necesario dedicarle un apartado para reflexionar sobre los objetivos que se han conseguido así como una conclusión personal que indica la experiencia y sensaciones obtenidas mientras se realizó el proyecto. Y para terminar se indicarán las líneas futuras a las que se puede llevar el proyecto, que son una serie de mejoras que se pueden hacer en proyectos posteriores.

7.1 Conclusiones del proyecto

El objetivo principal del proyecto era diseñar e implementar un vehículo a escala para controlarlo con un Smartphone, con el cual el usuario podrá interactuar de manera manual o mediante voz para controlar el vehículo.

Por lo tanto, tras la realización del trabajo y realizar diversas pruebas se puede afirmar que el resultado final obtenido es un resultado favorable, ya que se consiguió cumplir todos los objetivos que se propusieron de manera satisfactoria.

Los objetivos indicados al principio del proyecto que se han alcanzado con éxito son los siguientes:

- ✓ Diseñar e implementar una aplicación para Android que nos permita conectarnos al vehículo y controlarlo a través de Bluetooth de manera manual.

- ✓ Diseñar e implementar una aplicación para Android que nos permita conectarnos al vehículo y controlarlo a través de Bluetooth mediante comandos de voz.

- ✓ Diseñar e implementar un Hardware que sea válido para controlar el vehículo y recibir instrucciones a través de Bluetooth.

- ✓ Diseñar e implementar un código que ira cargado en la placa Arduino para darle a esta una funcionalidad útil deseada.

Así mismo también se cumplieron los siguientes sub-objetivo:

- ✓ Desarrollar un sistema un sistema anti-choque para evitar que el vehículo colisione frontalmente.

- ✓ Diseñar unas interfaces muy intuitivas para permitir que un usuario inexperto sea capaz de controlar el vehículo.

Para llegar a este resultado final favorable surgieron pequeños inconvenientes pero se solventaron de manera rápida y óptima, el problema más grande a destacar es el bajo conocimiento que tenía para programar sobre Android, esto hizo que se invirtió gran parte del tiempo de realización del proyecto en consultar diferentes tutoriales, un claro ejemplo es que en un principio creía que la interfaz gráfica de Android tendría Joystick por defecto, pero al no tenerlos hizo que los tuviese que diseñar y crear perdiendo bastante tiempo ya que fue una tarea bastante difícil al tener bajos conocimientos.

Tras los inconvenientes de mi nivel de Android no surgieron más de gran importancia, pero otro de los problemas es que al utilizar la tecnología Bluetooth al ser inalámbrica en ocasiones falla, pudiendo provocar que el comportamiento no sea el adecuado. Por lo demás no se tuvieron más problemas a destacar, ya que el Hardware funcionó bien y no se estropeo ningún elemento y programar sobre Arduino no fue ningún inconveniente. Además la comunicación entre Arduino y el Smartphone funcionó bien y se consiguió una conexión bastante eficiente para controlar el vehículo sin muchos retrasos, pero en el caso de la aplicación controlada mediante voz no solo depende de la conexión Bluetooth, sino que también depende de la conexión a internet ya que para pasar el audio a texto, el audio necesita ser enviado a un servidor el cual nos devuelve los resultados en formato texto.

Para concluir indicar que aun teniendo estos problemas el resultado obtenido fue el resultado esperado y por lo tanto se cumplió el objetivo del proyecto con éxito.

7.2 Conclusiones personales

Lo primero es indicar que el proyecto ha sido un proyecto muy gratificante para mí y por lo tanto se me a echo muy ameno realizarlo, ya que aun llevando mucho tiempo programando siempre programaba sobre un ordenador y solo realizaba un software, en este caso programe sobre un Smartphone y sobre un hardware diseñado por mí, cosa que nunca había realizado. Por lo tanto no realicé solo una parte lógica (software) sino que también realicé una parte física (hardware) para poder cumplir el objetivo del proyecto.

La parte del vehículo que implica la programación del hardware y del software para arduino se realizó de manera muy rápida y sencilla. Ya que tras realizar el diseño todo fue sin ningún problema, dado que el lenguaje de programación para arduino se basa en C y C++, los cuales son lenguajes de alto nivel fáciles de comprender y muy estudiados durante el grado. Y para montar el hardware existían muchos tutoriales muy buenos en internet, así como que ya se tenía conocimientos de muchos elementos vistos en alguna asignatura.

Por otro lado está la programación en Android que fue un poco más complicada, ya que aunque se estudia durante el grado y su lenguaje de programación sea JAVA que es un lenguaje sencillo, aún no tenía unos grandes conocimientos para programar sobre el Smartphone. Esto hizo que esta fuese la tarea más complicada del proyecto pero por suerte apoyándome en los pocos conocimientos que tenía y en unos buenos tutoriales en internet conseguí realizar de manera satisfactoria las dos aplicaciones. Las cuales tenían su complicaciones cada una: Por un lado crear la interfaz a tiempo real para realizar el control manual y por otro lado conseguir un reconocimiento de Voz con una interfaz válida, ya que la mayoría de reconocimiento de voz que se utilizan usan la interfaz de búsqueda de Google, la cual hacía que la interfaz que se diseñó para el proyecto fuese más engorrosa para el usuario y difícil de utilizar.

Para concluir indicar que no se tuvo ningún problema grave ya que todos los problemas que fueron apareciendo los supe ir solventando y no me tuvieron mucho tiempo parado esperando una solución óptima.

7.3 Trabajos futuros

Este proyecto deja la posibilidad de realizar sobre él varios trabajos futuros, con el fin de mejorar su desarrollo en el futuro se indicarán las diferentes propuestas que bajo mi punto de vista se pueden desarrollar:

- Conseguir realizar un traductor de voz a texto en local, para que no sea necesario enviar la voz a un servidor y que este no la traduzca, esto permitirá que podamos ejecutar la aplicación para el control de voz y no sea necesaria la conexión a internet.

- Utilizar el WIFI en lugar del Bluetooth, en este proyecto para la comunicación inalámbrica se utiliza el sistema Bluetooth y para trabajos futuros se propone cambiar esta conexión inalámbrica por una conexión WIFI que tenga mayor alcance y mayor ancho de banda.

- Otra mejora que se puede añadir en el futuro es realizar un estudio para intentar mejorar la velocidad de transmisión inalámbrica ya que en ocasiones no es tan rápida como se desea.

- Ponerle una cámara al vehículo para el reconocimiento de objetos y dotarlo de cierta inteligencia artificial.

- Integrar el Smartphone en el vehículo y darle inteligencia, para que sea capaz de seguir una línea a partir de la cámara del Smartphone y la capacidad de procesamiento de este.

- Añadir un sistema para la potencia ya que en el caso del control mediante voz el vehículo se desplaza demasiado rápido y es difícil de controlar.

8 Bibliografía

Referencias

[1] Anónimo, (2015). [En línea] Disponible en: http://developer.android.com/reference/android/bluetooth/BluetoothDevice.htm [Consulta: 15 Septiembre 2015].

[2] Anónimo, (2015). [En línea] Disponible en: http://pdf.datasheetcatalog.com/datasheet/SGSThomsonMicroelectronics/mXurruu.pdf [Consulta: 15 Septiembre 2015].

[3] Arduino.cc, (2015). *Arduino - Home*. [En línea] Disponible en: https://www.arduino.cc/ [Consulta: 15 Septiembre 2015].

[4] Arduino.cc, (2015). *Arduino - ArduinoBoardMega*. [En línea] Disponible en: https://www.arduino.cc/en/Main/arduinoBoardMega [Consulta: 15 Septiembre 2015].

[5] Arduino.cc, (2015). *Arduino - Reference*. [En línea] Disponible en: https://www.arduino.cc/en/Reference/HomePage [Consulta: 15 Septiembre 2015].

[6] Arduino.cc, (2015). *Arduino - Software*. [En línea] Disponible en: https://www.arduino.cc/en/Main/Software [Consulta: 15 Septiembre 2015].

[7] Ajpdsoft.com, (2015). *Conectar pantalla LCD a Arduino UNO e interactuar con ella, mostrar temperatura Proyecto AjpdSoft*. [En línea] Disponible en: http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=627 [Acceso 15 Sep. 2015].

[8] Barbus, E. (2014). *El cajón de Arduino: Tutorial: sensor ultrasonidos HC-SR04*. [En línea] Elcajondeardu.blogspot.com.es. Disponible en: http://elcajondeardu.blogspot.com.es/2014/03/tutorial-sensor-ultrasonidos-hc-sr04.html [Consulta: 15 Septiembre 2015].

[9] Developer.android.com, (2015). *android.speech | Android Developers*. [En línea] Disponible en: http://developer.android.com/reference/android/speech/package-summary.html [Consulta: 15 Septiembre 2015].

[10] Developer.android.com, (2015). *Intent | Android Developers*. [En línea] Disponible en: http://developer.android.com/reference/android/content/Intent.html [Consulta: 15 Septiembre 2015].

[11] Developer.android.com, (2015*). RecognitionListener | Android Developers*. [En línea] Disponible en: http://developer.android.com/reference/android/speech/RecognitionListener.html [Consulta: 15 Septiembre 2015].

[12] Developer.android.com, (2015). *Socket | Android Developers*. [En línea] Disponible en: http://developer.android.com/reference/java/net/Socket.html [Consulta: 15 Septiembre 2015].

[13] Developer.android.com, (2015). *SpeechRecognizer | Android Developers*. [En línea] Disponible en: http://developer.android.com/reference/android/speech/SpeechRecognizer.html [Consulta: 15 Septiembre 2015].

[14] Developer.android.com, (2015). *SurfaceView | Android Developers*. [En línea] Disponible en: http://developer.android.com/reference/android/view/SurfaceView.html [Consulta: 15 Septiembre 2015].

[15] DIYMakers, (2014). *Arduino + Bluetooth*. [En línea] Disponible en: http://diymakers.es/arduino-bluetooth/ [Consulta: 15 Septiembre 2015].

[16] Eclipse Foundation, I. (2015). [En línea] Eclipse.org. Disponible en: https://eclipse.org/home/index.php [Consulta: 15 Septiembre 2015].

[17] Es.wikipedia.org, (2015*). Android Gingerbread*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Android_Gingerbread [Consulta: 15 Septiembre 2015].

[18] Es.wikipedia.org, (2015). *Eclipse (software)*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Eclipse_(software) [Consulta: 15 Septiembre 2015].

[19] Es.wikipedia.org, (2015). *Placa de pruebas*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Placa_de_pruebas [Consulta: 15 Septiembre 2015].

[20] Fritzing.org, (2015). *Fritzing*. [En línea] Disponible en: http://fritzing.org/home/ [Consulta: 15 Septiembre 2015].

[21] García, D. (2013). *Bluetooth (II): Descubriendo dispositivos.* [En línea] Let's code something up!. Disponible en: https://danielggarcia.wordpress.com/2013/10/21/bluetooth-ii-descubriendo-dispositivos/ [Consulta: 15 Septiembre 2015].

[22] García, D. (2013). *Bluetooth (I): Activando y desactivando el Bluetooth en Android*. [En línea] Let's code something up!. Disponible en: https://danielggarcia.wordpress.com/2013/10/19/bluetooth-i-activando-y-desactivando-el-bluetooth-en-android/ [Consulta: 15 Septiembre 2015].

[23] Google Developers, (2015). *Google Plugin for Eclipse 4.4 (Luna) Installation Instructions*. [En línea] Disponible en: https://developers.google.com/eclipse/docs/install-eclipse-4.4 [Consulta: 15 Septiembre 2015].

[24] Guindon, C. (2015). [En línea] Eclipse.org. Disponible en: https://www.eclipse.org/luna/ [Consulta: 15 Septiembre 2015].

[25] Gupt, M. (2014). *Android Speech Recognition Without Dialog In A Custom Activity - Truiton*. [En línea] Truiton. Disponible en: http://www.truiton.com/2014/06/android-speech-recognition-without-dialog-custom-activity/ [Consulta: 15 Septiembre 2015].

[26] Ingeniería en Mantenimiento Industrial., (2015). *BLUETOOTH HC-06 | CONFIGURACIÓN CON ARDUINO*. [En línea] Disponible en: http://ingeerick.weebly.com/arduino/bluetooth-hc-06-configuracin-con-arduino [Consulta: 15 Septiembre 2015].

[27] Oracle.com, (2015*). Java SE - Downloads | Oracle Technology Network | Oracle*. [En línea] Disponible en: http://www.oracle.com/technetwork/java/javase/downloads/index.html [Consulta: 15 Septiembre 2015].

[28] YouTube, (2015). *Sensor de Distancia (proximidad) hasta 5 metros / HC-SR04 y ARDUINO*. [En línea] Disponible en: https://www.youtube.com/watch?v=IF1eN0WK3bU [Consulta: 15 Septiembre 2015].

[29]Developer.android.com, (2015). *Intent | Android Developers.* [En línea] Disponible en: http://developer.android.com/reference/android/content/Intent.html [Consulta: 15 Septiembre 2015].

[30]Es.wikipedia.org, (2015). *Portada*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Wikipedia:Portada [Consulta: 15 Septiembre 2015].

[31]Es.wikipedia.org, (2015*). Ambiente de desarrollo integrado*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Ambiente_de_desarrollo_integrado [Consulta: 15 Septiembre 2015].

[32]Es.wikipedia.org, (2015). *Android.* [En línea] Disponible en: https://es.wikipedia.org/wiki/Android [Consulta: 15 Septiembre 2015].

[33] Es.wikipedia.org, (2015*). APK (formato)*. [En línea] Disponible en: https://es.wikipedia.org/wiki/APK_(formato) [Consulta: 15 Septiembre 2015].

[34] Es.wikipedia.org, (2015). *Arduino*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Arduino [Consulta: 15 Septiembre 2015].

[35] Es.wikipedia.org, (2015). *Baudio*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Baudio [Consulta: 15 Septiembre 2015].

[36] Es.wikipedia.org, (2015). *Bluetooth*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Bluetooth [Consulta: 15 Septiembre 2015].

[37] Es.wikipedia.org, (2015*). Circuito integrado*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Circuito_integrado [Consulta: 15 Septiembre 2015].

[38] Es.wikipedia.org, (2015). *Eclipse (software*). [En línea] Disponible en: https://es.wikipedia.org/wiki/Eclipse_(software) [Consulta: 15 Septiembre 2015].

[39] Es.wikipedia.org, (2015). *Formato de compresión ZIP*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Formato_de_compresión_ZIP [Consulta: 15 Septiembre 2015].

[40] Es.wikipedia.org, (2015). *Hardware*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Hardware [Consulta: 15 Septiembre 2015].

[41] Es.wikipedia.org, (2015). *Interfaz de usuario*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Interfaz_de_usuario [Consulta: 15 Septiembre 2015].

[42] Es.wikipedia.org, (2015). *Kit de desarrollo de software*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Kit_de_desarrollo_de_software [Consulta: 15 Septiembre 2015].

[43] Es.wikipedia.org, (2015). *Puente H (electrónica*). [En línea] Disponible en: https://es.wikipedia.org/wiki/Puente_H_(electrónica) [Consulta: 15 Septiembre 2015].

[44] Es.wikipedia.org, (2015). *Sistema operativo*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Sistema_operativo [Consulta: 15 Septiembre 2015].

[45] Es.wikipedia.org, (2015). *Socket de Internet*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Socket_de_Internet [Consulta: 15 Septiembre 2015].

[46] Es.wikipedia.org, (2015). *Software*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Software [Consulta: 15 Septiembre 2015].

[47] Es.wikipedia.org, (2015). *Teléfono inteligente*. [En línea] Disponible en: https://es.wikipedia.org/wiki/Teléfono_inteligente [Consulta: 15 Septiembre 2015].

[48] Esi.unav.es, (2015). *Qué es el JDK (Java Development Kit).* [En línea] Disponible en: http://www.esi.unav.es/Asignaturas/Informat2/Clases/Clases9899/Clase01/JavaEntorno/tsld003.htm [Consulta: 15 Septiembre 2015].

[49] Developer.android.com, (2015). *BluetoothDevice | Android Developers*. [En línea] Disponible en: http://developer.android.com/reference/android/bluetooth/BluetoothDevice.html [Consulta: 15 Septiembre 2015].

[50] Programamos, (2015). Entornos gráficos de Programación con Arduino. [En línea] Disponible en: http://programamos.es/entornos-graficos-de-programacion-con-arduino/ [Consulta: 15 Septiembre 2015].

[51] SL, M. (2015). *Gantt Project (Windows).* [En línea] Uptodown.com. Disponible en: http://gantt-project.uptodown.com/ [Consulta: 15 Septiembre 2015].

9 Plan 2011

9.1 Introduction

This section will serve as document introduction and it´s about motivation witch had let us to finish the project, like so the objectives that we have gotten. To facilitate the document understanding and motivations that led you to perform this project is required the following section.

9.1.1 Motivation

The voice control technology is going on day to day, and this is one of the most interesting for the users, this is why today its one of the most implements technologies. This one is used to control instructions like a television control, or in a vehicle main console...

At the present time it's been developing a lot of android applications for smartphone to use its wireless connectivity, controlling by voice other tools that were not designed for this purpose.

In this case, it will be done a design and implementation of a scale vehicle control by voice, this is why i modify a radio control vehicle to use it through an android smartphone via Wi-Fi. In this project it will be control a vehicle in two different ways: Manual and Voice.

The manual control consist on directing the vehicle by 2 joysticks, using the smartphone screen, the right joystick control the moving forward and backward, and the left joystick control the moving right or moving left.

Voice control is based on moving the vehicle by voice commands, using the speech recognition powered by Google, the allowed commands are the followings: move forward, move backward, move left and move right .In this document will be detailed the steps follow for a correct design of the vehicle to lead the following designers.

9.1.2 Objective

The main objective of the project is to design and implements a scale vehicle, to control with an Android smartphone through Bluetooth. It'd why you need to have basic knowledge about arduino electronic boards and write code for its.

The project has 2 parts:

- The first, consist on designing and implementation of an electronic circuit to allow arduino to receive instructions through Bluetooth and before this, we will be able to control the vehicle.

- The second part consist on designing software, it will be designed and implements 2 applications, both of them to communicate arduino and the smartphone.

Before show witch is the main objective of the project, we will continue dividing this into smaller parts that we have to complete to finish the project:

- ❖ Designing and implementation of an android application, that let us to connect the vehicle via Bluetooth and control it with joysticks

- ❖ Designing and implementation of an android application that let us to connect the vehicle via Bluetooth and control it with the human voice.

- ❖ Designing and implementation a hardware that used for controlling and receiving commands via Bluetooth.

- ❖ Designing and implementing an algorithm to charge into arduino board to give this the functionality that we want.

To complete this objective it intends the following secondary objectives:

- ❖ Designing of a shockproof system to avoid the vehicle collision

- ❖ Designing an userfriendly interfaces to let noob users control the vehicle.

