Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE CARRERA

TÍTULO DEL TFC: Introducción al control y telemetría de vehículos R/C mediante dispositivos Android.

TITULACIÓN: Ingeniería Técnica Aeronáutica, esp. Aeronavegación.

AUTOR: Domingo Margareto Sanz.

DIRECTOR: Marcos Quílez Figuerola.

FECHA: 8 de Julio de 2013.

Título: Introducción al control y telemetría de vehículos R/C mediante dispositivos Android.

Autor: Domingo Margareto Sanz.

Director: Marcos Quílez Figuerola.

Fecha: 8 de Julio de 2013.

Resumen

El objetivo de este trabajo es explorar y presentar distintas formas de modificar un vehículo teledirigido por radiocontrol para que pueda ser controlado desde un dispositivo android. En este documento se describen cuatro prototipos distintos de dificultad creciente.

El primer prototipo sigue uno de los numerosos tutoriales disponibles en la red para usar un teléfono android como mando a distancia. En este caso se ha verificado que todos los pasos descritos sean correctos, completos y que puedan seguirse sin ambigüedades.

En el segundo prototipo se sustituye completamente de la electrónica del vehículo R/C. En su lugar se utiliza un Arduino combinado con una placa comercial para el control de motores.

En el tercer prototipo se construye un driver de potencia propio para activar los motores. Esto permite detectar y corregir algunos problemas de interferencias entre la electrónica de potencia y la electrónica digital del circuito. En este tercer prototipo también se programa una aplicación Android propia para el control del vehículo.

El cuarto prototipo se ha desarrollado de forma conjunta con otro trabajo de final de carrera. En este caso se utilizan dos smartphones: uno de ellos embarcado en el vehículo y el otro, en manos del usuario, se utiliza como mando. El teléfono embarcado transmite la localización (GPS), la señal de la cámara y de otros sensores al teléfono de control. El vídeo y los datos transmitidos son representados en el teléfono de control, de manera que sea posible conducir el coche sin tener visión directa del mismo. La parte hardware de este prototipo ha sido objeto de este proyecto, mientras que la parte telemática ha sido realizada como otro trabajo final de carrera.

La redacción de la memoria se ha organizado para que pueda servir de quía a aquellos estudiantes o aficionados que quieran construir su propio prototipo.

Title: Introduction to control and telemetry of an R/C vehicle using Android devices.

Author: Domingo Margareto Sanz

Director: Marcos Quílez Figuerola

Date: July, 8th 2013

Overview

The main goal of this document is to explore and present different ways of modifying a radio remote control vehicle to be controlled from an android device. This document describes four different prototypes of increasing difficulty.

The first prototype is one of the many tutorials available on the network to use an android phone as a remote control. In this case it has been verified that all steps are correct, complete and can be followed unambiguously.

In the second prototype is completely replaced vehicle electronics R / C. Instead it uses an Arduino board combined with a commercial board to control the R / C car's motors.

In the third prototype we build an own power driver to turn on and turn off the engines. This allows detecting and correcting some problems of interference between power electronics and digital electronics circuit. In this third prototype we also build an own Android's application to control the vehicle.

The fourth prototype has been developed in conjunction with another university final project. In this case we use two smartphones: the first of them boarded on the vehicle and the second one, that is used to control the vehicle, to the user. The on-board phone transmits the location (GPS), the signal from the camera and other sensors to the control phone. The video and the transmitted data is represented in the control phone, so that it is possible to drive the car without having direct view of the car. The hardware part of this prototype has been the subject of this project, while the telematics has been realized as another university final project.

This report is organized so that it can serve as a guide to students or hobbyists who want to build their own prototypes.

ÍNDICE

Índice de figuras.

INTRODUCCIÓN

El objetivo de este trabajo es explorar y presentar distintas formas de modificar un vehículo teledirigido por radiocontrol para que pueda ser controlado desde un dispositivo android. En este documento se describen cuatro prototipos distintos de dificultad creciente. Cada uno de ellos requiere mayores habilidades que el anterior.

El primer prototipo consiste en sustituir el controlador del coche por la conocida placa microcontroladora Arduino. En la red existen numerosos tutoriales sobre este tema. Aquí se ha seleccionado uno de ellos y se ha verificado que todos los pasos descritos sean correctos, completos y que puedan seguirse sin ambigüedades. Este tutorial servirá de base para los siguientes ya que aprenderemos a retocar el código de Arduino y a comprender como se realiza la conexión bluetooth Arduino-smartphone y el tipo de comandos que se pueden enviar.

El segundo prototipo difiere del primero en la sustitución completa de la electrónica del vehículo R/C. Para ello necesitaremos un driver de potencia basado en un puente en H de transistores para poder alimentar los motores del vehículo ya que Arduino sólo nos da salidas de 5 V como máximo, salidas que activaran los transistores que darán la potencia que requieren los motores para ser activados.

En el tercer prototipo crearemos nuestro propio driver de potencia para activar los motores. Esto nos permitirá detectar y corregir algunos problemas de interferencias entre la electrónica de potencia y la electrónica digital de nuestro circuito. En este tercer prototipo también programaremos nuestra propia aplicación Android para controlar el vehículo..

El cuarto prototipo, es el más ambicioso y se ha desarrollado de forma conjunta con otro trabajo de final de carrera. En este caso se utilizan dos smartphones: uno de ellos embarcado en el vehículo y el otro, en manos del usuario, se utiliza como mando. El teléfono embarcado transmite la localización (GPS), la señal de la cámara y de otros sensores al teléfono de control. El vídeo y los datos transmitidos son representados en el teléfono de control, de manera que

sea posible conducir el coche sin tener visión directa del mismo. La parte hardware de este prototipo ha sido objeto de este proyecto, mientras que la parte telemática ha sido realizada como otro trabajo final de carrera por Víctor Nieto.

La redacción de la memoria se ha organizado para que pueda servir de quía a aquellos estudiantes o aficionados que quieran construir su propio prototipo.

1 Primer prototipo: control remoto de un vehículo R/C mediante un smartphone

1.1 Objetivo

El principal objetivo de este prototipo es conseguir la primera plataforma “coche con Arduino y smartphone” estable, que nos permita controlar el coche. Para ello se ha seguido uno de los mucho tutoriales que podemos encontrar a día de hoy en internet.

Hemos realizado una selección de tutoriales. Entre los consultados hemos elegido “http://www.mobot.es/MobotBTCar.html” ya que contiene información sobre la electrónica general que tiene un coche de radiocontrol eléctrico, el código a introducir en Arduino y la aplicación android necesaria para conectar Arduino al Smartphone y controlar el coche mediante botones y acelerómetros.

Este prototipo nos dará las bases para la realización de los siguientes prototipos.

1.2 Descripción de materiales y herramientas

A continuación describiremos y daremos los datos técnicos del material que hemos utilizado para esta fase.

1.2.1 Arduino

En la página oficial de Arduino, http://www.arduino.cc/es/. [ref 1]

“Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP).

Las placas pueden ser hechas a mano o compradas montadas de fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta, así pues eres libre de adaptarlos a tus necesidades.”