To conclude this project is necessary to complete the main objective, like so all the secondary objectives, only this way we will say that it was done.

9.1.3 Document Structure

To facilitate the reading documents of the project, we going to do a little description of the different parts that compose the project:

1. **Introduction**: The introducing section that explain you the motivation, objectives and the structure of the document.

2. **State of the art**: section that describe an actual situation and the different ways that exist to resolve a problem.

3. **System analysis**: section in which is indicated the user needs and define the requirements and use cases.

4. **Design and implementation**: this section describes the planification and how it has implemented to reach a feasible solution.

5. **Tests and evaluation**: Part that describe a planification the test that it's going to execute to verify the correct work of the applications.

6. **Planning and the budget**: Part in which is described the planning that we have follow to develop and predict the budget that we need.

7. **Conclusions and future work**: Part in witch is realized the conclusion of the project and the personal conclusions, to conclude it will be include new ways and ideas to improve the project.

8. **Bibliography**: Part in witch is found the references used along of the project

9.2 Abstract

To begin we have the project documents that show the hardware and software systems, controlling via voice or manually, and what are the advantages and disadvantages. The voice controls of the vehicles allow the user to have the control over the vehicle don´t keep his eyes on the road. Other systems are televisions controlling by voice and gestures, the smart TV is a new technological step, because TVs can be used without a controller, this way the user don't have to be worried about the controller and where it is. In addition exist domotic houses that recognize your voice to do something.

Before show the advantages of this evolving technology, the next step will be to describe the different tools for the hardware development, in this case for being a resume we will show the different alternatives, and the necessary tools, with the final purpose of explaining the use and the understanding of the project.

Arduino mega: Arduino mega was used to receive the instructions from the Smartphone and change this into commands that direct the vehicle.

Module Bluetooth: The Bluetooth module (Hc-06) is used for the communication between arduino and the smartphone.

Chip L293B: It is used to realize a bridge H with transistors, to control the engine by direct current.

LCD Keypad Shield: LCD Keypad Shield is a module for arduino, which we will show the user the vehicle mode and the information of itself.

Module: HC-SR04: The module HC-SR04 is a remote sensor, using to avoid the vehicle hit against obstacles.

R/C scale vehicle: This vehicle controlling by radio signal will be a toy that will be unmounted to remove the circuit, except the direct current engine.

Arduino Battery: the battery is used to feed the arduino board to no necessity of being connected to an E/S of the computer.

Wires: The wires are used to all the connections between the components.

Battery 9V: The battery the 9 volitions is used to feed the arduino mega board and so to work and not have to be connected to the computer.

Test Board: The test board is a simply board in which is connected the components, for later wire up it.

Android OS: To install the applications that control the vehicle have been used an Orange MonteCarlo with Gingerbread Androids OS

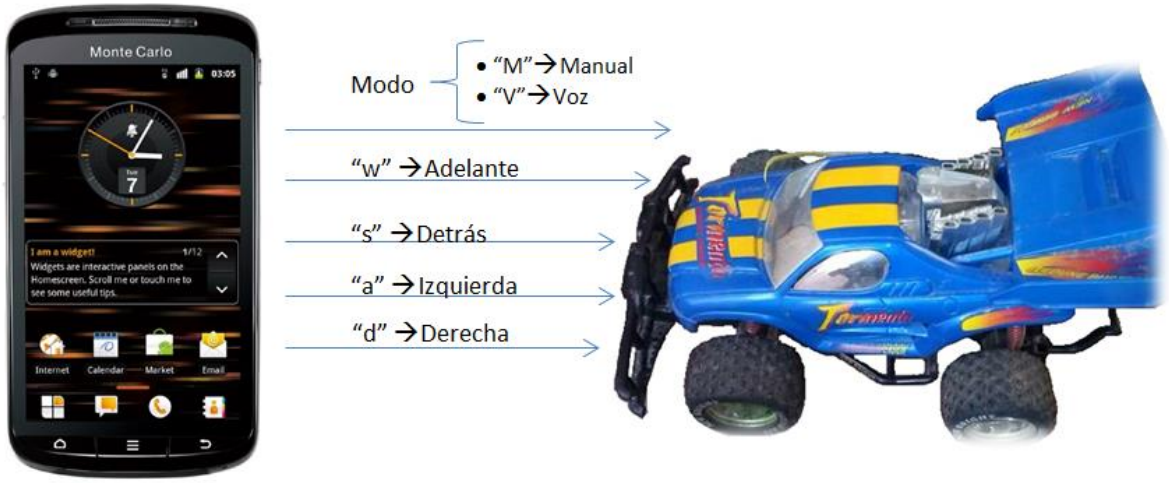
Programing environment for Arduino: the programing environment for Arduino is used for writing the code and loading in arduino mega board (programing the board).

Eclipse: To the programing on android was used an Eclipse software in the moon version.

To continue it will realize a study of the system for which it will realize a definition of all user requirements and system requirements, for all the application we need. Also it will realize the use cases of the 3 apps.

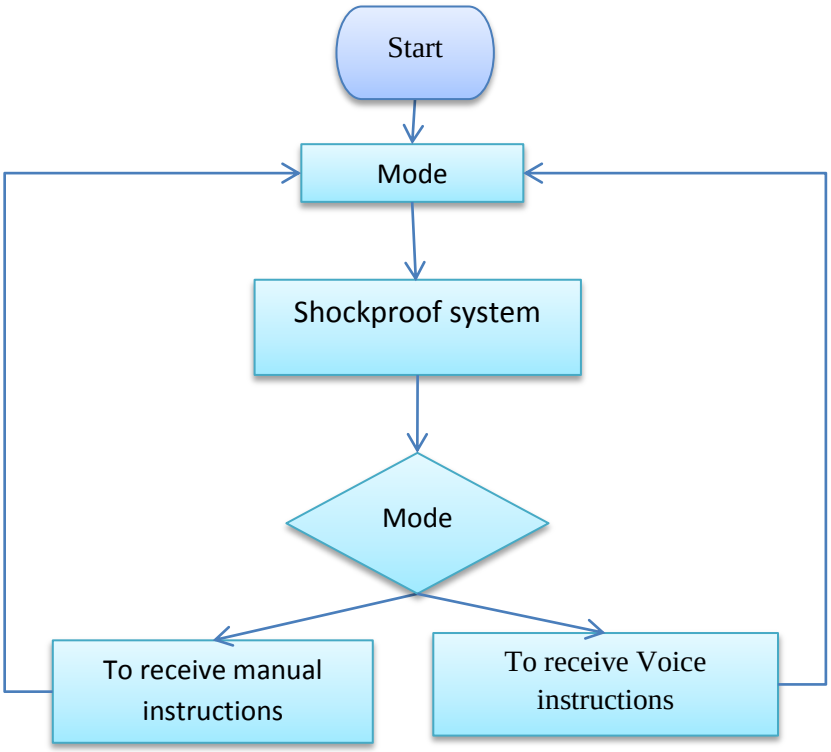
Once we have this we will continue analyzing, developing and implementing, being the last in which spend more time.

Before beginning we have to install the programing environments to android like so arduino, in which we will develop the code for the correct functionality of these. The communication will be via sending character from the smartphone to the vehicle, and to conclude the characters will be showed:



For being a resume we'll leave out the interfaces and we will pass to describe the main code structure:

1. Arduino Structure:

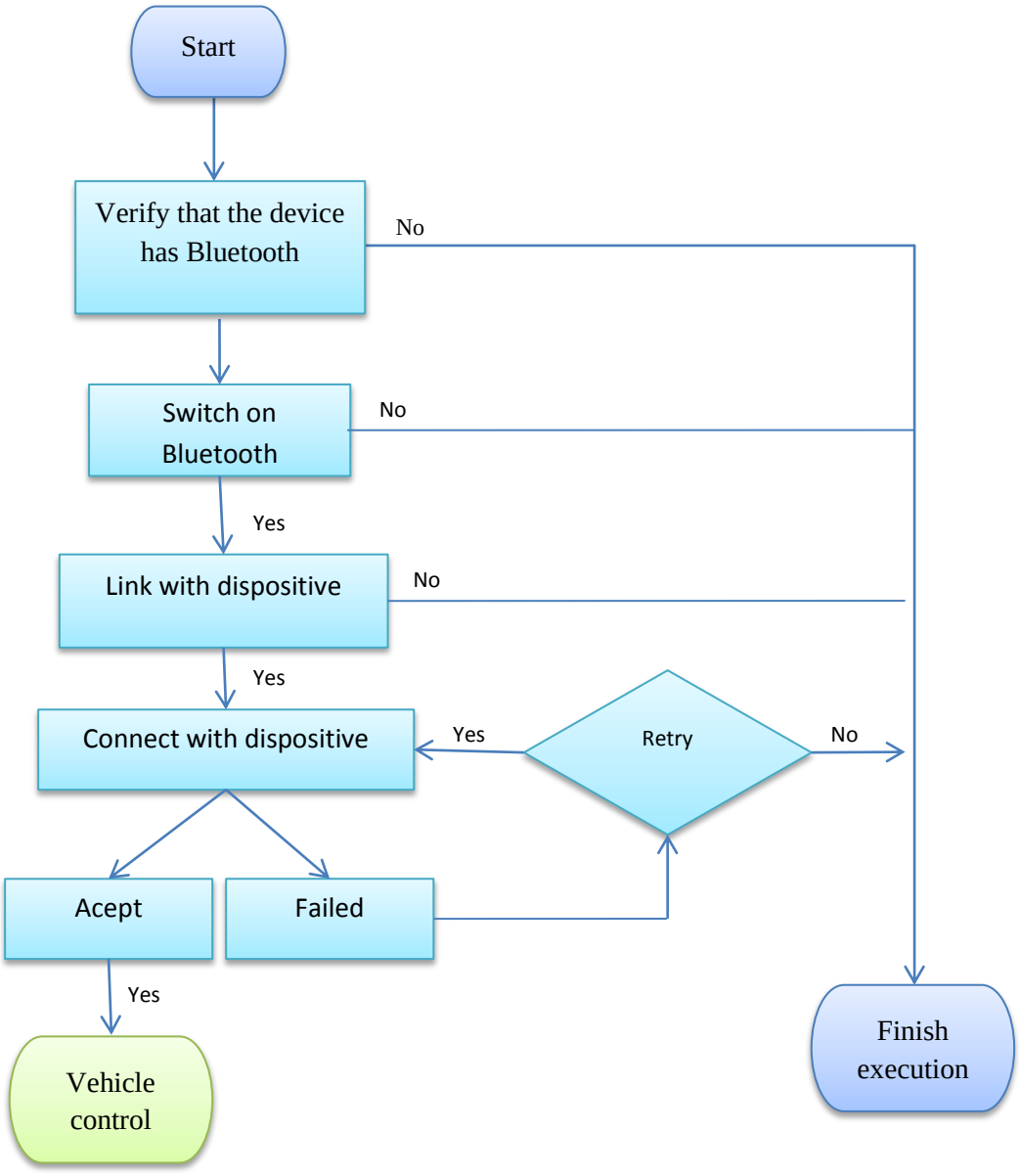


In the flow diagram we can see two modes: one manual and other for voice control. The mode will have the obligation of showing all the commands that have been executed and if the mode is manual or voice.

When we know which is the execution mode, it will be showed on a LCD screen of the vehicle, and it will be modified a global variable to response later to the question which mode is the vehicle, and if the vehicle can receive commands and instructions successfully. Before this we will continue ending the arduino code.

2. Application Android Structure:

Once the code for the arduino board is finished, we will continue with the developing and the implementation for the android app. For this the first we can do is the designing of the code that is showed in the following flow diagram:

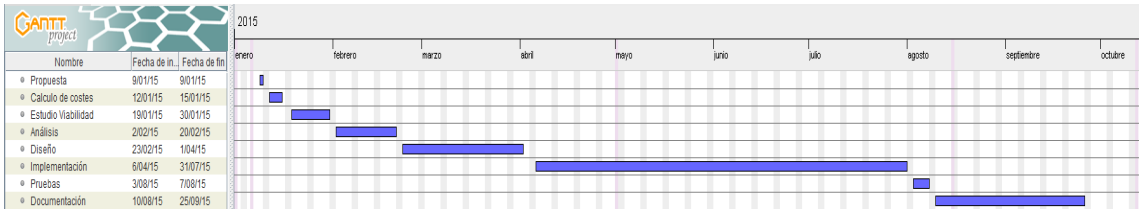


How is appreciated on the diagram the first we realize is the connection that is the same on voice control or manual control, and to do this work successfully, we have to implement the software.

When we have the designing and the implementation of the software and the hardware, we will continue with a list of test to verify the functionality of all elements of the system, and if was developed successfully. In this case we will do stress tests because the vehicle only supports one of the wireless connections.

Before we realized the planification and the budget, the point named planification we establish the first planification for the project, but this planification is possible that can suffers modification. This app is detailed in a list which we will show all the activities that can be realized for the project, how long we have to spend to finish and the Gantt diagram.

Activity	Start date	End date	Duration
Proposal	09/01/2015	09/01/2015	1
Costing	12/01/2015	15/01/2015	5
Feasibility Study	19/01/2015	30/01/2015	10
Analysis	02/02/2015	20/02/2015	15
Design	23/02/2015	01/04/2015	28
Implementation	06/04/2015	31/07/2015	85
Testing	03/08/2015	07/08/2015	5
Documentation	10/08/2015	25/09/2015	35



How you can see on the Gantt diagram, the project begins on 9th January in 2015, and the project is about 9 moths.

The budget will be an estimated budget for the project, in this resume it only showed the final budget that is the adding up of all cost calculated later, and then add up the IVA. To calculate the total budget it has been added up the next indirect costs: it is considered that the risk is about 10% and the money we want to earn is about 35 %.

Description of costs	Total cost (€)
Staff costs	24.963,00 €
Hardware costs	210,75 €
Software costs	210,00 €
Subtotal	25.383,75 €
Risk (10%)	2.538,38 €
Benefit (35%)	8.884,31 €
Total without IVA	36.806,44 €
TOTAL (21% IVA included)	44.535,79 €

The total cost for the project without I.V.A is about 36.806,44 € and applying the 21% the I.V.A for final project price ascend to **44.535,79 €.**

9.3 Conclusions and Future Work

Before to conclude the project is necessary to have a section to think about the objective that have been gotten, like so a personal conclusion that indicate the experience and sensations gotten while the project was realized. To conclude it will be indicated the new ways that the project can follow, this improvements can be used for later projects.

9.3.1 Conclusions Project

The main objective of the project was designs and implements a scale vehicle to control it with a smartphone; it will be used by the user with manual control, or voice control to direct the vehicle.

Then, before the ending of project and doing many tests, it is confirmed that the final result is a nice result, because it is gotten all goals satisfactorily.

The objectives indicated in the beginning of the project have been reached successfully and these are the followings:

- Designing and implementation of an android application, that let us to connect the vehicle via Bluetooth and control it with joysticks
- Designing and implementation of an android application that let us to connect the vehicle via Bluetooth and control it with the voice human.
- Designing and implementation a hardware that used for controlling and receiving commands via Bluetooth.
- Designing and implementing an algorithm to charge into arduino board to give this the functionality that we want.

Also the secondary object has been completed, and these are the followings:

- Designing of a shockproof system to avoid the vehicle collision
- Designing a userfriendly interfaces to let noob user control the vehicle.

Before to get the fair final result, we have a lot of small problems, but these were fixed and solved in a fast and optimal way, the biggest problem was to program on android because i haven't experience and need to find and study tutorials and programing books. A perfect example was the designing and developing of 2 joysticks used to direct the vehicle.

The second major problem was to use Bluetooth technology, because this is wireless and sometimes the connection is lost and add up hours to the project. The rest of the project all work good and i have no problem with the software or Android hardware.

This is the way that guide me to get a firm and stable connection, that give you tools to direct the vehicles without many delays, but not so in the voice control mode, because it used a external Google service to change the voice input to a text output, this process require time and this makes the vehicle suffer delays, and of course, it needed internet connection to use this service, that why the application not only depends on Bluetooth, depends on smartphone Internet too.

To conclude, i have to say that despite of all problems, the application is fully operational, and cover all the requirements.

9.3.2 Personal conclusions

First it´s report that the project have been a gratifying, because i spend a lot of time to program the software and design the arduino circuit, both things i like, and i have to say that i never do this before. Therefore it was a funny multidisciplinary job for me.

On the one hand, the part of the project that involves arduino software was done easily and quickly, therefore before to make the software design, it was easy to use C and C++ to program the arduino software, because C language was so studied in during my university education.

On the other hand, the android programing was a little more complicated, because on despite of all the career we was studying Java, i have no a big knowledge about Smartphone programing, that do the work more complicated, but seeing videos and seeing books i can improve and end the two applications satisfactorily. Both applications had problems but these were solved: create an interface that works in real time, get the speech recognition use Google services…

To conclude, i have to indicate that there had not hard problems, because all of this was fixed by me, therefore i think that i reach all goals that was planned for my project, and i think that the university court will my effort and value me positively.

9.3.3 Future Work

This project give the possibility to realize future work using it, to improve the developing of this following work it will be indicated the different propositions that we think that can be developed for following improvements:

- Getting make a local voice translator, this way we don´t be needed to send the voice from the smartphone to a Google server, this allow to execute the application without the need of an Internet connection.

- Using Wi-Fi instead of Bluetooth, because in this project is used a Bluetooth and future works could change this connection and use only Wi-Fi, this way the vehicle can be directed via internet.

- Other improvement that can be added is to realize a study to improve the velocity of the wireless connection, because is not as fast as we deserve.

- Add a camera to the vehicle recognize objects and give it some artificial intelligence

- Integrating the smartphone into the vehicle, this way using the smartphone camera the vehicle could have a better computing capacity

- Add a system to control the vehicle power, because the voice control does the vehicle run so fast.

10 Anexos

10.1 Anexo 0: Instalación del entorno

10.1.1 Instalación del entorno de programación para Android con Eclipse

En esta apartado se indicaran los pasos a seguir para la descarga e instalación del entorno de programación Eclipse.

10.1.2 Descargar e instalar java JDK

Para descargar java JDK, iremos a su página oficial de descarga [27] y seguiremos los siguientes pasos:

Paso 1: En dicha página podremos observar la imagen de la ilustración 16, tras ello seleccionaremos java plataform (JDK) 8u51.

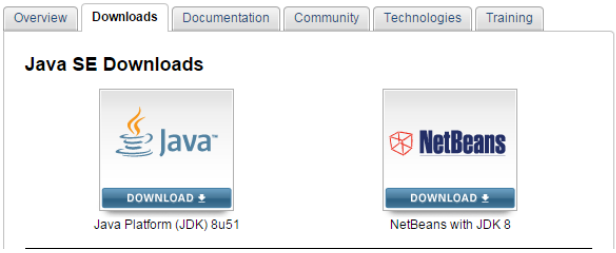


Ilustración 56: Descargar Java JDK 1

Paso 2: Y nos parecerá la siguiente pantalla (Ilustración 57: Descargar Java JDK 2), en la cual aceptaremos los términos de uso y seleccionaremos nuestro sistema operativo para comenzar la descarga.

Java SE Development Kit 8u51

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux x86	146.9 MB	jdk-8u51-linux-i586.rpm
Linux x86	166.95 MB	jdk-8u51-linux-i586.tar.gz
Linux x64	145.19 MB	jdk-8u51-linux-x64.rpm
Linux x64	165.25 MB	jdk-8u51-linux-x64.tar.gz
Mac OS X x64	222.09 MB	jdk-8u51-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.36 MB	jdk-8u51-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	98.8 MB	jdk-8u51-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	139.79 MB	jdk-8u51-solaris-x64.tar.Z
Solaris x64	96.45 MB	jdk-8u51-solaris-x64.tar.gz
Windows x86	176.02 MB	jdk-8u51-windows-i586.exe
Windows x64	180.51 MB	jdk-8u51-windows-x64.exe

Ilustración 57: Descargar Java JDK 2

Paso 3: Tras realizar la descarga ejecutaremos como administrador el archivo descargado para instalar java JDK, en la primera pantalla que se muestre pulsaremos sobre Next:

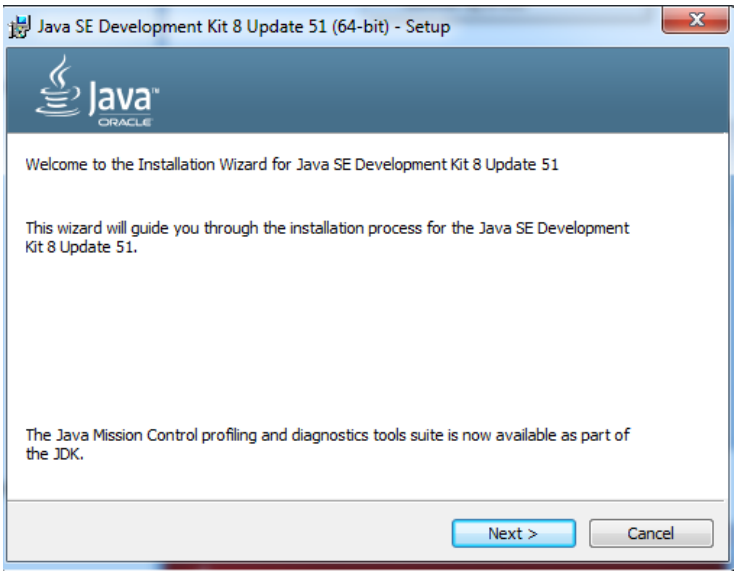


Ilustración 58: Instalar Java JDK 1

Paso 4: Volvemos a pulsar sobre Next.

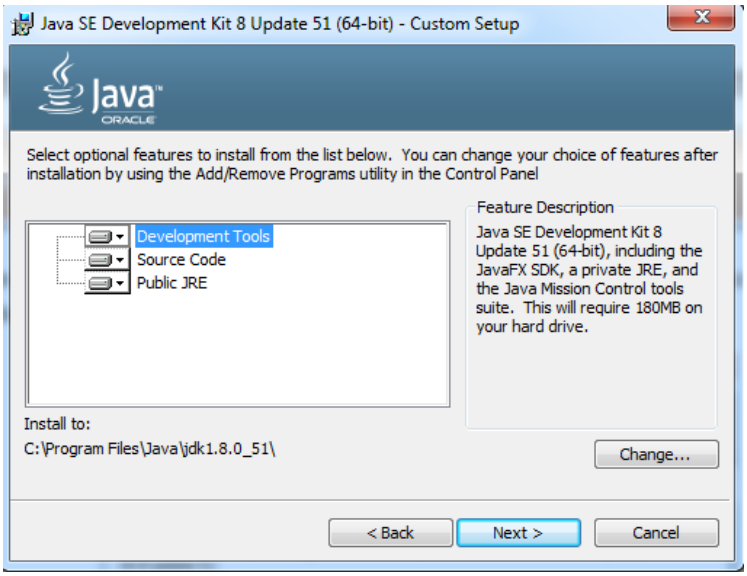


Ilustración 59: Instalar Java JDK 2

Paso 5: Esperamos a que se complete la operación.

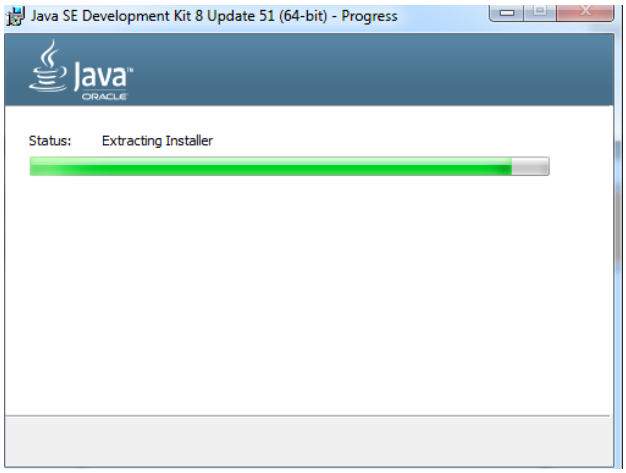


Ilustración 60: Instalar Java JDK 3

Paso 6: Elegimos el lugar donde lo queremos instalar y pulsamos siguiente.

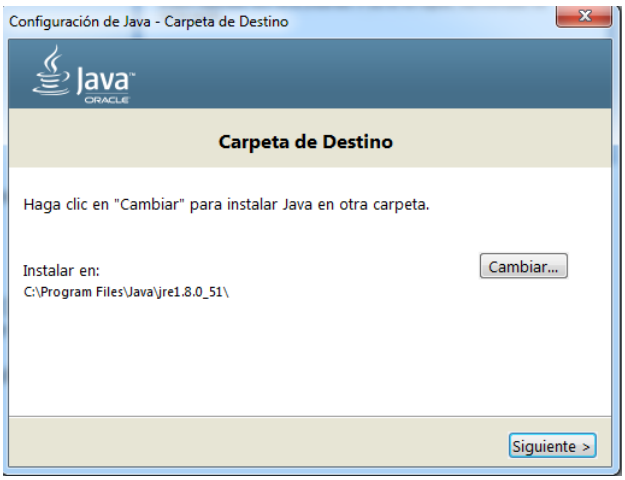


Ilustración 61: Instalar Java JDK 4

Paso 7: Esperamos a que se complete la operación.

Ilustración 62: Instalar Java JDK 5

Paso 8: Pulsamos Close para completar la instalación.

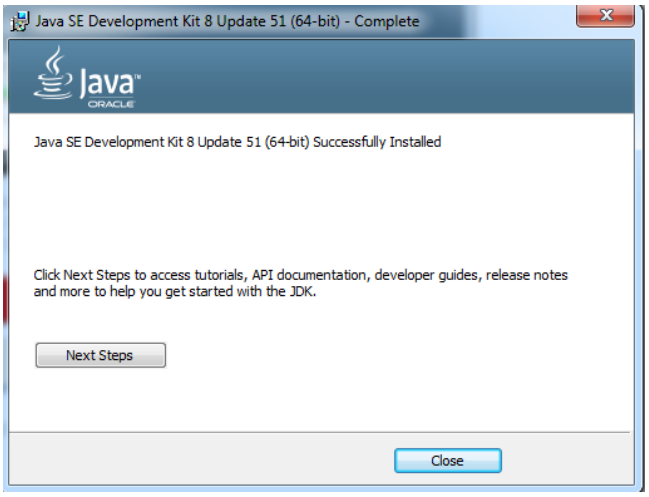


Ilustración 63: Instalar Java JDK 6

10.1.3 Instalación de Eclipse con el conjunto ADT

Tras instalar Java JDK el siguiente paso será instalar Eclipse Luna con el conjunto ADT, para ello nos iremos a la página oficial de Eclipse [16] y seguiremos los siguientes pasos:

Paso 1: Una vez en ella pulsaremos sobre DOWNLOAD.

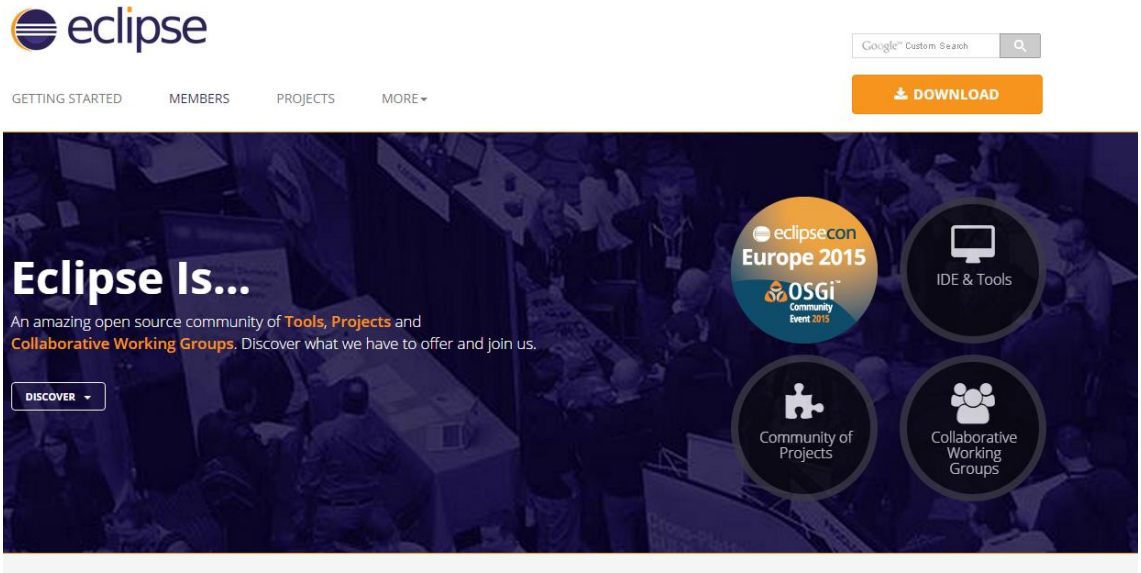


Ilustración 64: Instalar Eclipse 1

Paso 2: En la siguiente pantalla que nos aparezca buscaremos MORE DOWNLOADS, y pulsaremos sobre Eclipse Luna (4.4), para dirigirnos a la siguiente página web [24] que nos llevara a la Ilustración 66: Instalar Eclipse 3.

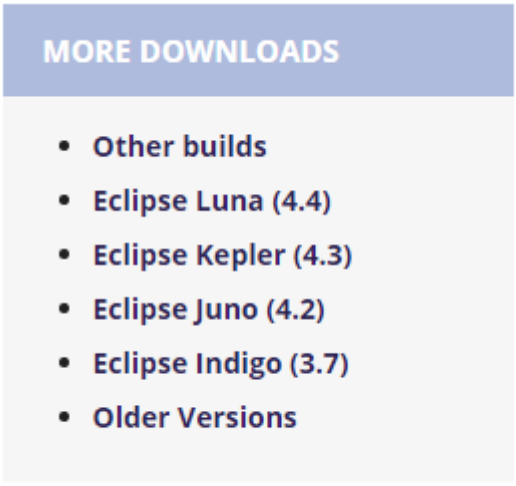


Ilustración 65: Instalar Eclipse 2

Paso 3: Pulsaremos sobre DOWNLOADS.

Ilustración 66: Instalar Eclipse 3

Paso 4: Tras pulsar sobre DOWNLOADS se nos abrirá la siguiente página, una vez en ella elegimos la versión del sistema operativo que tengamos.

Ilustración 67: Instalar Eclipse 4

Paso 5: Cuando seleccionemos el sistema operativo se nos abrirá la siguiente página indicándonos lo que se descargara, para qué final mente no lo descarguemos pulsaremos sobre DOWNLOAD.

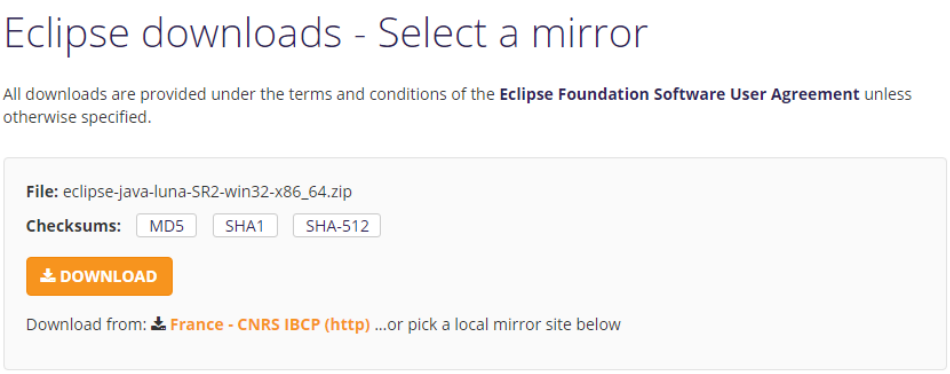
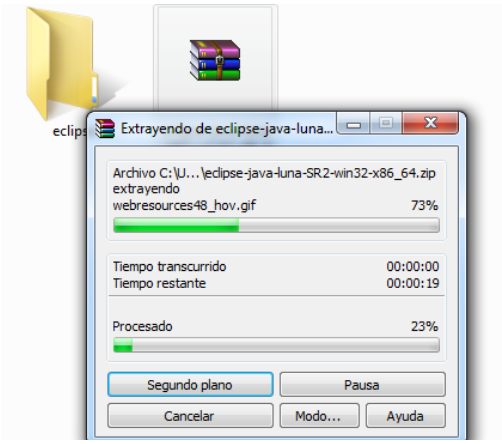


Ilustración 68: Instalar Eclipse 5

Paso 6: Para finalizar descomprimiremos el archivo descargado y ya tendremos eclipse descargado e instalado.



10.1.4 Instalar el conjunto ADT

Una vez tengamos instalado Eclipse instalaremos el conjunto ADT para Eclipse, para ello seguiremos los siguientes pasos:

Paso 1: Nos dirigiremos a la siguiente dirección web [23].

Paso 2: Una vez en ella copiaremos el siguiente texto:

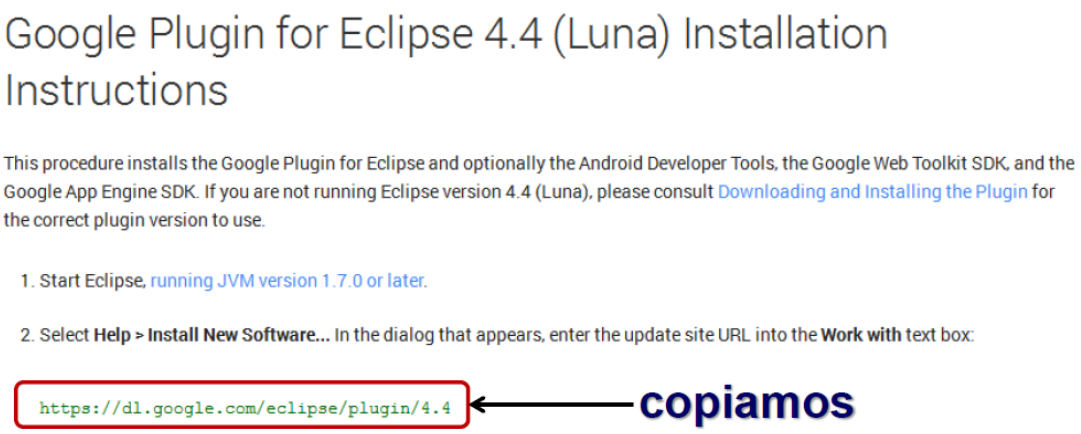


Ilustración 69: Instalar el conjunto ADT 1

Paso 3: Tras copiar el texto iniciaremos eclipse y seleccionaremos help→Install New Software.

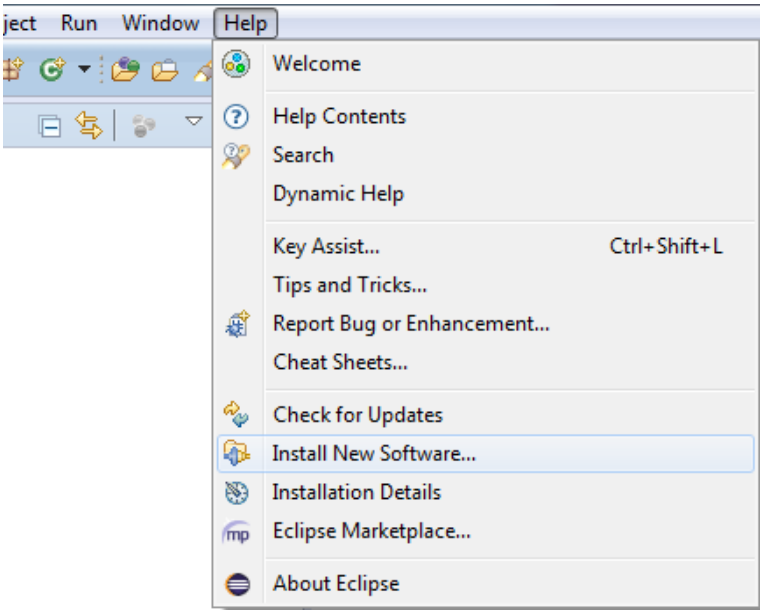


Ilustración 70: Instalar el conjunto ADT 2

Paso 4: Se abrirá la siguiente ventana donde pegaremos el texto copiado anteriormente y pulsaremos sobre Add:

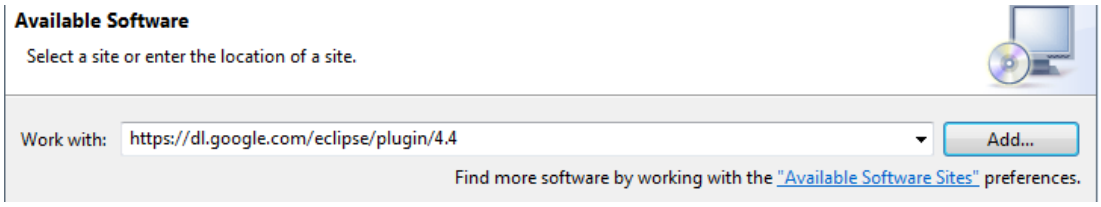


Ilustración 71: Instalar el conjunto ADT 3

Paso 5: Nos saldrá una nueva ventana donde tendremos que especificar el nombre.