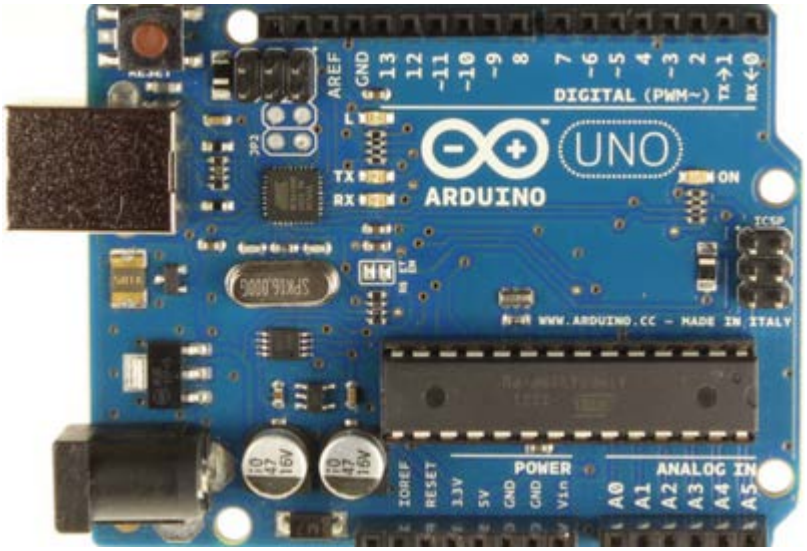


Fig 1.1 Placa arduino Uno.

1.2.2 Módulo de comunicación bluetooth

Para todos los métodos utilizaremos el siguiente transceptor bluetooth: JY-MCU BT BOARD V1.05.

1.2.2.1 Características

Permite a su dispositivo tanto para enviar o recibir los datos mediante la tecnología Bluetooth TTL sin conectar un cable serial al ordenador.
Funciona con cualquier adaptador Bluetooth USB.

1.2.2.2 Especificaciones:

Bluetooth versión: V2.0 + EDR.
Voltaje de funcionamiento: 5V.
Configuración por defecto: 9600,8, 1, n.

La cobertura de la señal: 10 metros.
Tamaño del transceptor BT: 4.3 x 1.6 x 0.7cm.

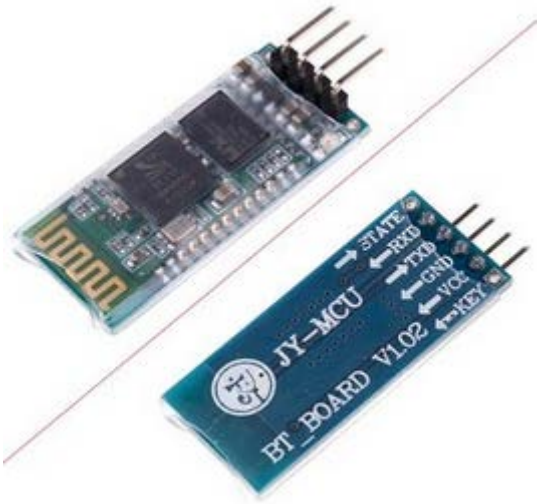


Fig 1.2 Módulo bluetooth para Arduino

1.2.3 Smartphone con sistema operativo Android

Para este caso lo que hemos utilizado ha sido un Samsung Galaxy Scl con sistema operativo Android versión 2.3.6.

Fig 1.3 Smartphone utilizado en el proyecto, GT-i9003.

1.2.4 Vehículo teledirigido R/C

Lo que realmente nos interesa es tener un chasis con 2 motores que controlen la tracción y la dirección del coche.

En esta primera fase hemos aprovechado parte de la electrónica original del coche. Le hemos quitado el controlador y el receptor de radiofrecuencia, después hemos aprovechado los transistores que controlan tracción y dirección. Hemos soldado unos conectores para poder conectar fácilmente las salidas de Arduino al control del coche.

En el apartado 1.3.3 se explica cómo realizar las conexiones.

1.2.5 Entorno de programación para Arduino

Se obtiene de la página oficial de Arduino.

http://www.arduino.cc

El entorno de código abierto Arduino hace fácil escribir código y cargarlo a la placa E/S. Funciona en Windows, Mac OS X y Linux. El entorno está escrito en Java y basado en Processing, avr-gcc y otros programas también de código abierto.

En el ANEXO I se presenta el código de Arduino obtenido de la página del tutorial de referencia [ref 2].

1.2.6 Aplicación de control remoto para Android

Para transmitir órdenes desde el Smartphone a nuestro Arduino utilizaremos una aplicación que está disponible de manera gratuita en “Google Play”. La aplicación se llama “MOBOT BT CAR”.[ref 2]

1.3 Diseño, realización y configuración del prototipo

Este apartado describe el hardware y la programación de nuestro módulo Arduino, que se encuentra en el Anexo I.

1.3.1 Configuración del hardware

A continuación describimos el montaje y programación del prototipo y las pequeñas modificaciones que hemos de realizar en la electrónica del coche, especificado en el apartado 1.3.3.

1.3.2 Configurando Arduino con el módulo bluetooth

La alimentación del módulo de bluetooth la haremos mediante la salida de alimentación de 5V que nos proporciona Arduino. Y el “ground” del módulo conectado a unos de los GND que nos proporciona la placa Arduino.

La conexión de las patillas RX (recepción) y TX (transmisión) del módulo bluetooth las conectaremos a las patillas de conexión serie de la placa Arduino de la siguiente manera:

Patilla RX del módulo bluetooth conectada a la entrada TX de la placa Arduino y la patilla TX del módulo BT conectada a la entrada RX de la placa Arduino. Esquemáticamente:

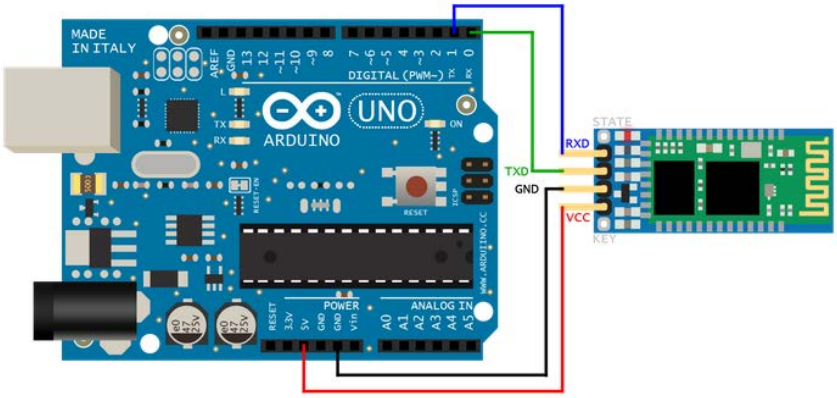


Fig 1.4 Esquema de conexión del módulo BT en la placa Arduino.

1.3.3 Configurando Arduino con los motores

En este primer prototipo conectamos las salidas de Arduino directamente sobre los transistores a los que llega la alimentación de los motores. Previamente necesitaremos soldar unos pines hembra en los colectores de los transistores del propio coche.

Reconoceremos los transistores siguiendo los cables que van de los motores del coche a la placa integrada que tiene el coche. Una vez reconocidos soldaremos 2 pines por motor, así podremos invertir la polaridad de activación de los motores.

Otra manera de identificar estos transistores es siguiendo las tiras que salen del chip de control que trae el vehículo. Este tipo de vehículos de producción barata están basados en su gran mayoría en la misma electrónica o muy similar, en la página Mobot BTCar encontraremos que salidas del microcontrolador que trae el coche son las que nos interesa seguir ya que sólo utilizaremos 4 salidas: adelante, atrás, izquierda y derecha.