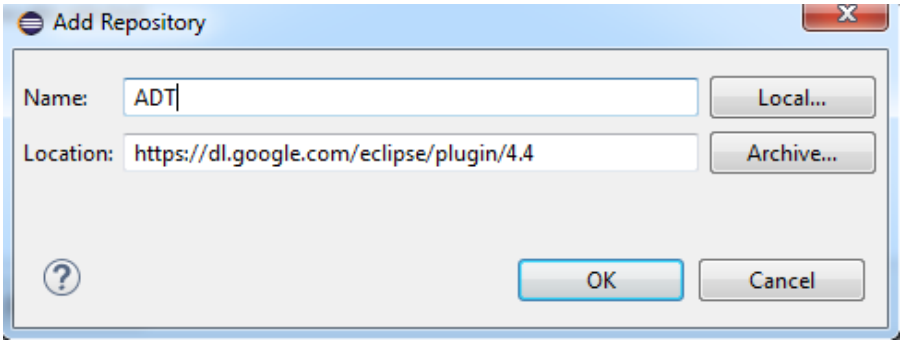


Ilustración 72: Instalar el conjunto ADT 4

Paso 6: Una vez puesto el nombre seleccionamos las herramientas a utilizar y pulsaremos sobre Next.

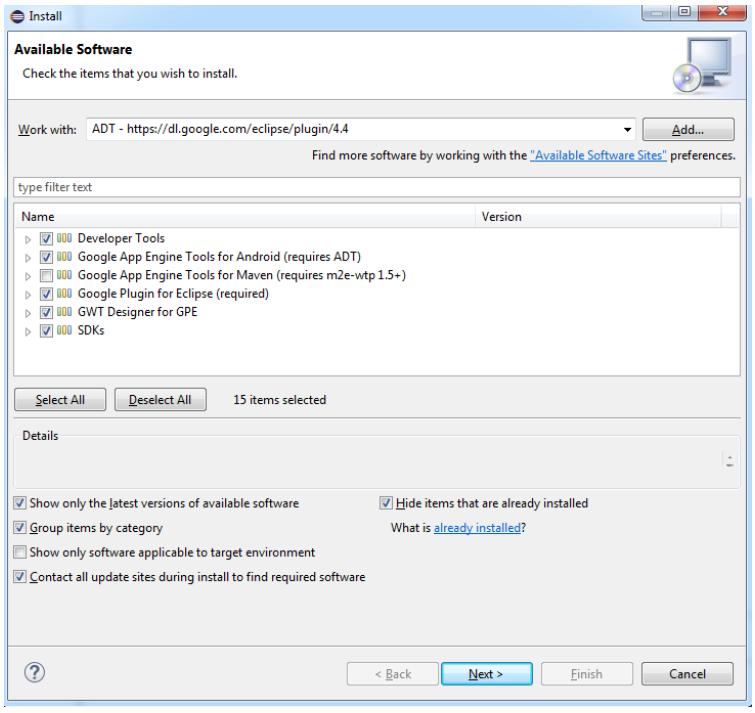


Ilustración 73: Instalar el conjunto ADT 5

Paso 7: Tras pulsar Next nos indicara lo que vamos a instalar, así que volvemos a pulsar Next.

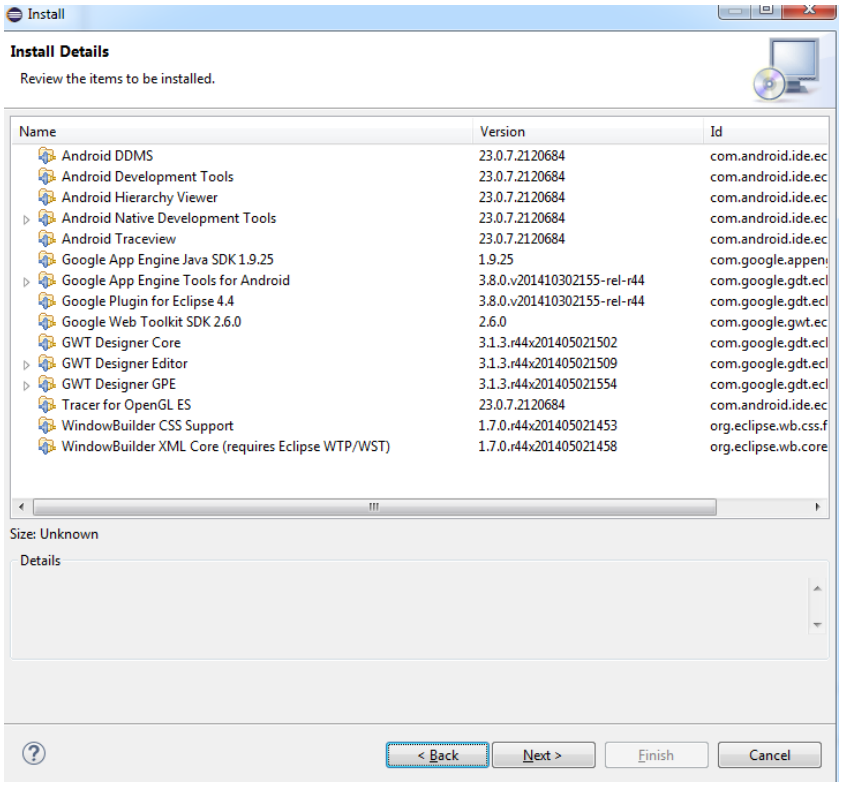


Ilustración 74: Instalar el conjunto ADT 6

Paso 8: Leemos y aceptamos los términos de licencia y pulsamos sobre Finish.

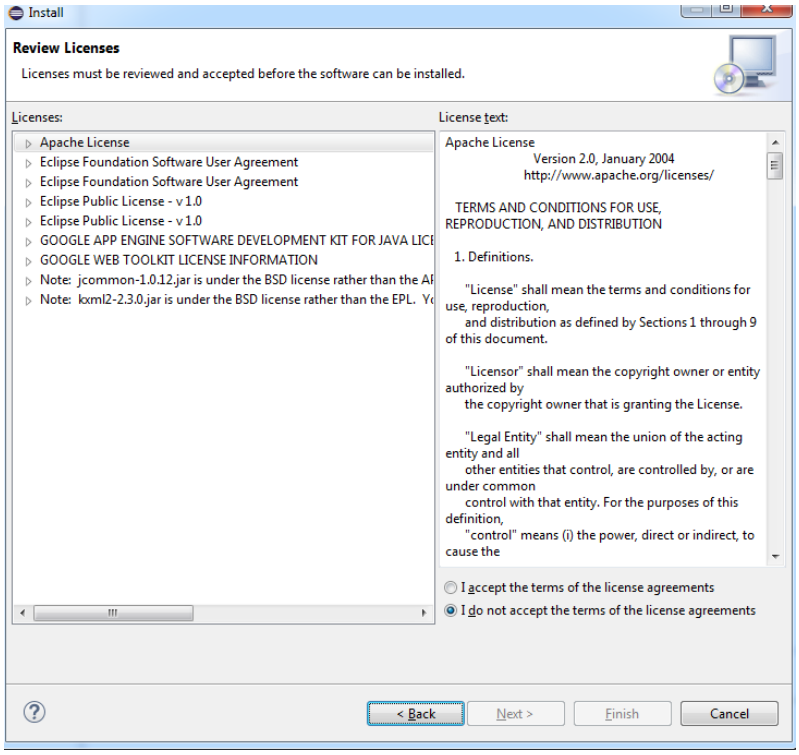


Ilustración 75: Instalar el conjunto ADT 7

Paso 9: Una vez pulsado Finish se procederá a la instalación del ADT.

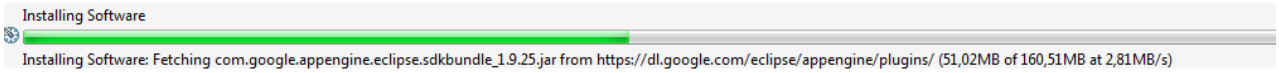


Ilustración 76: Instalar el conjunto ADT 8

Paso 10: Tras aceptar la advertencia y terminar la instalación reiniciamos Eclipse.

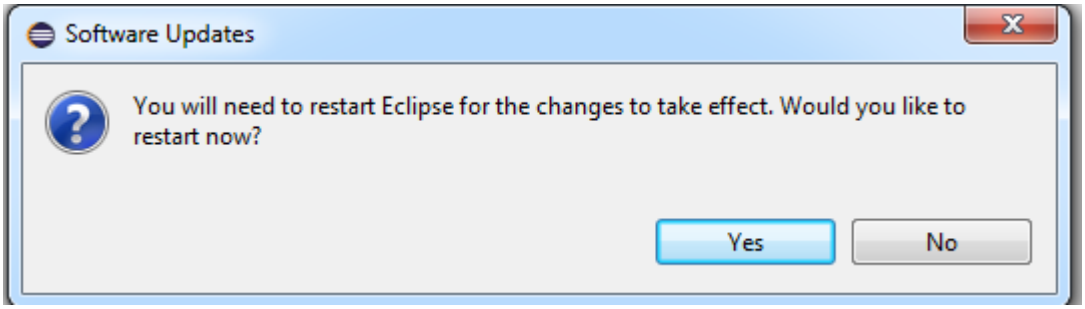


Ilustración 77: Instalar el conjunto ADT 9

10.1.5 Instalar SDK de Android

En este apartado se mostrará como descargar e instalar el SDK de Android e iniciar el emulador, para ello los pasos a seguir son los siguientes:

Paso 1: Tras reiniciar Eclipse nos dirigiremos a Windows→Android SDK Manager

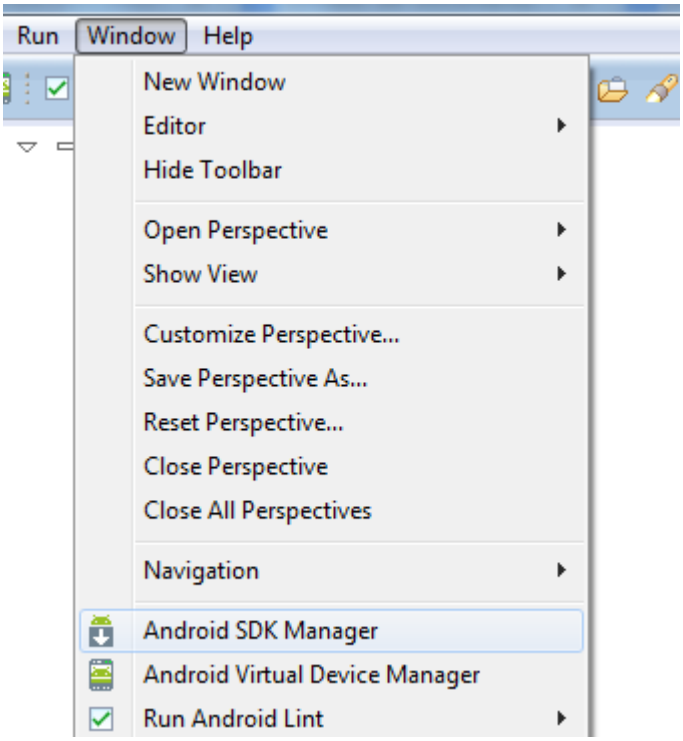


Ilustración 78: Instalar SDK de Android 1

Paso 2: Instalamos los Android SDK Build Tools, para ello seleccionamos la carpeta Tools.

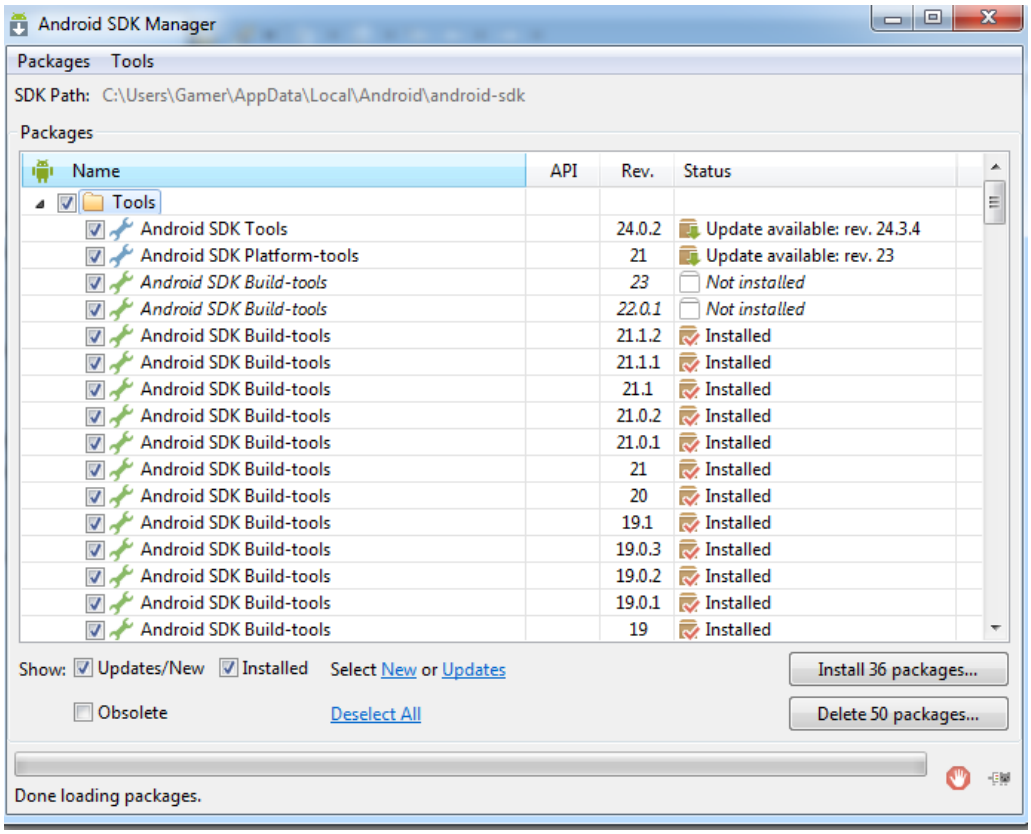


Ilustración 79: Instalar SDK de Android 2

Paso 3: Instalamos la licencia y procedemos a su instalación.

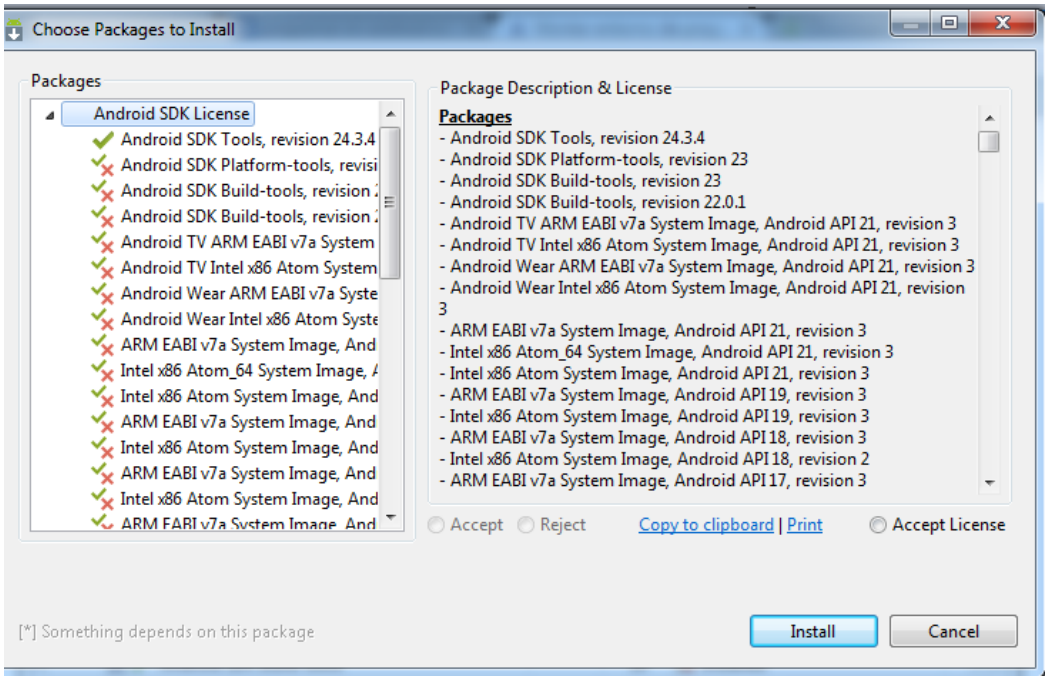


Ilustración 80: Instalar SDK de Android 3

10.1.6 Crear e iniciar el emulador

Tras instalar todo lo anterior procedemos a crear e iniciar un emulador de Android (Versión en un ordenador del Smartphone usada para realizar pruebas de la aplicación de manera rápida y fiable), para ello seguiremos los siguientes pasos:

Paso 1: Nos dirigiremos a Windows→Android Virtual Device Manager.

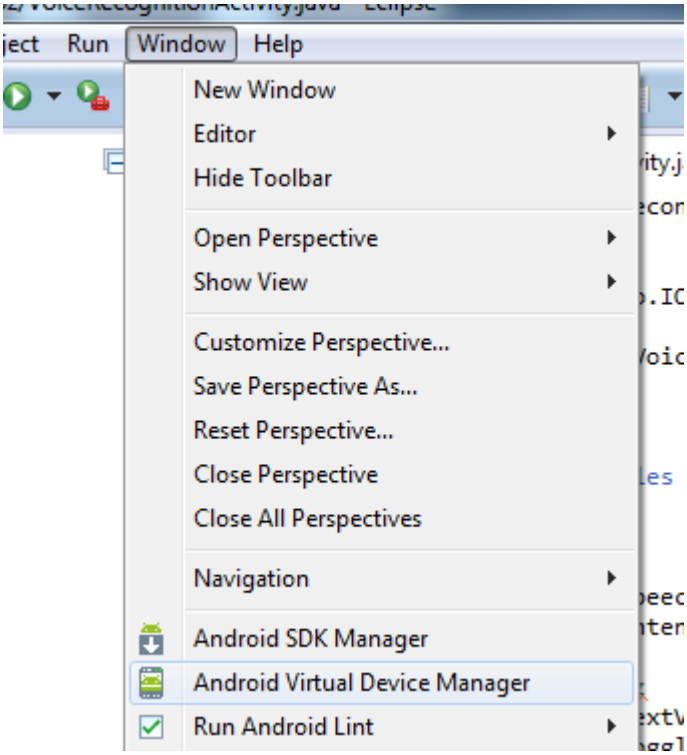


Ilustración 81: Crear el emulador 1

Paso 2: Pulsaremos sobre créate para crearnos uno nuevo.

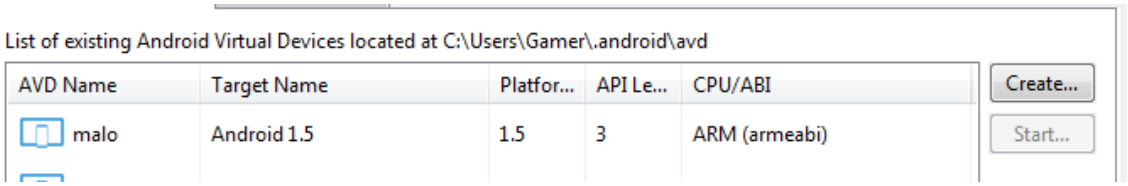


Ilustración 82: Crear el emulador 2

Paso 3: Configurar y crear el emulador.