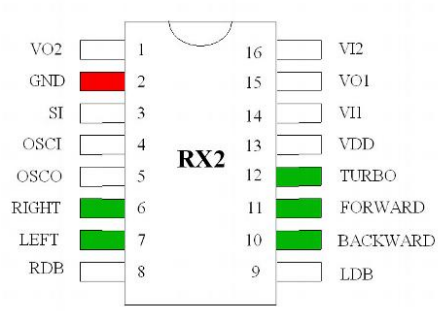


Fig 1.5 Salidas asignadas del microcontrolador original del vehículo.

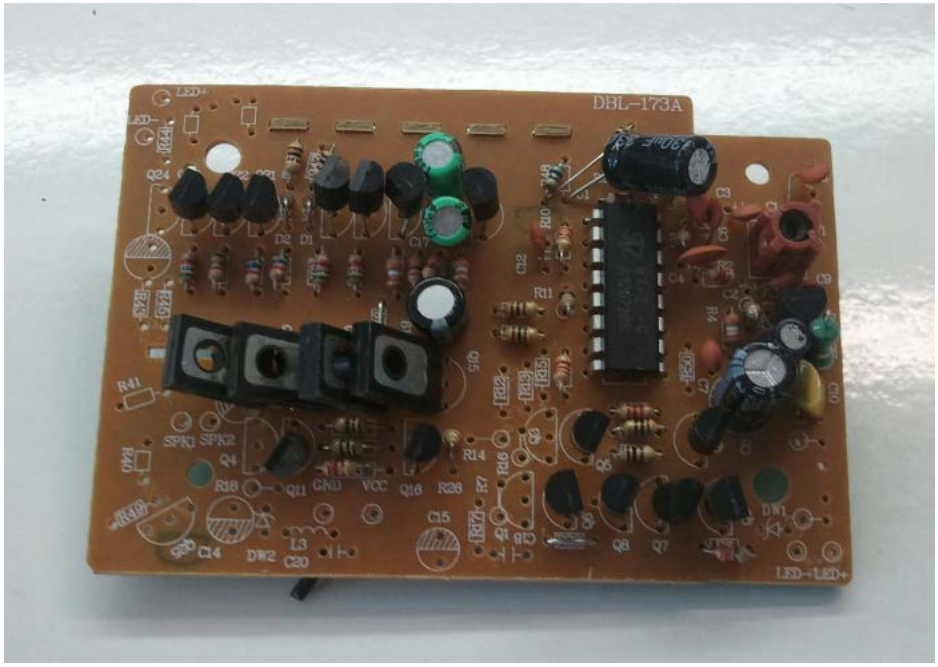


Fig 1.6 Electrónica original del coche.

Siguiendo las pistas de las salidas 6, 7, 10 y 11 llegaremos a los colectores de los transistores donde hemos de soldar los pines hembra. (Fig 2.5)

En la figura 2.6 vemos la electrónica original del coche en la que hemos de seguir las pistas mencionadas anteriormente para conectar nuestras señales de control.

1.3.3.1 Motor de tracción

Conectaremos el motor de tracción a las salidas 5 y 6 de Arduino. Lo conectamos a estas salidas ya que tienen la capacidad para hacer modulación por ancho de pulso (Pulse Width Modulation,PWM) y será la manera de poder controlar electrónicamente la velocidad de rotación del motor. A continuación

explicamos el funcionamiento de la modulación PWM, información extraída de la página de Arduino [ref 3].

“La **modulación por ancho de pulsos** (también conocida como **PWM**, siglas en inglés de *pulse-width modulation*) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica para controlar la cantidad de energía que se envía a una carga.

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

$$D = \frac{\tau}{T}$$

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

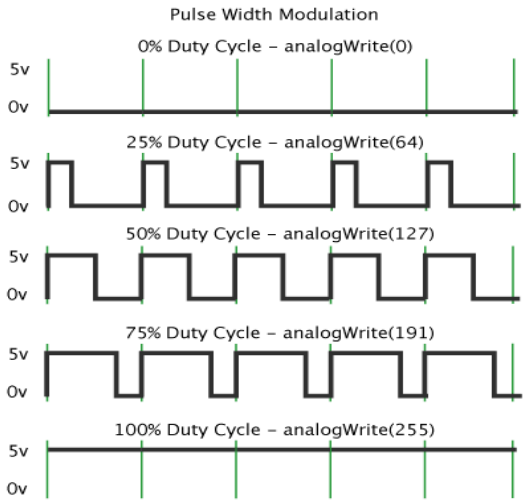


Fig 1.7 Modulación por ancho de pulso PWM imagen extraída de [3].

1.3.3.2 Motor de dirección

Los polos del motor destinado al control de tracción los conectaremos a las salidas 10 y 9 de Arduino. El motor de dirección tiene dos topes físicos de manera que el motor se activará con todo el voltaje que provenga de la batería del coche. Asignamos las salidas 10 y 9 de la placa Arduino por que no necesitamos controlar la velocidad de giro del motor. Giro derecha todo y giro izquierda todo.

2 Segundo prototipo: utilización de un driver de potencia comercial

2.1 Objetivo

En este prototipo se busca sustituir toda la electrónica del coche por un Arduino y un driver de potencia, dejando la batería original y el chasis con motores y ruedas del coche.

2.2 Descripción de materiales y herramientas

Necesitaremos lo mismo que en la primera fase mas un driver de potencia cuya función es activar los motores. A continuación describimos los materiales que utilizamos y no se han usado en el protoripo anterior.

2.2.1 Vehículo R/C sin electrónica de control

Compramos un coche teledirigido y quitamos toda la electrónica que hay en su interior, dejando únicamente los cables de alimentación de la batería y las alimentaciones de los 2 motores (tracción y dirección). A continuación ilustramos el proceso paso a paso.

Fig 2.1 Coche teledirigido.

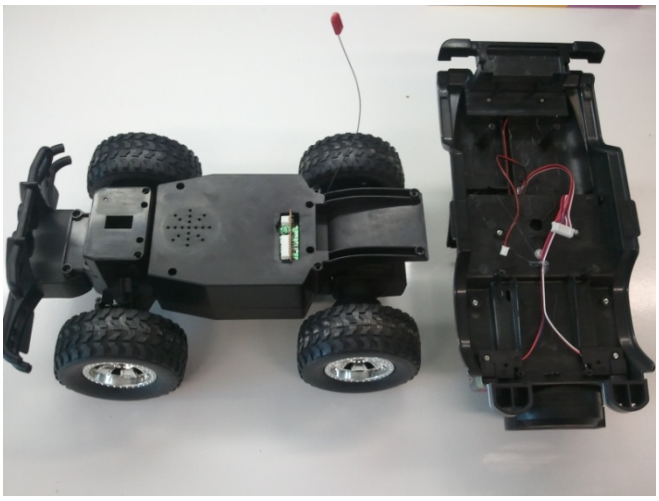


Fig 2.2 Coche teledirigido sin carcasa externa.

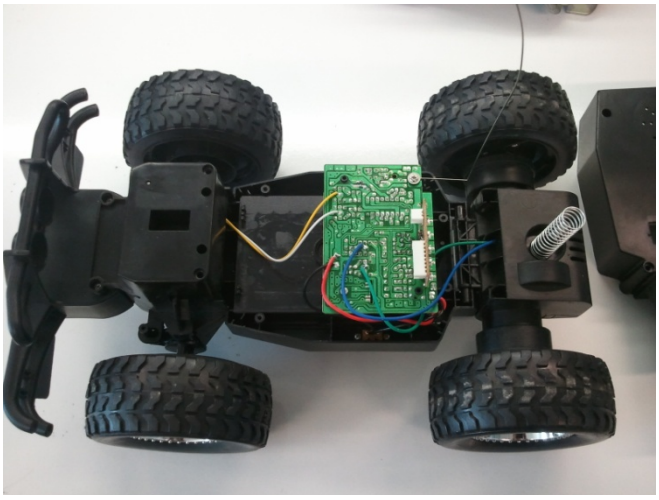


Fig 2.3 Acceso a la electronica del coche teledirigido.

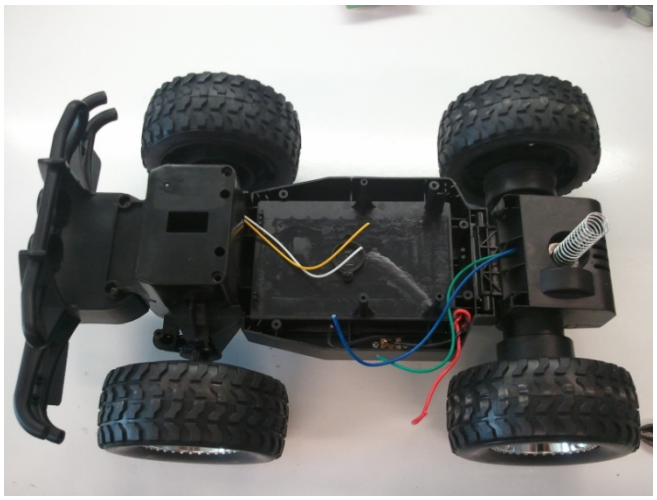


Fig 2.4 Extracción de la electrónica completa del coche.

2.2.2 Driver de potencia “L298 Dual H-Bridge Motor Driver”

El driver de potencia es un shield conectable al Arduino que incorpora un puente en H de transistores. Con este puente en H podremos invertir la polaridad de la activación de motores y conseguir giro izquierda y giro derecha con 2 salidas y marcha adelante y marcha atrás con otras dos salidas. Esquemáticamente:

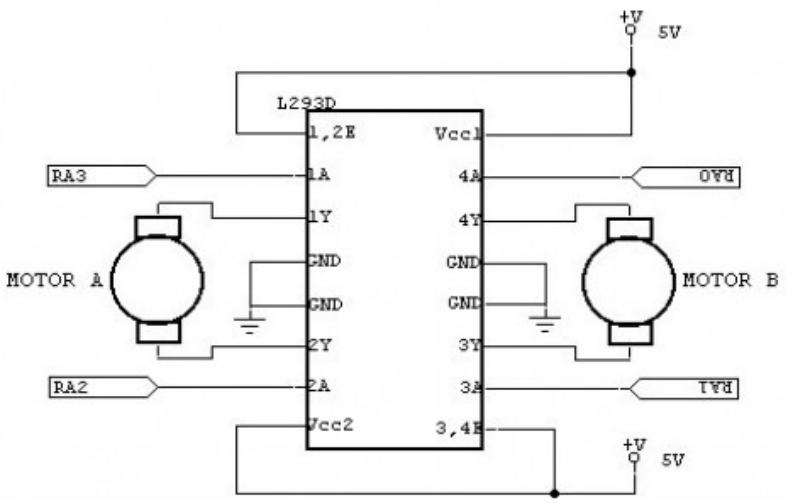


Fig 2.5 Esquema del driver L293B.

Utilizaremos el driver comercial “L293 Dual H-Bridge Motor Driver”

Y tiene las siguientes especificaciones técnicas:

Driver: L298.
Alimentación: +5V~+46V.
Corriente máxima Io: 2ª.
Salida para Lógica Vss: +5~+7V.
Corriente para Lógica: 0~36mA.
Potencia máxima: 25W (Temperatura 75ºC).

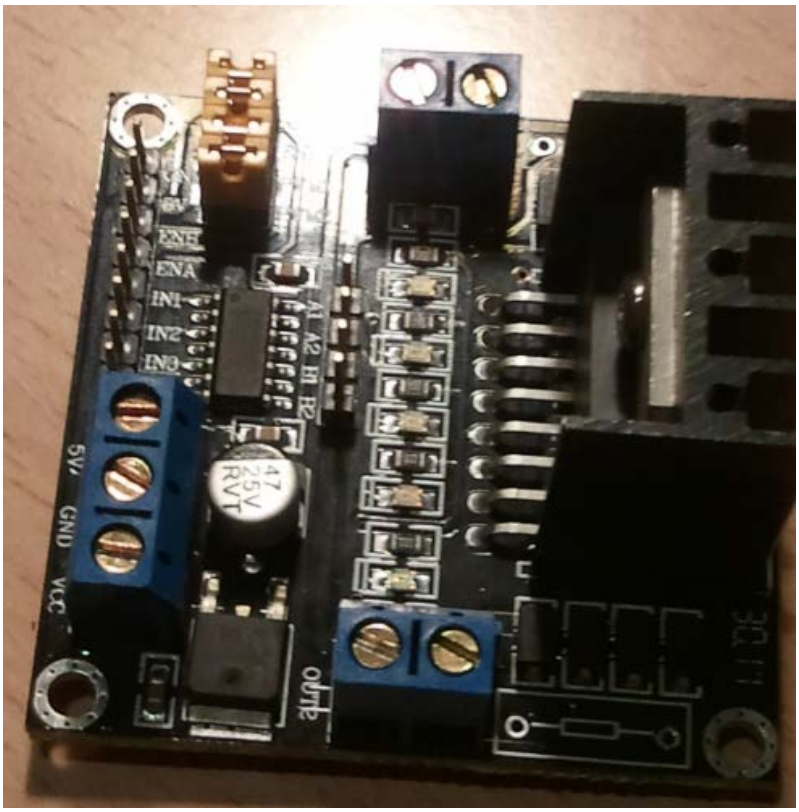


Fig 2.6 Driver de potencia comercial utilizado en el segundo prototipo.

En la figura 3.6 observamos el driver de potencia comercial utilizado para la realización de este prototipo.

A la izquierda de la imagen vemos una regleta de conexión a la que conectaremos directamente la bateria del coche. Encima de estos conectores vemos los siguientes pines de abajo hacia arriba son: IN4, IN3, IN2, IN1, ENA, ENB, 5V y GND. Donde IN son “entradas”, EN es el “activar” de cada salida y GND es la referencia de masa.

El conector de la parte supierior de la imagen es la salida de potencia 1 (OUT1) controlada por los pines IN1 e IN2. Y el conector de la parte inferior de la imagen es OUT2 que se activa con las entradas IN3 e IN4.

Es imprescindible contar con un driver de potencia ya que las salidas de Arduino son de 5V y no podremos activar motores suficientemente potentes para mover el vehículo, y en el caso en que pudiese activar motores, la activación de éstos provocaría una caída de tensión que el controlador no soportaría y se reiniciaría cada vez que activásemos los motores.

2.2.3 Software y herramientas de programación

Utilizaremos el mismo software tanto en Arduino como en el Smartphone que se han utilizado en el capítulo 1.