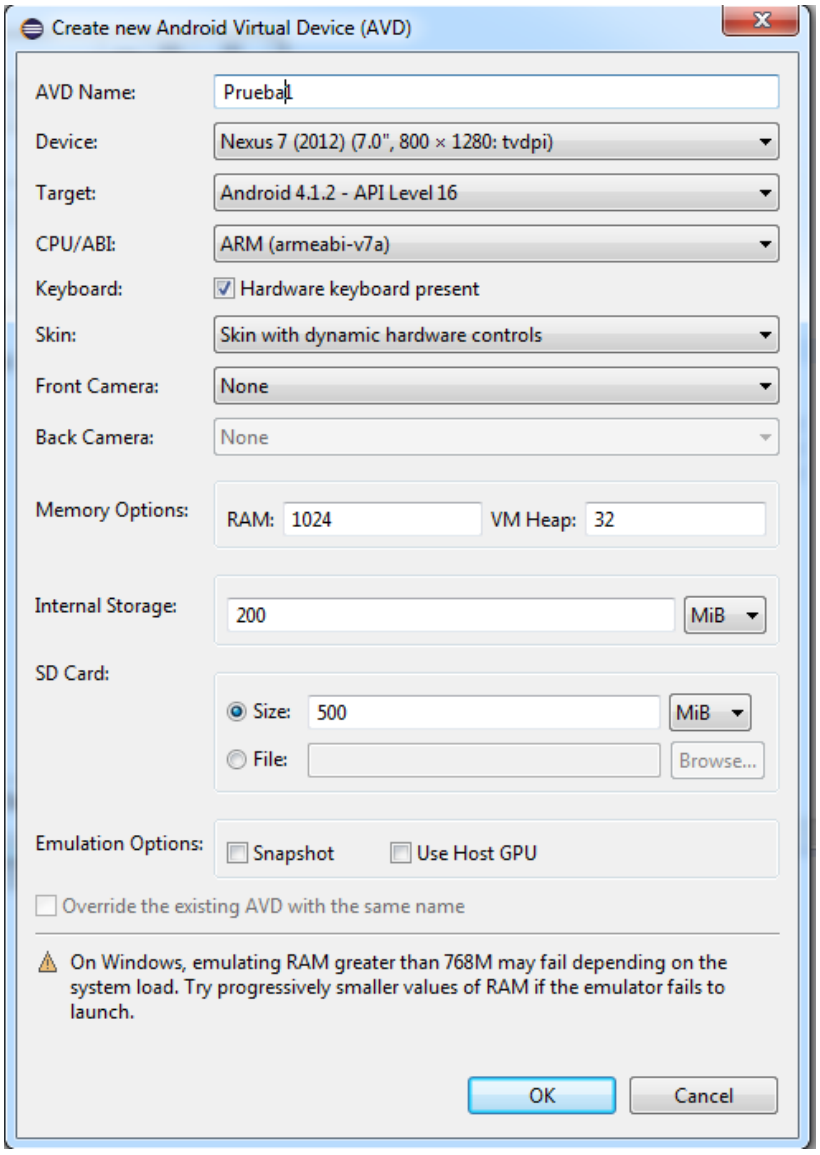


Ilustración 83: Crear el emulador 3

Paso 4: Para iniciarlo seleccionamos el dispositivo y presionamos sobre Start…

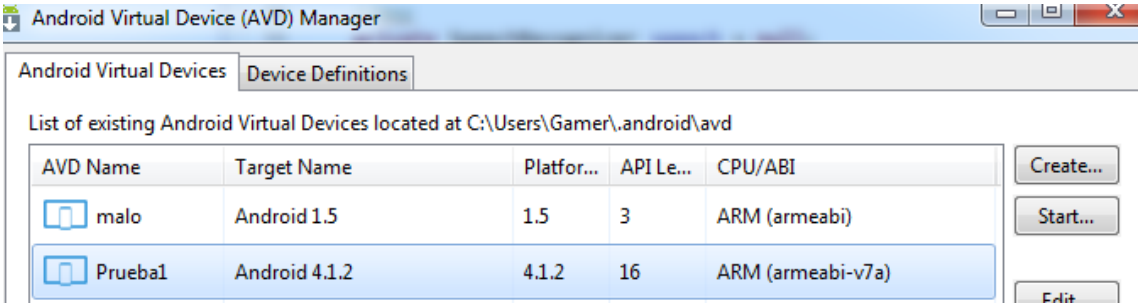


Ilustración 84: Iniciar el emulador 1

Paso 5: Esperar a que se inicie.

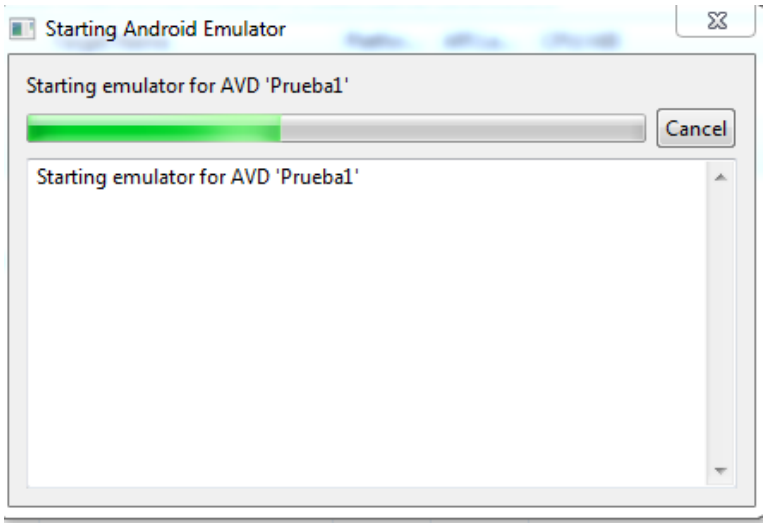


Ilustración 85: Iniciar el emulador 2

Paso 6: Una vez iniciado ya está listo para ser usado e instalar las aplicaciones en formato APK en él.

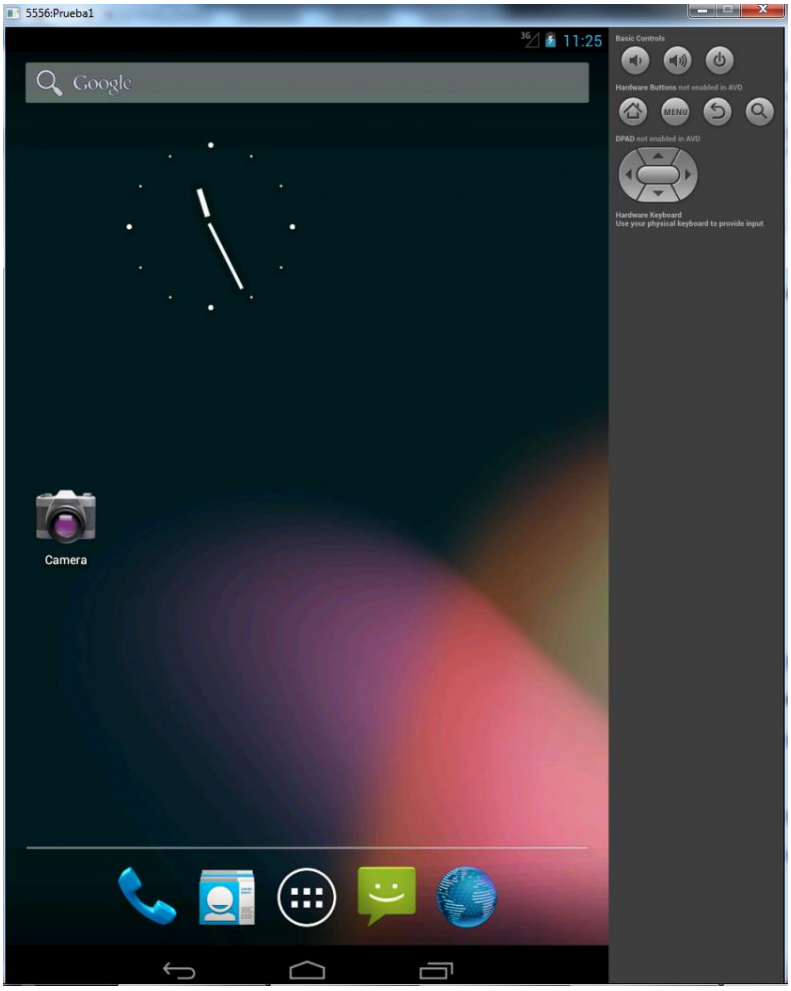


Ilustración 86: Emulador iniciado

10.1.7 Instalación del entorno de programación para Arduino

En esta apartado se indicaran los pasos a seguir para la descarga e instalación del entorno de programación para Arduino. Para ello iremos a la página oficial de Arduino [3] y se seguirán los siguientes pasos:

Paso 1: Pulsar sobre Download.

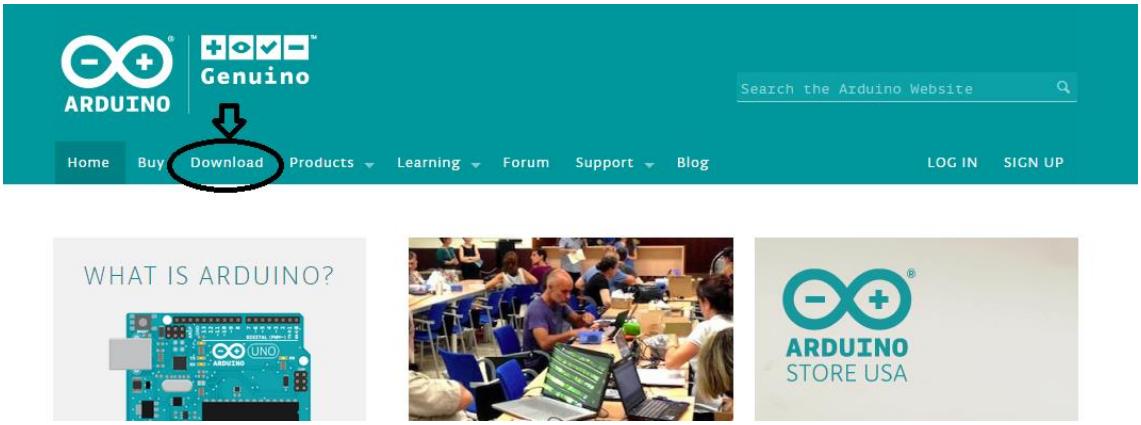


Ilustración 87: Instalación del entorno de programación para Arduino 1

Paso 2: Descargar según el sistema operativo.

Ilustración 88: Instalación del entorno de programación para Arduino 2

Paso 3: Una vez descargado lo iniciamos para comprobar que la instalación fue la correcta.

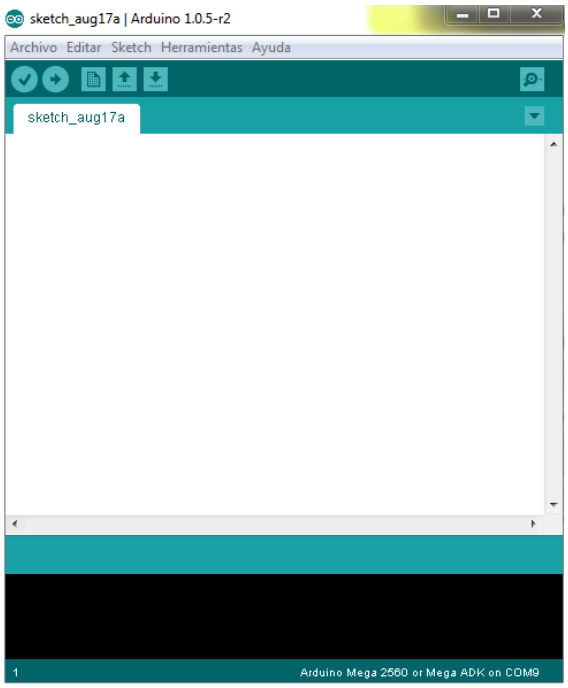


Ilustración 89: Instalación del entorno de programación para Arduino 3

Paso 4: Lo configuramos para nuestro arduino, en este caso para Arduino Mega.

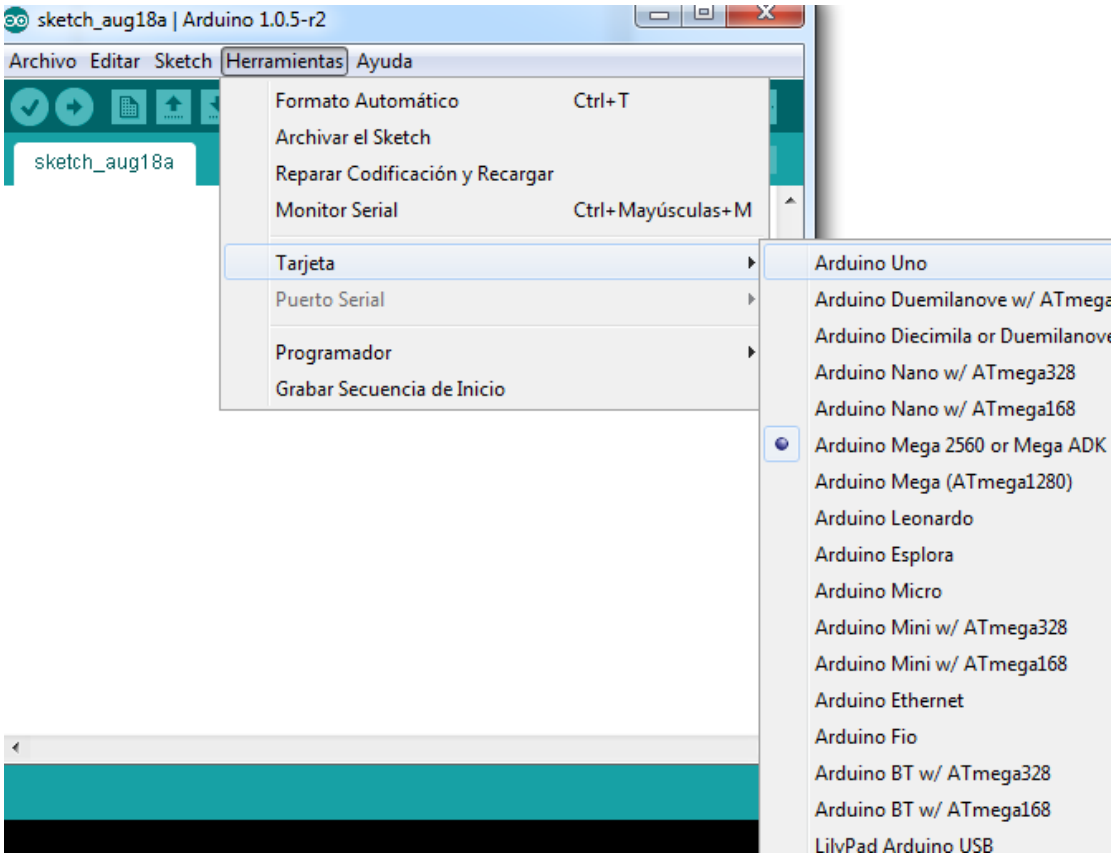


Ilustración 90: Instalación del entorno de programación para Arduino 4

Paso 5: Le indicamos el puerto serial donde se encuentra nuestro dispositivo.

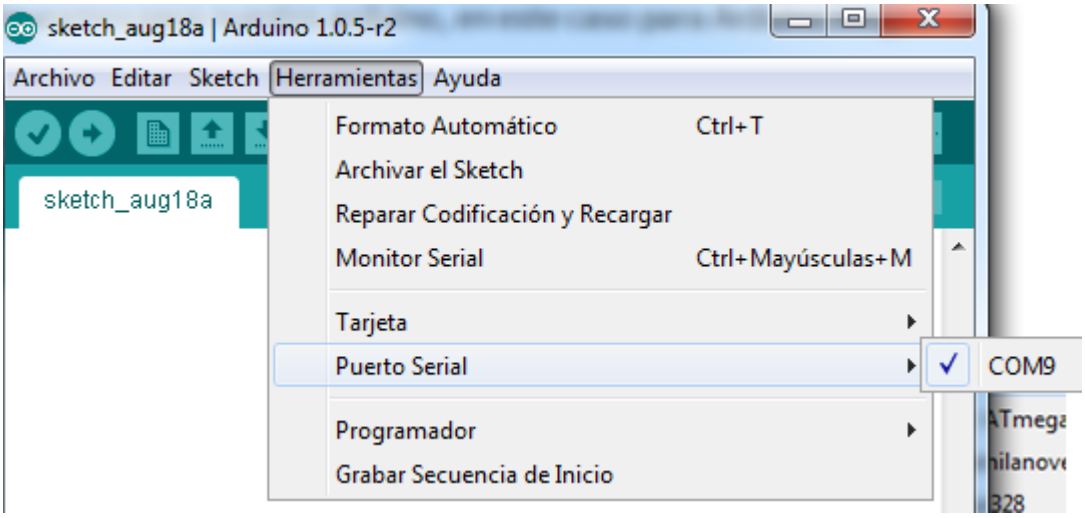


Ilustración 91Instalación del entorno de programación para Arduino 5

Una vez realizado estos 5 pasos ya tendremos nuestro entorno de programación para Arduino listo para poder empezar a trabajar.

10.2 Anexo I: Manual del Chip L293B

SGS-THOMSON MICROELECTRONICS

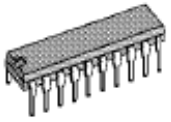
L293B
L293E

PUSH-PULL FOUR CHANNEL DRIVERS

- OUTPUT CURRENT 1A PER CHANNEL
- PEAK OUTPUT CURRENT 2A PER CHANNEL (non repetitive)
- INHIBIT FACILITY
- HIGH NOISE IMMUNITY
- SEPARATE LOGIC SUPPLY
- OVERTEMPERATURE PROTECTION

DIP16

ORDERING NUMBER : L293B



POWERDIP (16 + 2+ 2)

ORDERING NUMBER : L293E

DESCRIPTION

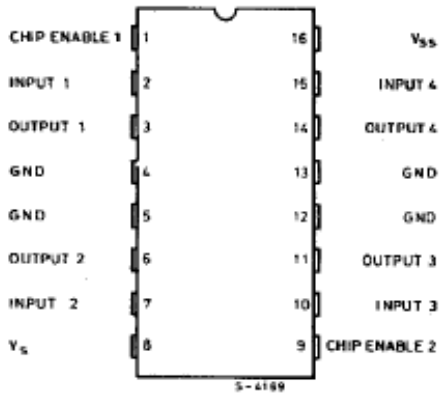
The L293B and L293E are quad push-pull drivers capable of delivering output currents to 1A per channel. Each channel is controlled by a TTL-compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off all four transistors. A separate supply input is provided for the logic so that it may be run off a lower voltage to reduce dissipation.