2.3 Diseño, realización y configuración del prototipo

2.3.1 Conexión Arduino-driver potencia-motores

En este método utilizaremos las mismas salidas que en el prototipo 1. Ahora conectaremos las salidas a los activadores del driver de potencia, como antes: el motor de tracción a las salidas 5 y 6 de Arduino, y el motor de tracción a las salidas 4 y 7 de Arduino.

En el driver de potencia conectamos la batería del coche a la alimentación del driver y los motores a los conectores que hay en el driver de potencia para activarlos. Ver figura 3.7.

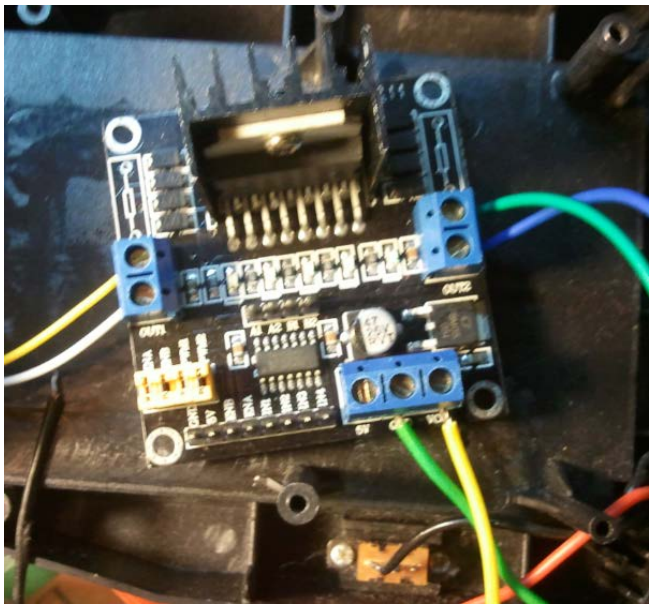


Fig 2.7 Driver comercial conectado a la batería y a los motores.

Para poder conectar las entradas de una manera sencilla nos preparamos un conector con placa de tiras, pines hembra y 8 cables a los que conectaremos con Arduino. Ver figura 3.8.

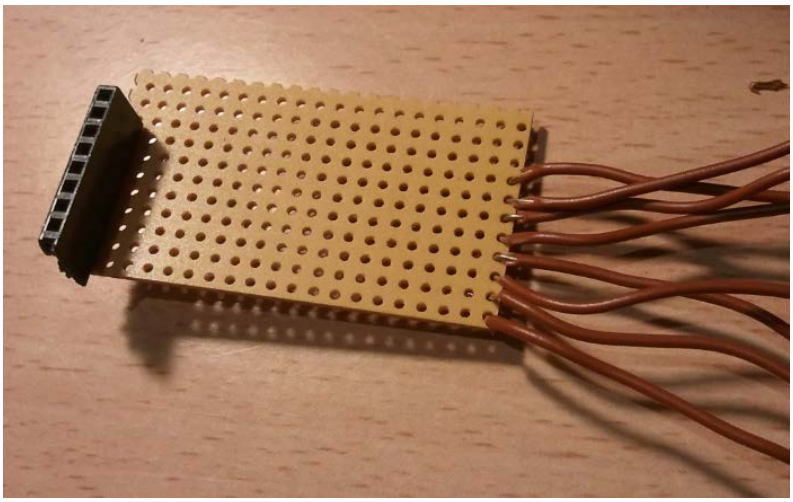


Fig 2.8 Conector para el driver comercial con Arduino.

Finalmente conectamos el driver a la placa Arduino y asignamos las conexiones de la siguiente manera:

Entrada IN1 al pin número 9 de Arduino.

Entrada IN2 al pin número 10 de Arduino.

Entrada IN3 al pin número 11 de Arduino.

Entrada IN4 al pin número 12 de Arduino.

Entrada ENA al pin salida de 5 V de Arduino.

Entrada ENB al pin salida de 5 V de Arduino.

Entrada GND al pin GND de Arduino.

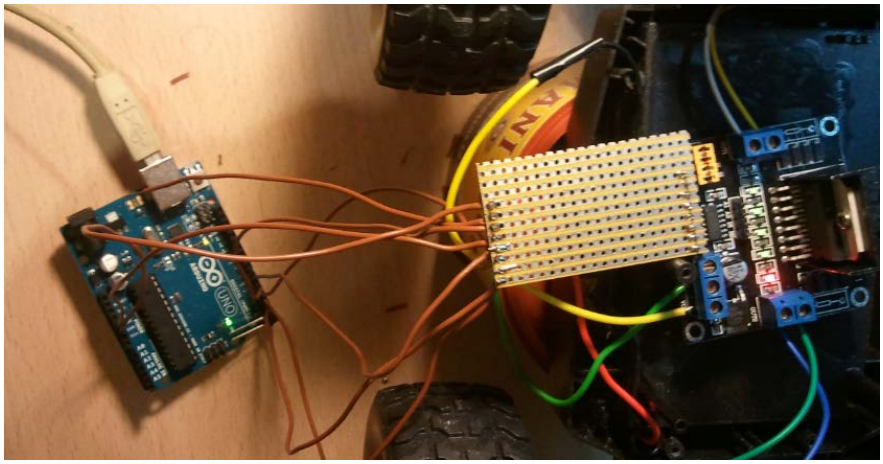


Fig 2.9 Driver comercial conectado a Arduino.

3 Tercer prototipo: montaje de un driver de potencia

3.1 Objetivo

En este capítulo nos centraremos en desarrollar nuestro propio driver de potencia y además nuestro propio programa para android.

3.2 Descripción de materiales y herramientas

Para el tercer prototipo utilizaremos el montaje del prototipo 2 y motaremos nuestro driver de potencia encima de Arduino, diseñándolo de manera que se pueda conectar la placa del driver sobre Arduino directamente.

3.2.1 Material necesario para realizar el driver de potencia

- Placa de tiras.
- Pines macho.
- Puente en H integrado L293B (datasheet en Anexo III).

3.2.2 Software y herramientas de programación

Para la realización de nuestro propio programa de android utilizaremos una aplicación web desarrollada por el MIT esta aplicación web se llama “appinventor” y nos permitirá programar nuestro teléfono haciendo programación en sketch o por piezas de puzle. [ref4]

Necesitaremos tener una cuenta de gmail ya que para usar esta herramienta es necesario iniciar sesión en google.

3.3 Diseño, realización y configuración del prototipo

3.3.1 Driver de potencia

El principio de funcionamiento de nuestro driver de potencia está basado también en un puente en H para el control de los motores. La diferencia con el driver comercial es que el nuestro lo alimentamos directamente desde Arduino con la salida Vin, la cual nos dará la misma corriente con la que alimentamos el Arduino.