Additionally, the L293E has external connection of sensing resistors, for switchmode control.

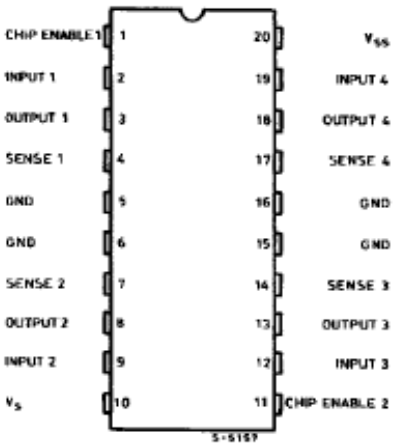
The L293B and L293E are package in 16 and 20-pin plastic DIPs respectively ; both use the four center pins to conduct heat to the printed circuit board.

PIN CONNECTIONS

DIP16 - L293B

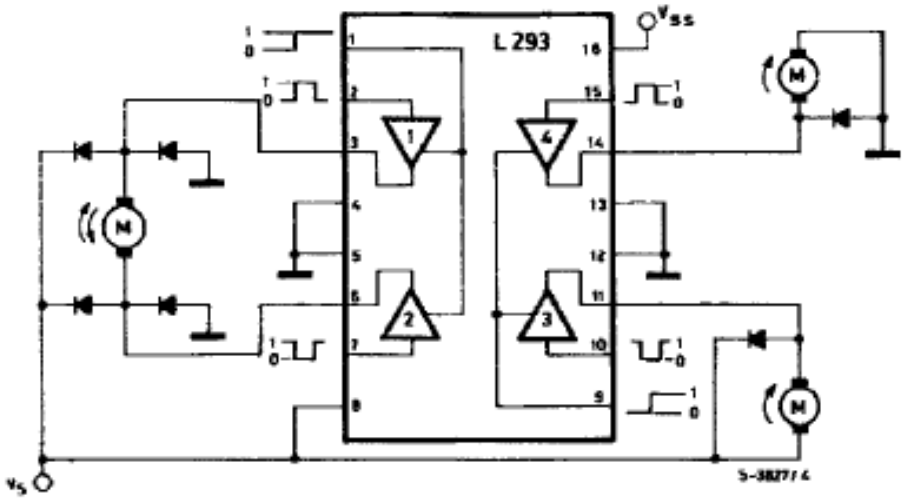


POWERDIP (16+2+2) - L293E

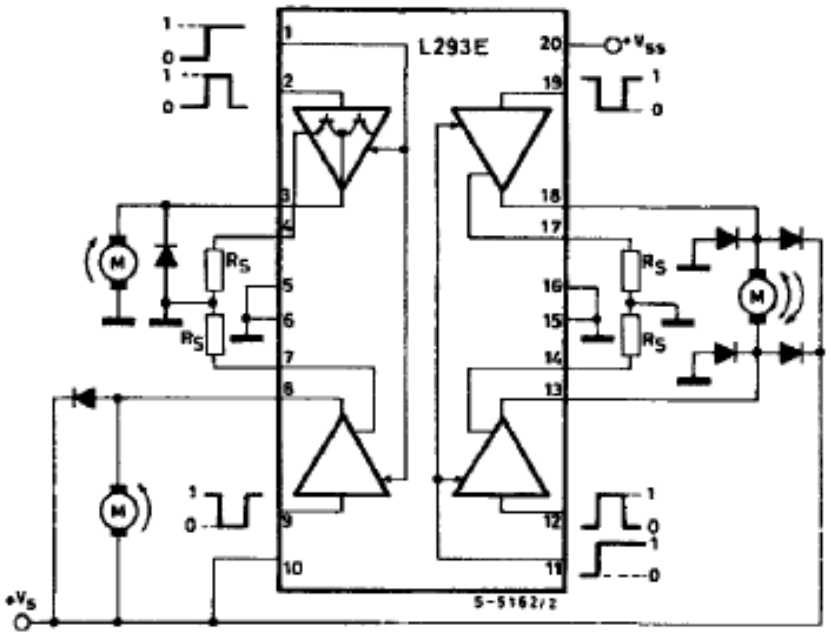


BLOCK DIAGRAMS

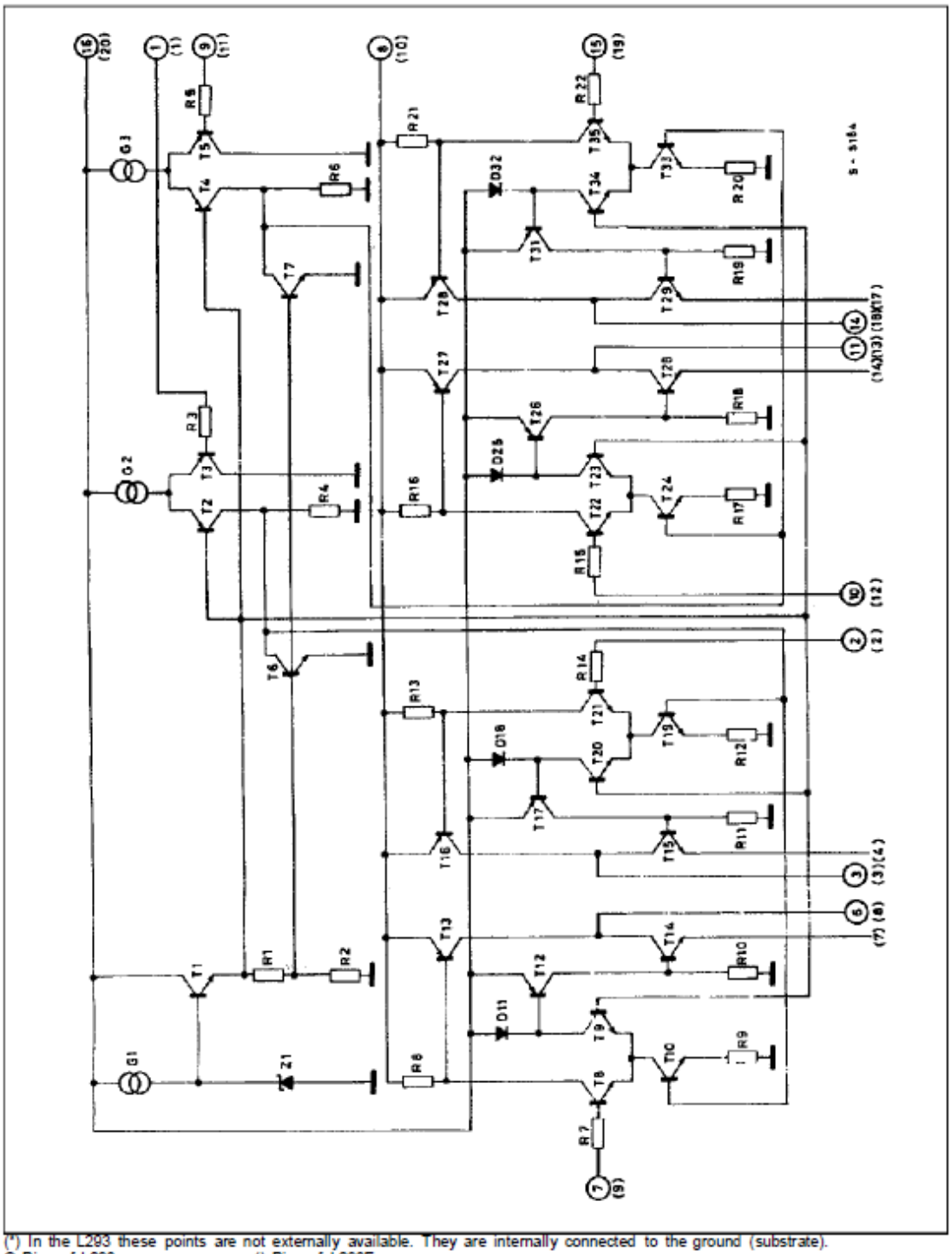
DIP16 - L293B



POWERDIP (16+2+2) - L293E



SCHEMATIC DIAGRAM



(*) In the L293 these points are not externally available. They are internally connected to the ground (substrate).
O Pins of L293 () Pins of L293E.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply Voltage	36	V
V_{ss}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{inh}	Inhibit Voltage	7	V
I_{out}	Peak Output Current (non repetitive t = 5ms)	2	A
P_{tot}	Total Power Dissipation at $T_{ground-pins}$ = 80°C	5	W
T_{stg}, T_j	Storage and Junction Temperature	–40 to +150	°C

THERMAL DATA

Symbol	Parameter		Value	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	80	°C/W

ELECTRICAL CHARACTERISTICS

For each channel, V_S = 24V, V_{SS} = 5V, T_{amb} = 25°C, unless otherwise specified

Symbol	Parameter	Test Conditions	Min.	TYp.	Max.	Unit
V_s	Supply Voltage		V_{ss}		36	V
V_{ss}	Logic Supply Voltage		4.5		36	V
I_s	Total Quiescent Supply Current	V_i = L I_o = 0 V_{inh} = H V_i = H I_o = 0 V_{inh} = H V_{inh} = L		2 16	6 24 4	mA
I_{ss}	Total Quiescent Logic Supply Current	V_i = L I_o = 0 V_{inh} = H V_i = H I_o = 0 V_{inh} = H V_{inh} = L		44 16 16	60 22 24	mA
V_{IL}	Input Low Voltage		-03.		1.5	V
V_{IH}	Input High Voltage	$V_{SS} \leq 7V$ $V_{ss} > 7V$	2.3 2.3		V_{ss} 7	V
I_{IL}	Low Voltage Input Current	V_{il} = 1.5V			-10	µA
I_{IH}	High Voltage Input Current	$2.3V \leq V_{IH} \leq V_{ss} - 0.6V$		30	100	µA
V_{InhL}	Inhibit Low Voltage		-0.3		1.5	V
V_{InhH}	Inhibit High Voltage	$V_{SS} \leq 7V$ $V_{ss} > 7V$	2.3 2.3		V_{ss} 7	V
I_{InhL}	Low Voltage Inhibit Current	V_{InhL} = 1.5V		-30	-100	µA
I_{InhH}	High Voltage Inhibit Current	$2.3V \leq V_{InhH} \leq V_{ss} - 0.6V$			±10	µA
V_{CEsatH}	Source Output Saturation Voltage	I_o = -1A		1.4	1.8	V
V_{CEsatL}	Sink Output Saturation Voltage	I_o = 1A		1.2	1.8	V
V_{GEN9}	Sensing Voltage (pins 4, 7, 14, 17) (**)				2	V
t_r	Rise Time	0.1 to 0.9 V_o (*)		250		ns
t_f	Fall Time	0.9 to 0.1 V_o (*)		250		ns
t_{on}	Turn-on Delay	0.5 V_i to 0.5 V_o (*)		750		ns
t_{off}	Turn-off Delay	0.5 V_i to 0.5 V_o (*)		200		ns

* See figure 1
** Referred to L293E

TRUTH TABLE

V_I (each channel)	V_o	V_{inh} (–)
H	H	H
L	L	H
H	X (°)	L
L	X (°)	L

(*) High output impedance
(**) Relative to the considerate channel

Figure 1 : Switching Timers

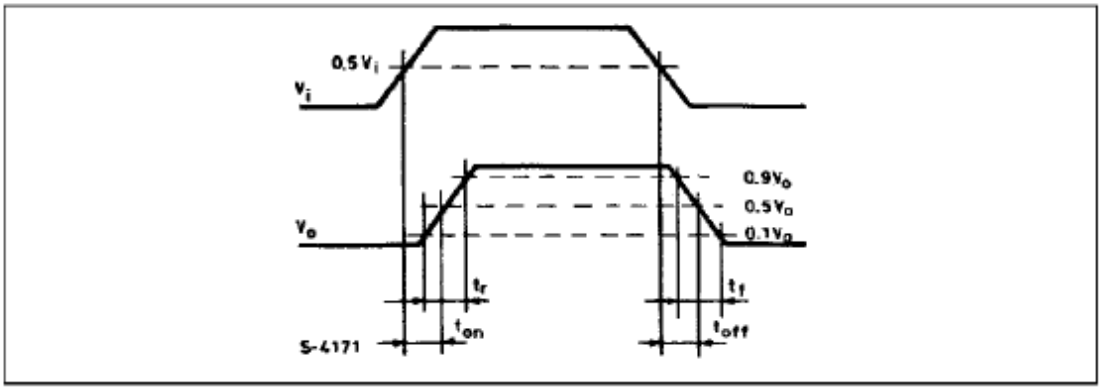


Figure 2 : Saturation voltage versus Output Current

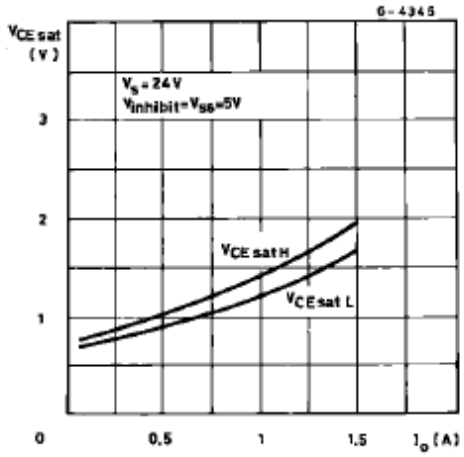


Figure 3 : Source Saturation Voltage versus Ambient Temperature

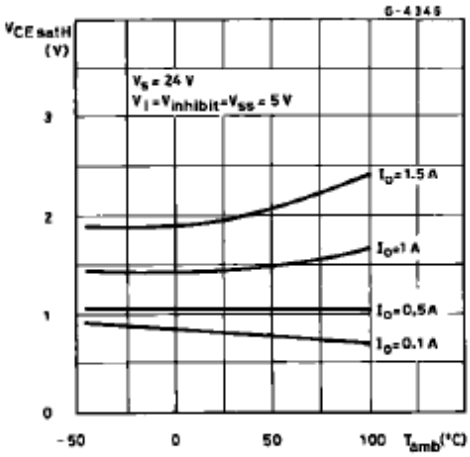


Figure 4 : Sink Saturation Voltage versus Ambient Temperature

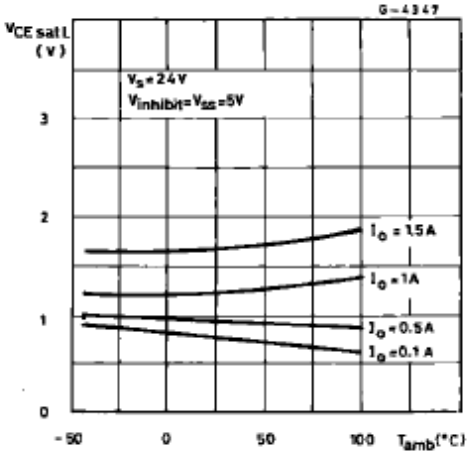


Figure 5 : Quiescent Logic Supply Current versus Logic Supply Voltage

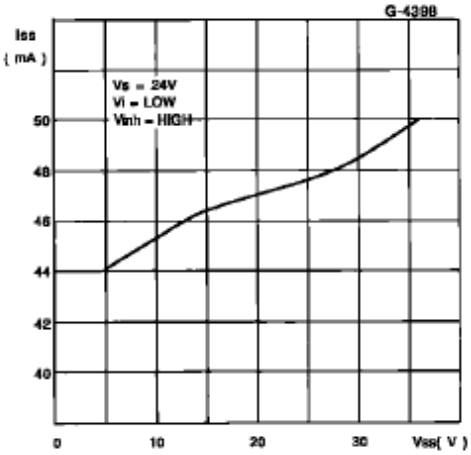


Figure 6 : Output Voltage versus Input Voltage

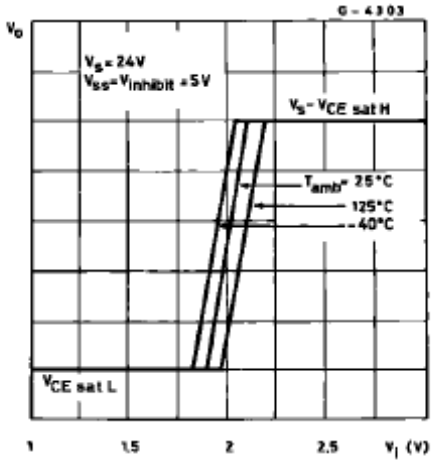
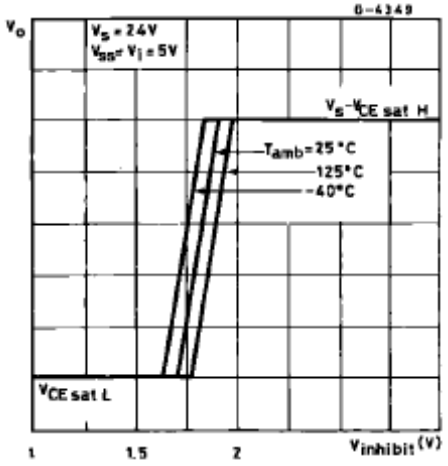
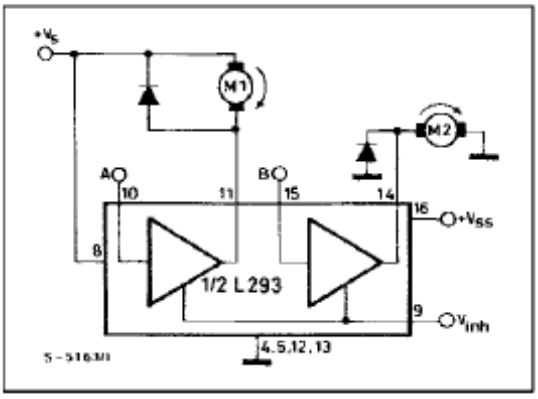


Figure 7 : Output Voltage versus Inhibit Voltage



APPLICATION INFORMATION

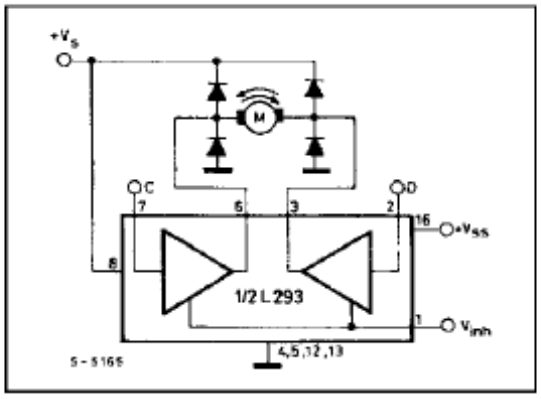
Figure 8 : DC Motor Controls (with connection to ground and to the supply voltage)



V_{inh}	A	M1	B	M2
H	H	Fast Motor Stop	H	Run
H	L	Run	L	Fast Motor Stop
L	X	Free Running Motor Stop	X	Free Running Motor Stop

L = Low H = High X = Don't Care

Figure 9 : Bidirectional DC Motor Control



Inputs	Function	
V_{inh} = H	C = H ; D = L	Turn Right
	C = L ; D = H	Turn Left
	C = D	Fast Motor Stop
V_{inh} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't Care

Figure 10 : Bipolar Stepping Motor Control

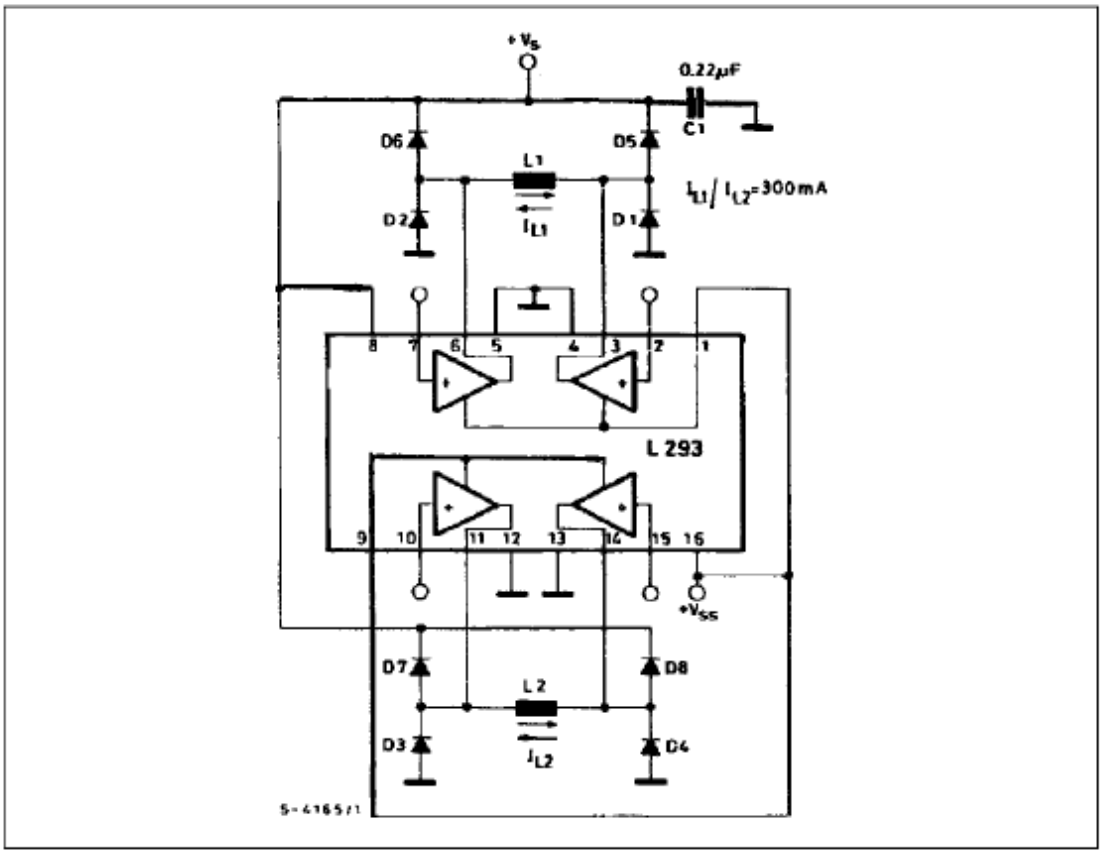
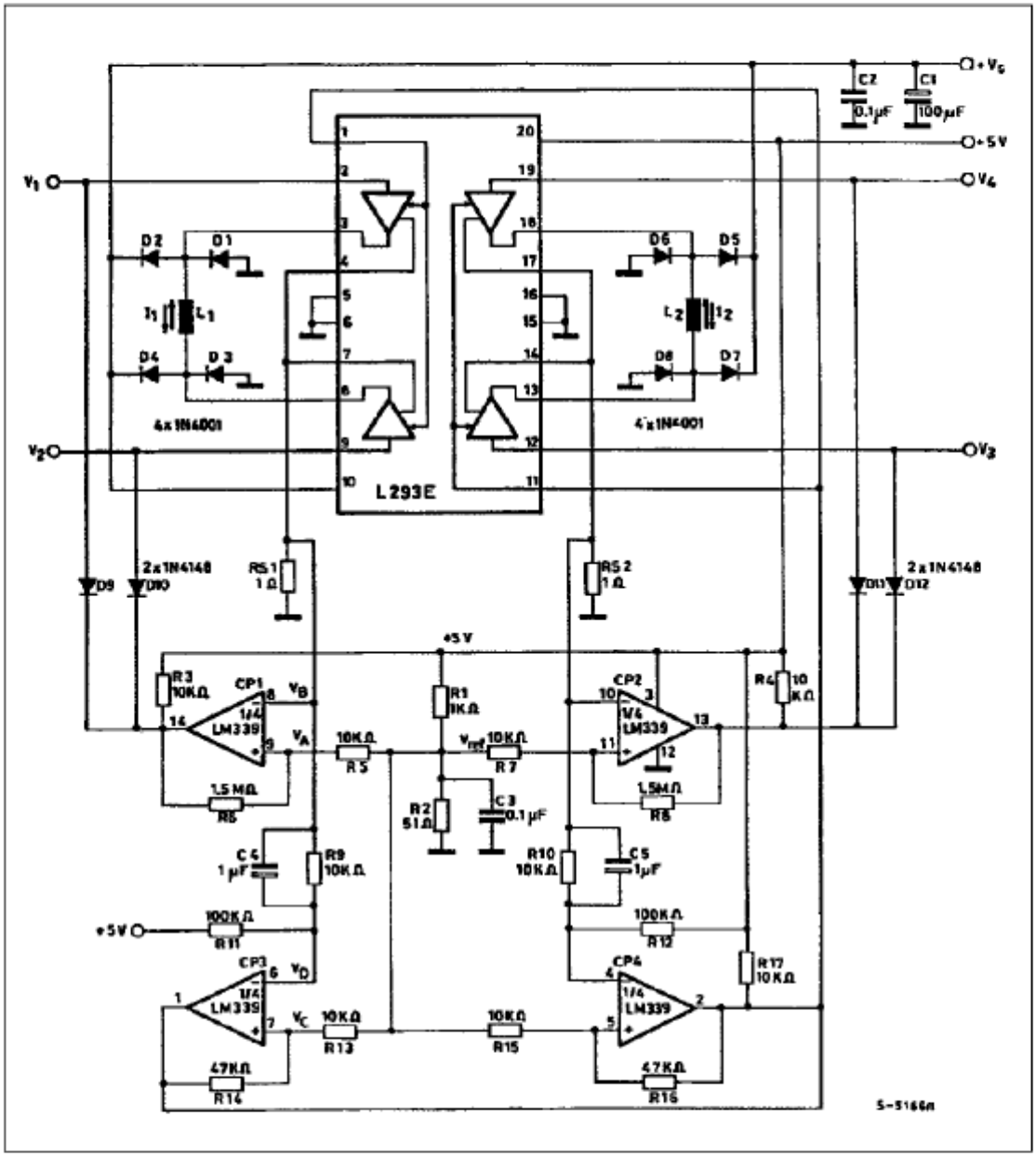


Figure 11 :Stepping Motor Driver with Phase Current Control and Short Circuit Protection



MOUNTING INSTRUCTIONS

The $R_{th\ j\text{-}amb}$ of the L293B and the L293E can be reduced by soldering the GND pins to a suitable copper area of the printed circuit board as shown in figure 12 or to an external heatsink (figure 13).

During soldering the pins temperature must not exceed 260°C and the soldering time must not be longer than 12 seconds.

The external heatsink or printed circuit copper area must be connected to electrical ground.

Figure 12 : Example of P.C. Board Copper Area which is Used as Heatsink

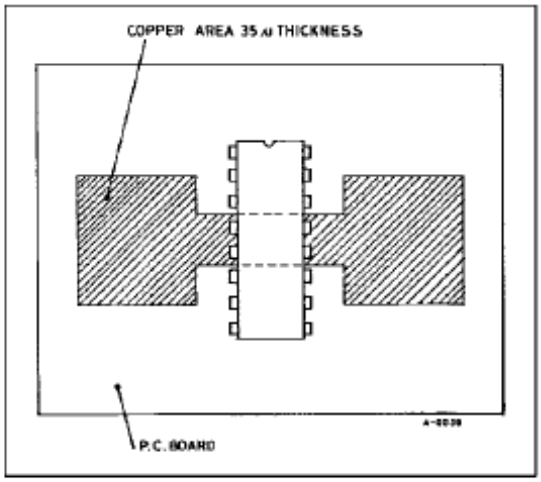
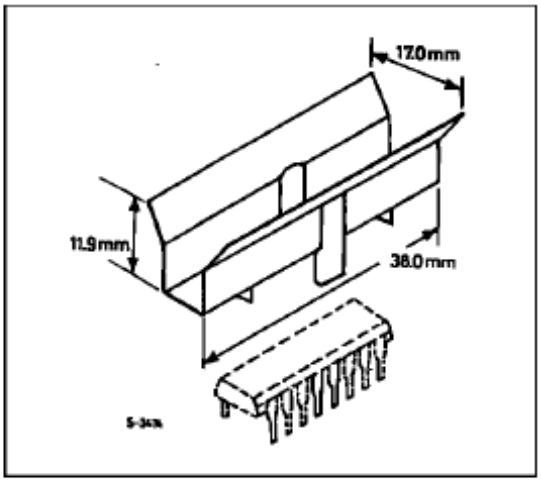


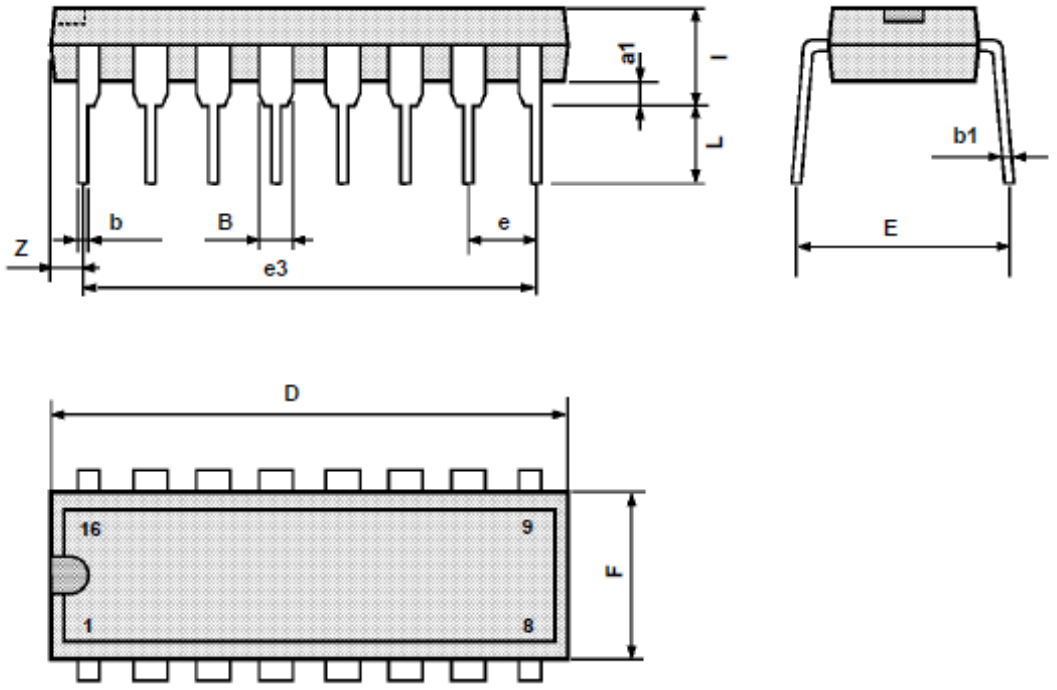
Figure 13 : External Heatsink Mounting Example (Rth = 30°C/W)



DIP16 PACKAGE MECHANICAL DATA

Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
a1	0.51			0.020		
B	0.77		1.65	0.030		0.065
b		0.5			0.020	
b1		0.25			0.010	
D			20			0.787
E		8.5			0.335	
e		2.54			0.100	
e3		17.78			0.700	
F			7.1			0.280
i			5.1			0.201
L		3.3			0.130	
Z			1.27			0.050

DIP16PWTBL

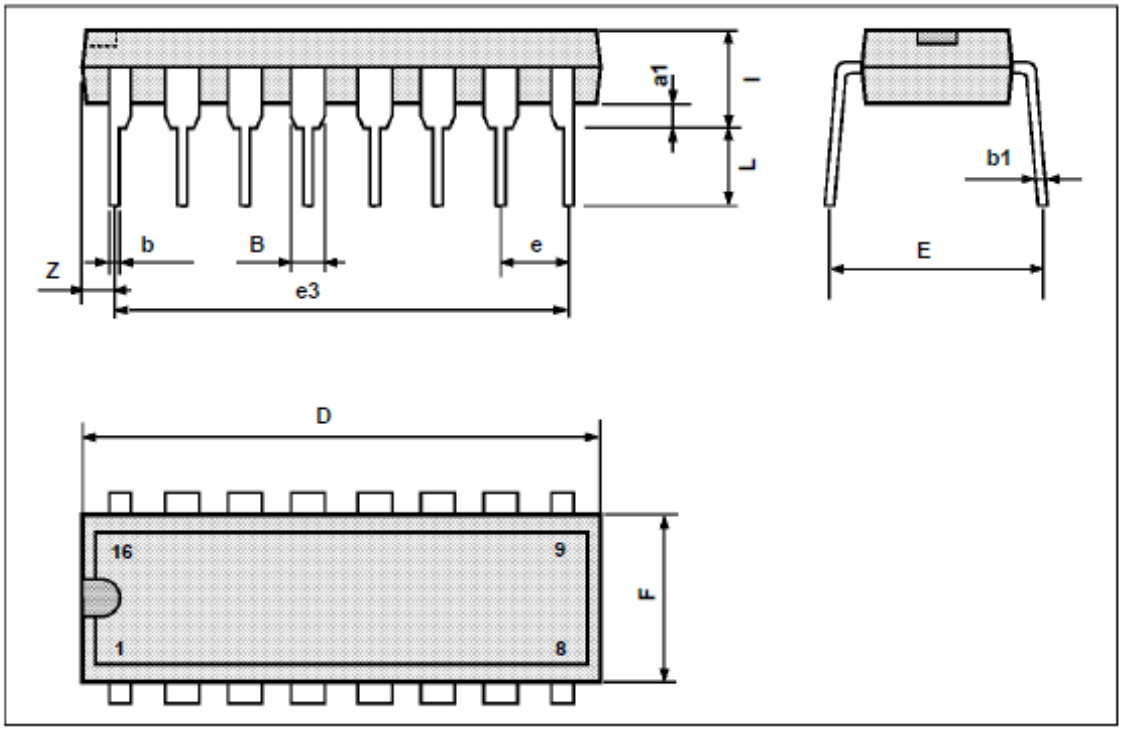


PMDIP16W.EPS

DIP16 PACKAGE MECHANICAL DATA

Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
a1	0.51			0.020		
B	0.77		1.65	0.030		0.065
b		0.5			0.020	
b1		0.25			0.010	
D			20			0.787
E		8.5			0.335	
e		2.54			0.100	
e3		17.78			0.700	
F			7.1			0.280
i			5.1			0.201
L		3.3			0.130	
Z			1.27			0.050

DIP16MWTBL

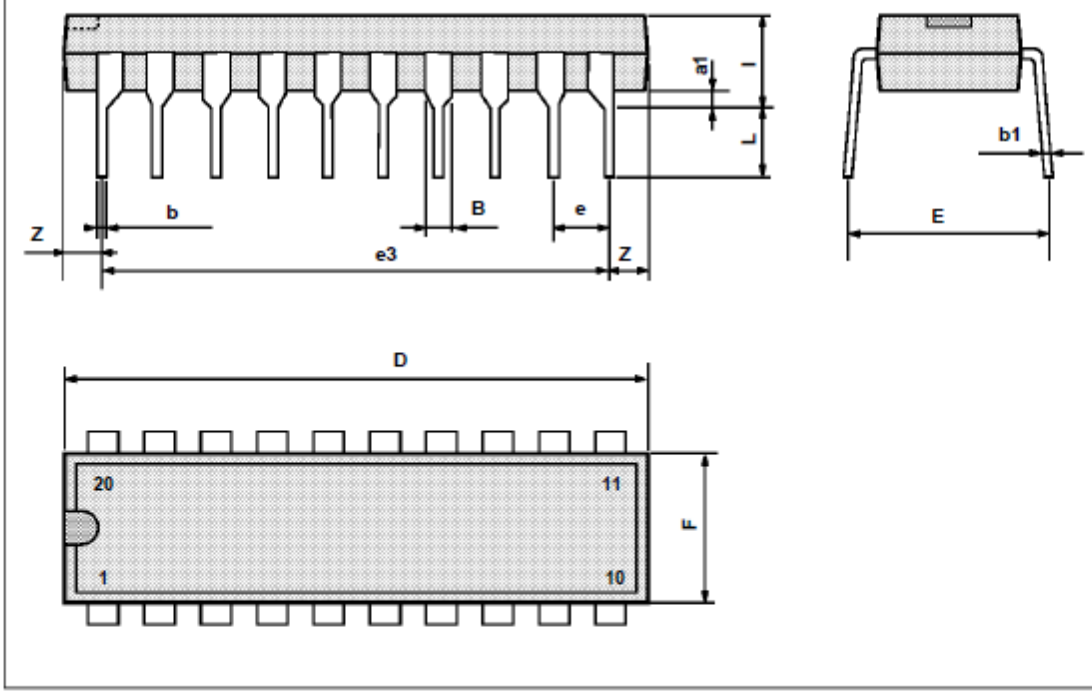


PMDIP16MEPS

POWERDIP (16+2+2) PACKAGE MECHANICAL DATA

Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
a1	0.51			0.020		
B	0.85		1.4	0.033		0.055
b		0.5			0.020	
b1	0.38		0.5	0.015		0.020
D			24.8			0.976
E		8.8			0.346	
e		2.54			0.100	
e3		22.86			0.900	
F			7.1			0.280
i			5.1			0.201
L		3.3			0.130	
Z			1.27			0.050

DIP20PN.TBL



PMDIP20WEPS

10.3 Anexo II: Manual del módulo ultrasónico HC-SR04

www.elecfreaks.com

HC-SR04 User Guide

Part 1 Ultrasonic Introduction

1. 1 Ultrasonic Definition

The human ear can hear sound frequency around 20HZ ~ 20KHZ, and ultrasonic is the sound wave beyond the human ability of 20KHZ .