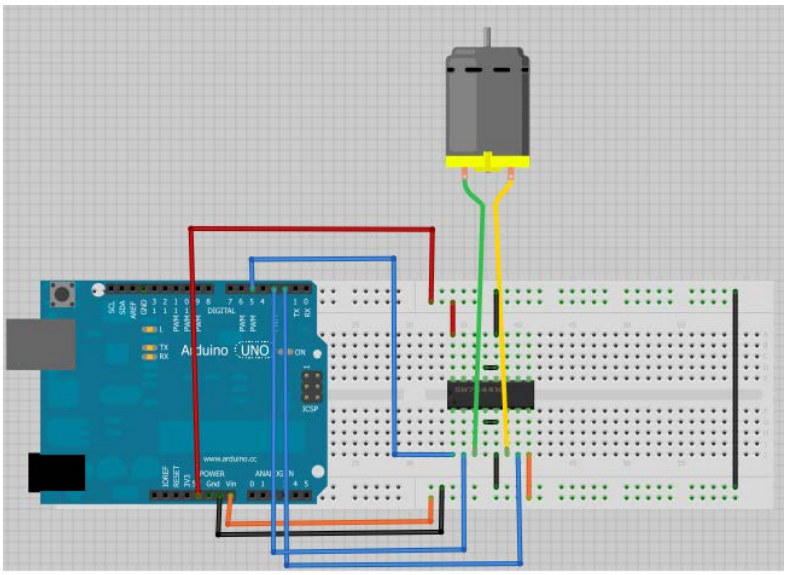


Fig 3.1 Esquema en Fritzing [5] de conexion de un motor.

3.3.1.1 Realización

Realizaremos nuestro driver en una placa de tiras y dejaremos el espacio necesario para poder pinchar nuestro driver directamente sobre la placa Arduino. Esquema del circuito con el chip:

(Recomiendo hacer un esbozo en hoja cuadriculada antes de soldar o cortar pistas en la placa de tiras)

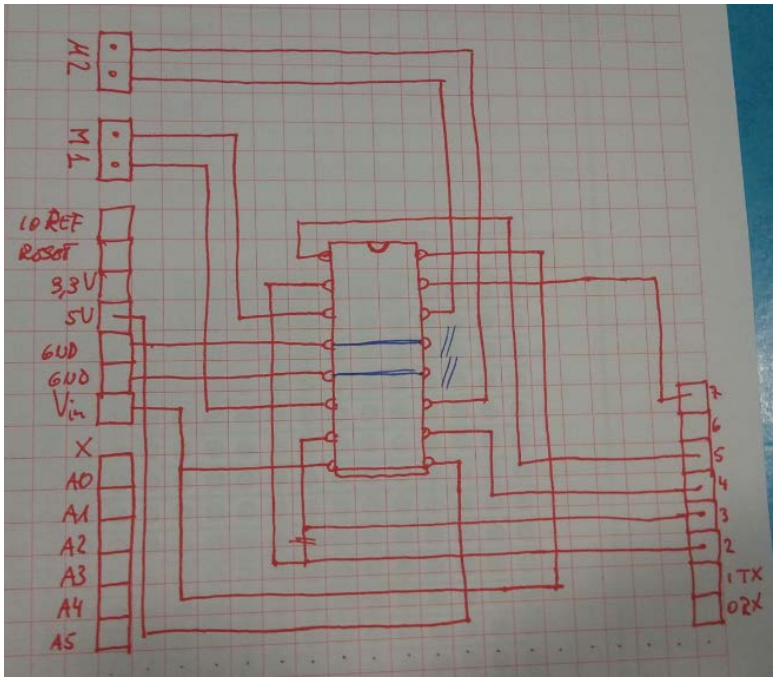


Fig 3.2 Planificación del driver en hoja cuadriculada para pasar a placa de tiras, cada cuadro representa un topo de la placa, así nos aseguramos que las conexiones coincidan con la placa Arduino Uno.

Soldamos los conectores en la placa de tiras, cortamos las tiras de la placa de manera que podamos realizar las conexiones necesarias.

Recomiendo soldar nuestro puente en H de 16 conectores en el centro de la placa. Después hacemos las conexiones verticales de nuestro esquema con cable normal por la parte que no están las tiras conductoras.

Finalmente conectaremos los conectores de 2 conexiones que alimentarán a los motores del coche, uno al lado del otro.

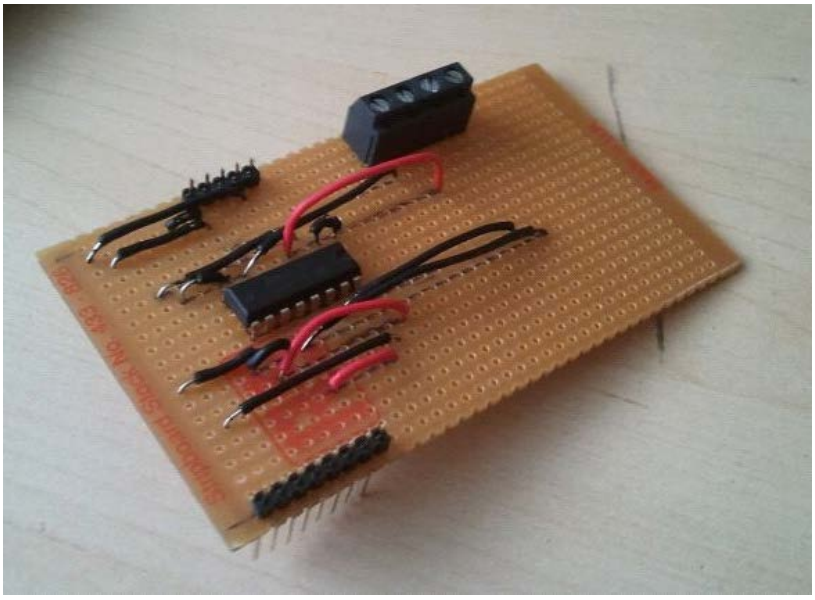


Fig 3.3 Imagen del primer driver “casero”.

Finalmente conectado a Arduino nos queda el siguiente bloque:

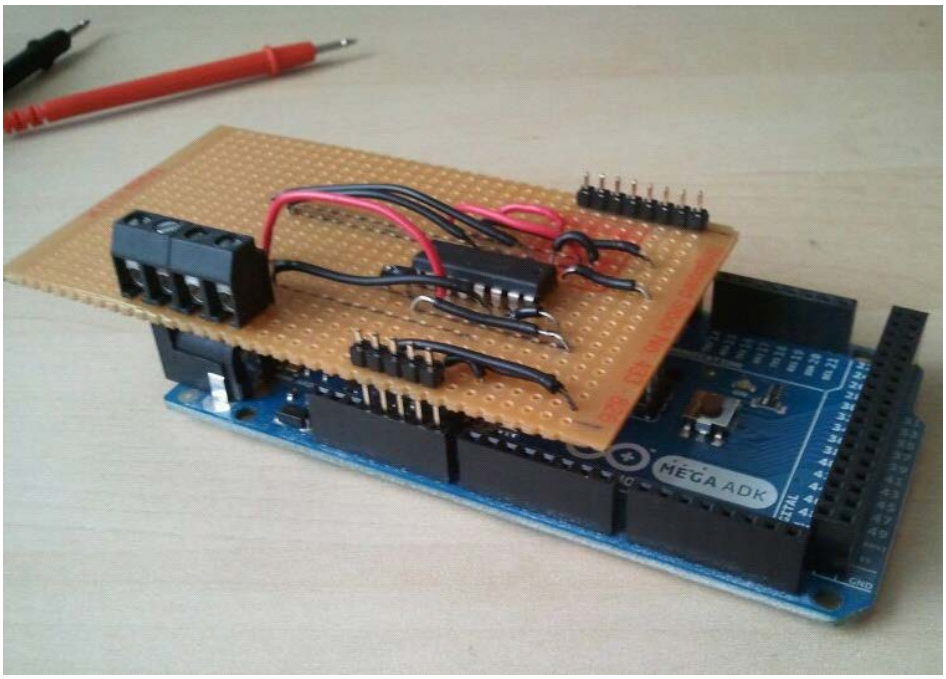


Fig 3.4 Foto del bloque (Arduino-driver).

3.3.1.2 Problemas de EMC: Interferencias conducidas

El driver de potencia diseñado en los apartados anteriores nos presenta un problema en la alimentación de nuestro procesador.

El problema se localiza en la alimentación ya que al activar motores el circuito electrónico aparecen transitorios de 1V de amplitud. Este transitorio nos provoca que Arduino se reinicie cada vez que se activa el motor y a efectos del control del coche genera que el motor se apague y se encienda cada vez que Arduino se reinicia haciendo imposible el control en la tracción del vehículo.

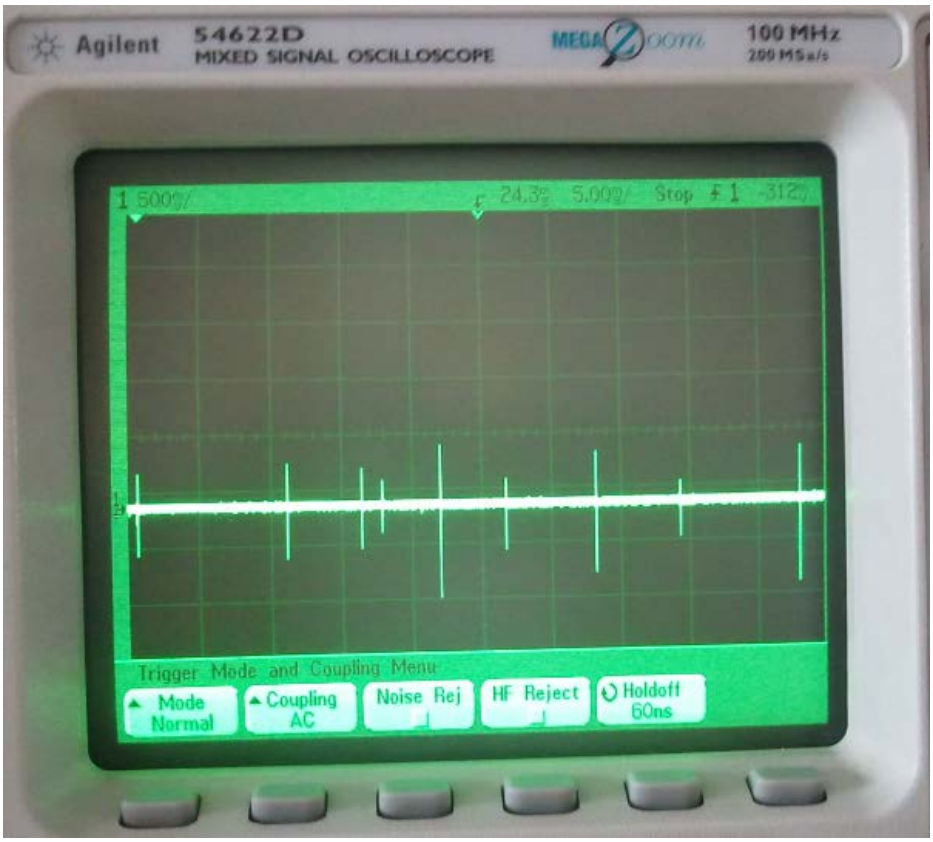


Fig 3.5 Imagen de osciloscopio, caidas de tensión al activar motor de tracción. Escala de 500mV / 500ms.

3.3.2 Driver de potencia mejorado

Para solucionar el problema de los transitorios en nuestro circuito diseñamos un nuevo driver, se diferencia del anterior en 2 puntos concretos:

- Añadir condensadores para suavizar y minimizar la amplitud del transitorio.
- Y una mejora en la alimentación de Arduino desde el mismo driver.

Utilizaremos 2 tipos de condensadores: 1 de 100µF (electrolítico) y 5 de 0,1µF (cerámicos). Esquemáticamente (fig 4.6):

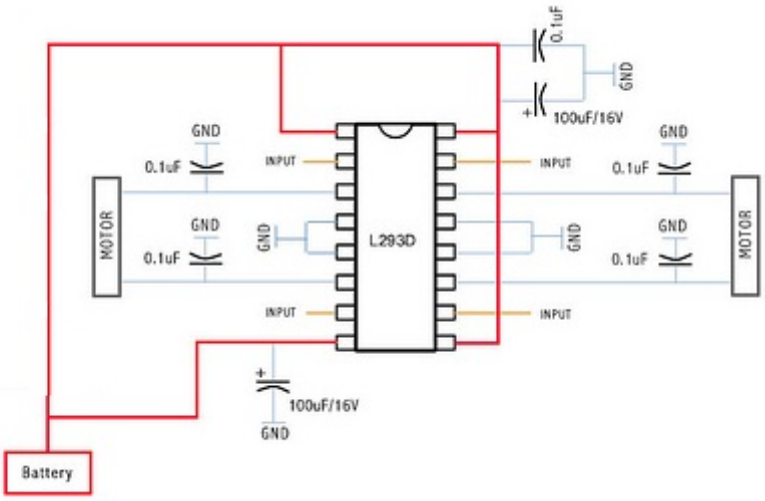


Fig 3.6 Esquema del nuevo driver con adición de condensadores.[ref 5]

El condensador de 100µF va conectado justo a la entrada de la alimentación junto con otro condensador de 0,1 µF y los cuatro condensadores de 0,1µF van en las 4 salidas de los motores a tierra. A continuación se muestra el nuevo diseño del driver en un croquis a mano alzada para pasar a placa de tiras.

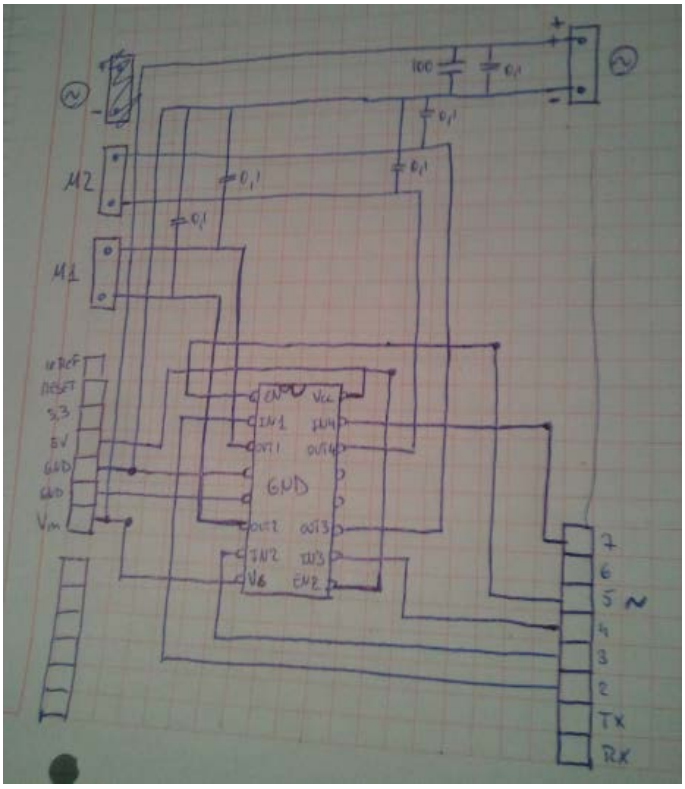


Fig 3.7 Planificación del driver en hoja cuadriculada para pasar a placa de tiras, cada cuadro representa un topo de la placa, así nos aseguramos que las conexiones coincidan con la placa Arduino Uno.