1.2 Ultrasonic distance measurement principle

Ultrasonic transmitter emitted an ultrasonic wave in one direction, and started timing when it launched. Ultrasonic spread in the air, and would return immediately when it encountered obstacles on the way. At last, the ultrasonic receiver would stop timing when it received the reflected wave. As Ultrasonic spread velocity is 340m / s in the air, based on the timer record **t**, we can calculate the distance (s) between the obstacle and transmitter, namely: s = 340t / 2, which is so- called time difference distance measurement principle

The principle of ultrasonic distance measurement used the already-known air spreading velocity, measuring the time from launch to reflection when it encountered obstacle, and then calculate the distance between the transmitter and the obstacle according to the time and the velocity. Thus, the principle of ultrasonic distance measurement is the same with radar.

Distance Measurement formula is expressed as: L = C X T

In the formula, L is the measured distance, and C is the ultrasonic spreading velocity in air, also, T represents time (T is half the time value from transmitting to receiving).

1.3 Ultrasonic Application

Ultrasonic Application Technology is the thing which developed in recent decades. With the ultrasonic advance, and the electronic technology development, especially as high-power semiconductor device technology matures, the application of ultrasonic has become increasingly widespread:

- Ultrasonic measurement of distance, depth and thickness;
- Ultrasonic testing;
- Ultrasound imaging;
- Ultrasonic machining, such as polishing, drilling;
- Ultrasonic cleaning;
- Ultrasonic welding;

Part 2 HC-SR04 Ultrasonic Module Introduction

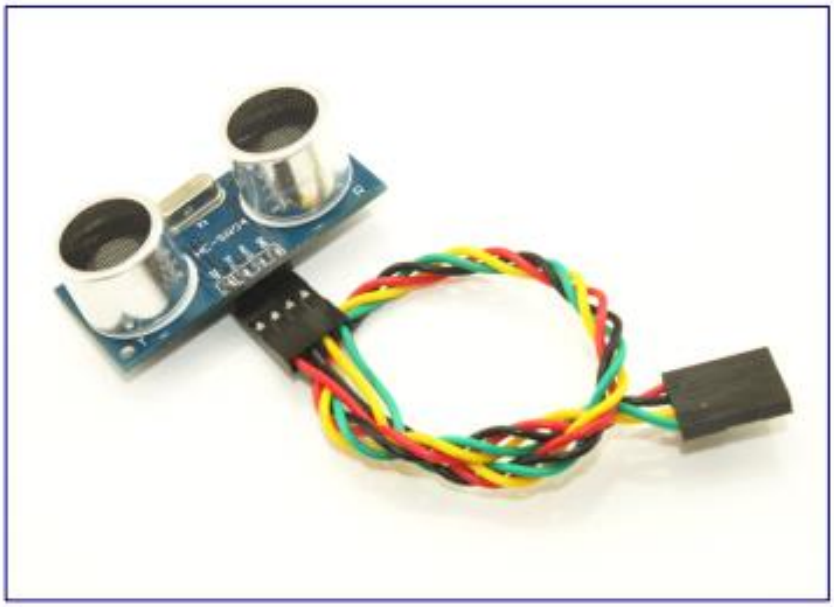
2.1 Product Features

- Stable performance
- Accurate distance measurement
- High-density
- Small blind

Application Areas:

- Robotics barrier
- Object distance measurement
- Level detection
- Public security
- Parking detection

2.2 Product Image



2.3、Module pin definitions

Types	Pin Symbol	Pin Function Description
HC-SR04	VCC	5V power supply
	Trig	Trigger pin
	Echo	Receive pin
	GND	Power ground

2.4、Electrical parameters

Electrical Parameters	HC-SR04 Ultrasonic Module
Operating Voltage	DC-5V
Operating Current	15mA
Operating Frequency	40KHZ
Farthest Range	4m
Nearest Range	2cm
Measuring Angle	15 Degree
Input Trigger Signal	10us TTL pulse
Output Echo Signal	Output TTL level signal, proportional with range
Dimensions	45*20*15mm

2.5 Module operating Principle

Set low the Trig and Echo port when the module initializes , firstly, transmit at least 10us high level pulse to the Trig pin (module automatically sends eight 40K square wave), and then wait to capture the rising edge output by echo port, at the same time, open the timer to start timing. Next, once again capture the falling edge output by echo port, at the same time, read the time of the counter, which is the ultrasonic running time in the air. According to the formular: test distance = (high level time * ultrasonic spreading velocity in air) / 2, you can calculate the distance to the obstacle.

10.4 Anexo III: Manual del módulo Bluetooth HC-06

Robotale

Bluetooth to serial HC-06 wireless module

Product Description:

1，Mainstream CSR Bluetooth chip, Bluetooth V2.0 protocol standards

2，serial module operating voltage 3.3V.

3，the baud rate for 1200, 2400, 4800, 9600, 19200, 38400 and users can be set

4，core module size: 28mm x 15 mm x 2.35mm.

5，the working current: 40mA

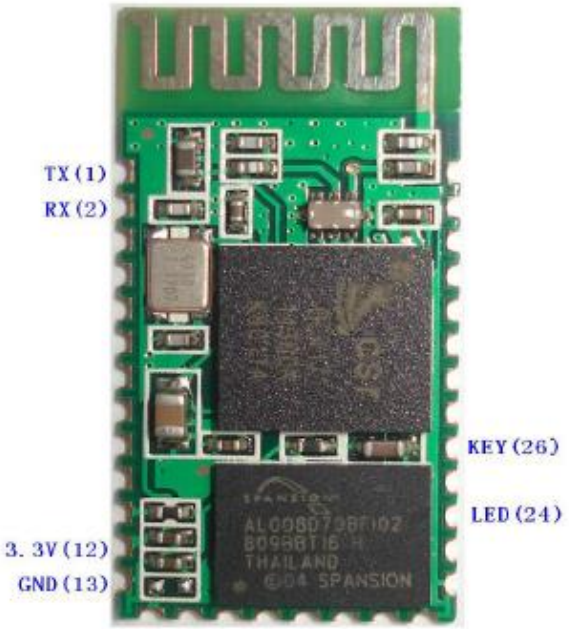
6，Sleep current: <1mA

7，for the GPS navigation system, utility meter reading system, the industrial field, collecting and controlling system.

8，with a Bluetooth laptop computer to the Bluetooth adapter, PDA and other devices to seamlessly connect

The module's host and slave, the host and slave pairing communication from the machine and from the machine or between the host and the host can not communicate, communication function and computers, mobile phones and other Bluetooth pairing purchase default slave , requires that the host needs to be indicated]

Main distinction: 1, if the chip is not specified on, the lights flash slow main fast from; September 2,2009, all manufactured host will be playing in the IC a hook or paste There are the "main" characters, there is no hook or not affixed to the word "master" is the slave. The date of manufacture can be obtained from the Bluetooth address]



The factory default parameters:

Slave, baud rate: 9600, n, 8,1. Passkey: 1234; need host mode, indicate when the orders.

Second, AT command set as follows:

1, test communications

Send: AT (return OK, one second left and right)

Back: OK

2, change the Bluetooth serial communication baud rate

Send: AT + BAUD1

Back to: OK1200

Send: AT + BAUD2

Back to: OK2400

.....

1 --------- 1200

2 --------- 2400

3 --------- 4800

4 --------- 9600

5 --------- 19200

6 --------- 38400

7 --------- 57600

8 --------- 115200

9 --------- 230400

A --------- 460800

B --------- 921600

C --------- 1382400

Not recommended to use more than 115200 baud rate, signal interference causes the system to instability.

The settings over 115,200 with a computer is not available, use microcontroller programming in higher than 115200 to use this baud rate and re-issue the AT command set low baud rate

AT command set baud rate, the next power do not need to set up and can be powered down to save the baud rate.

3, change the Bluetooth name

Send: AT + NAMEname

Back to: OKname

Parameter name: To set the current name, the name of the Bluetooth search. 20 characters or less.

Example: Sending AT + NAMEbill_gates

Back OKname

The Bluetooth name changed to bill_gates

The parameters can be powered down to save, simply modify the time. PDA the end refresh can see the Bluetooth name changed.

4, change the Bluetooth pairing password

Send: AT + PINxxxx

Returns: OKsetpin

Parameter xxxx: To set a passcode, 4 bytes, this command can be used from the machine or host. The slave adapter or mobile phone pops up to enter when pairing the password window, manually enter this parameter can be connected from the machine. Host in the main Bluetooth module connected digital camera, digital camera from the machine, find the camera pairing password, and then set up the White Bluetooth module, the main Bluetooth module can automatically connect the camera.

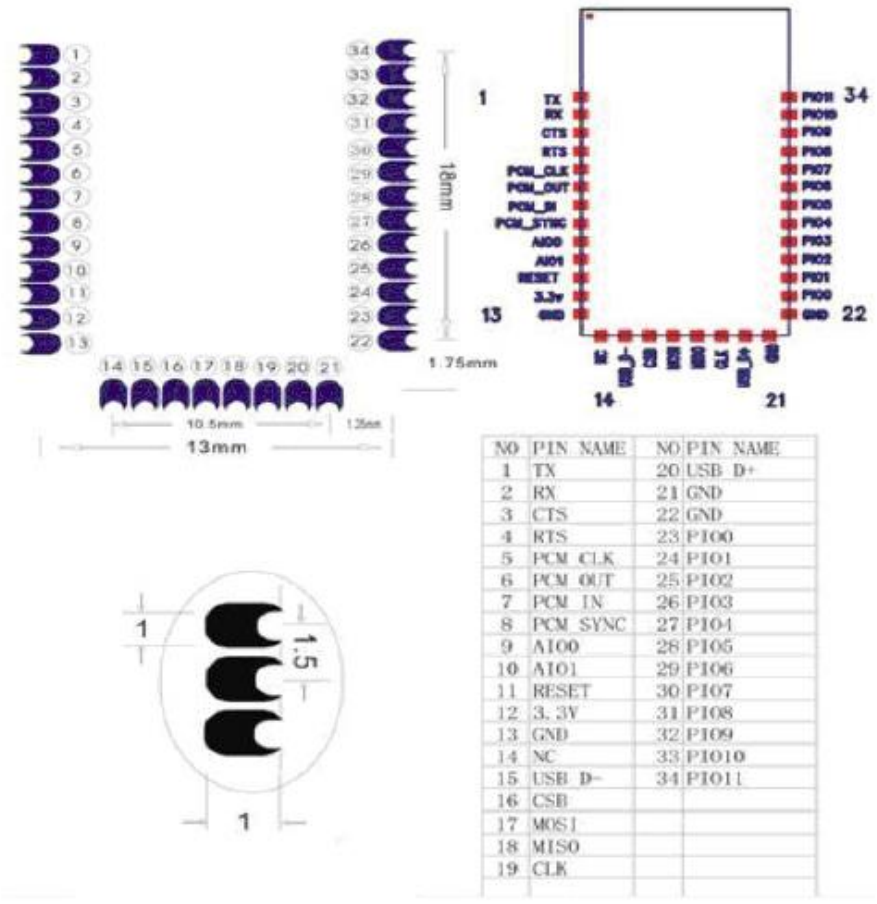
Example: Sending AT + PIN8888

Back OKsetpin

The Bluetooth pairing password to 8888, module paired at the factory default password is 1234.

The parameters can be powered down to save, simply modify the time.

Package pin diagram:

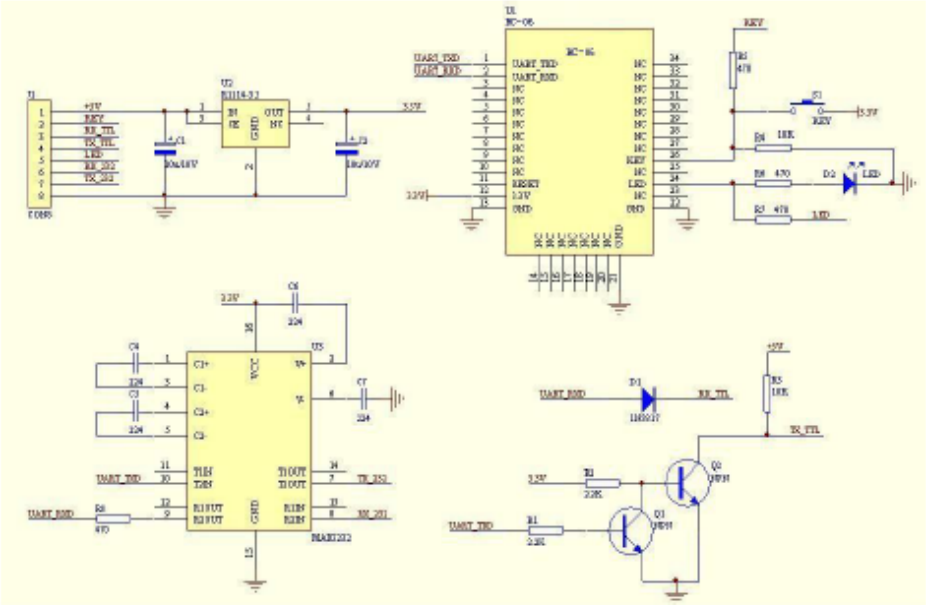


NO	PIN NAME	NO	PIN NAME
1	TX	20	USB D+
2	RX	21	GND
3	CTS	22	GND
4	RTS	23	PIO0
5	PCM CLK	24	PIO1
6	PCM OUT	25	PIO2
7	PCM IN	26	PIO3
8	PCM SYNC	27	PIO4
9	AIO0	28	PIO5
10	AIO1	29	PIO6
11	RESET	30	PIO7
12	3.3V	31	PIO8
13	GND	32	PIO9
14	NC	33	PIO10
15	USB D-	34	PIO11
16	CSB		
17	MOSI		
18	MISO		
19	CLK		

The old customer please refer to (schematics, wiring diagrams, etc.)

This module is for upgrade BCM_LV module version

The figure below shows the core module peripheral circuit schematic diagram of the typical



application

If you use the factory default baud rate, do not want to modify the contents of the baud rate below do not need to care about:

The AT command for the master and slave. When used in pairs, the master and slave can be different baud rate can also transfer data, but the host and the device connected to the host baud rate to be the same, but also slave and slave devices connected to the same baud rate .

Setting mode:

Initial communication parameters for 9600, N, 8,1, before pairing (pairing indicator flashes) Send to modify the baud rate command

Before sending AT commands to ensure that the hardware is connected as follows:

The eight-pin interface with base plate, the first pin external power supply (3.3 to 5V), the sixth leg connected to the computer COM1 port (DB9 male) pin, pin 7 computer COM1 port pin, the eighth pin is connected to the computer COM1 port 5 feet. Using HyperTerminal or serial debugging assistant to open the COM1 port of the computer, enter the text "AT" to manually send.

Special note: master and slave pair can communicate between master and slave baud rate is required to be consistent between master and slave is to take the Bluetooth protocol, rather than serial port protocol.

1, test communications

Send: AT (return OK, one second left and right)

Back: OK

2, change the Bluetooth serial communication baud rate

Send: AT + BAUD1

Back to: OK1200

Send: AT + BAUD2
Back to: OK2400
.....
1 --------- 1200
2 --------- 2400
3 --------- 4800
4 --------- 9600
5 --------- 19200
6 --------- 38400
7 --------- 57600
8 --------- 115200
9 --------- 230400
A --------- 460800
B --------- 921600
C --------- 1382400
Not recommended to use more than 115200 baud rate, signal interference causes the system to instability.
The settings over 115,200 with a computer is not available, use microcontroller programming in higher than 115200 to use this baud rate and re-issue the AT command set low baud rate
AT command set baud rate, the next power do not need to set up and can be powered down to save the baud rate.

3, change the Bluetooth name (February 2008 after 24 new features)
Send: AT + NAMEname
Back to: OKname
Parameter name: To set the current name, the name of the Bluetooth search. 20 characters or less.
Example: Sending AT + NAMEbill_gates
Back OKname
The Bluetooth name changed to bill_gates
The parameters can be powered down to save, simply modify the time. PDA the end refresh can see the Bluetooth name changed.
4, change the Bluetooth pairing password
Send: AT + PINxxxx
Returns: OKsetpin
Parameter xxxx: To set a passcode, 4 bytes, this command can be used from the machine or host. The slave adapter or mobile phone pops up to enter the pairing password window, then manually enter this parameter
The number can be connected to the slave. Host in the main Bluetooth module connected digital camera, digital camera from the machine, find the pairing password of the camera, and then set up the White Bluetooth module, the master Bluetooth module can be。

10.5 Anexo IV: Manual del módulo LCD Keypad Shield

ITead Studio
Make innovation easier

Tech Support: support@iteadstudio.com

LCD Keypad shield

- 1602 LCD and 6 AD buttons for Arduino

Overview

This Arduino 1602 LCD Keypad shield is developed for Arduino compatible boards, to provide a user-friendly interface that allows users to go through the menu, make selections etc. It consists of a LCD1602 white character blue backlight LCD. The keypad consists of 5 keys — select, up, right, down and left. To save the digital IO pins, the keypad interface uses only one ADC channel. The key value is read through a 5 stage voltage divider.

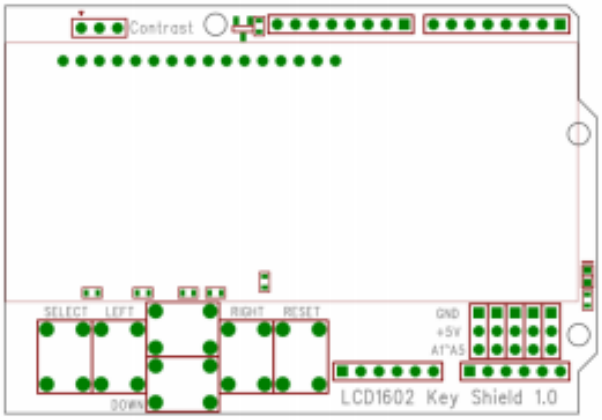
Specifications

PCB size	82.8mm X 56.6mm X 1.6mm
Power supply	5V DC
RoSH	Yes

Electrical Characteristics

Specification		Min	Type	Max	Unit
Power Voltage		4.5	5	5.5	VDC
Input Voltage VH	Target Voltage = 3.3V	3	3.3	3.6	V
	Target Voltage = 5V	4.5	5	5.5	
Input Voltage VL:		-0.3	0	0.5	V
Current Consumption		-	20	40	mA

Pins description



Pin	Function
A0	Button (select, up, right, down and left)
D4	DB4
D5	DB5
D6	DB6
D7	DB7
D8	RS (Data or signal display selection)
D9	LCD1602 Enable
D10	Backlight control

Revision History

Rev.	Description	Release date
v1.0	Initial version	6/23/2010