Una vez diseñado, adaptamos la medida de la placa de tiras, colocamos los elementos en la placa y los soldamos. Ver fig 4.8.

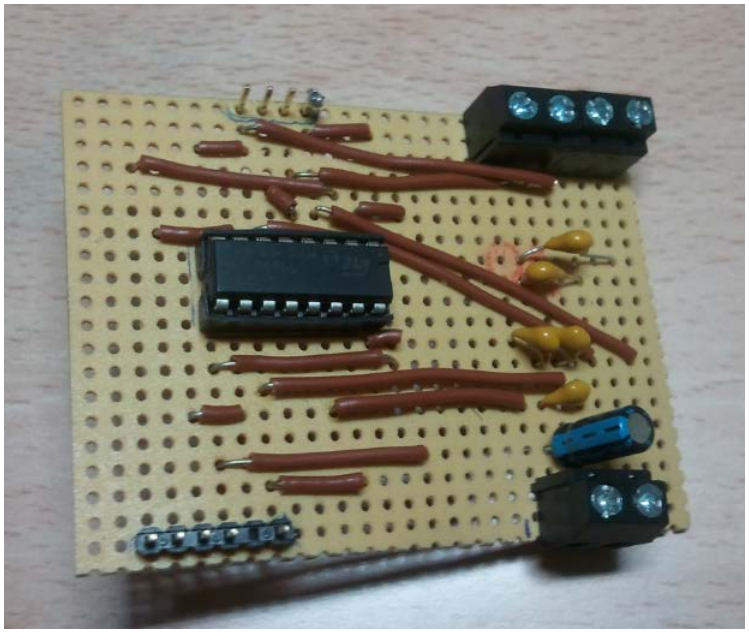


Fig 3.8 Imagen del nuevo driver con filtrado.

Una vez soldado nuestro nuevo driver, lo conectamos a Arduino (ver Fig 4.9) y lo testeamos en el laboratorio. (ver fig 4.10).

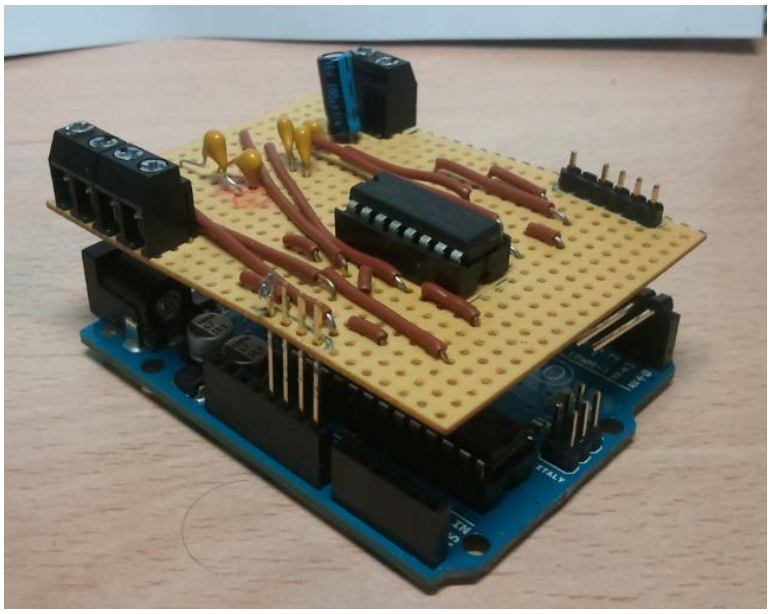


Fig 3.9 Bloque (Arduino-driver filtrado).

En la Fig 4.10 podemos ver que nuestro driver filtra la caída de tensión de manera significativa.

Hemos disminuido la caída de tensión a un 10% de la caída de nuestro anterior diseño. En la fig 4.10 se puede apreciar que la caída de tensión en nuestro nuevo driver es de 0,1 V. Hemos logrado que ahora el procesador no se reinicie cada vez que activemos un motor.

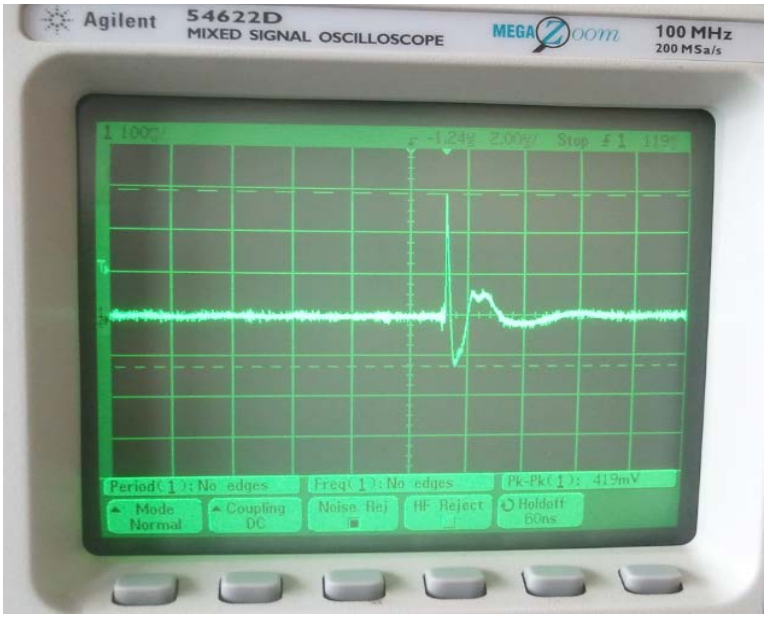


Fig 3.10 Imagen de osciloscopio; filtrado del transitorio al activar motor de tracción. Escala del osciloscopio 100mV / 200ms.

3.3.3 Control remoto táctil mediante Smartphone

La interfaz en el móvil será muy simple, 6 botones como muestra la figura 4.11

En nuestra aplicación necesitamos botones para el control del coche como son: avanzar, marcha atrás, izquierda y derecha. Y 2 botones más para: establecer conexión con Arduino mediante bluetooth y otro botón para cerrar la aplicación de android.

Fig 3.11 Interfaz del programa de control en Android.

A continuación mostraremos los bloques en los que podemos ver las relaciones entre los botones de la interfaz y las órdenes que tienen.

3.3.3.1 Conexión BT

El emparejamiento BT de Arduino con el Smartphone lo realizamos conociendo la MAC de la antena BT conectada a Arduino. Para saber la MAC de la antena la hemos de tener conectada y con el móvil buscar dispositivos BT. En nuestro caso es el dispositivo “Linvor” que es el identificador que viene por defecto en la antena y la contraseña para establecer el emparejamiento es dependiendo de la antena; 0000 o sinó 1234.

En nuestro caso tenemos la MAC 00:11:11:21:01:95 y la contraseña es 0000. Para establecer el emparejamiento en nuestro programa definimos la DireccionMAC como 00:11:11:21:01:95

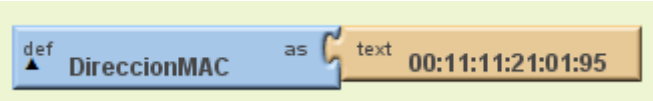


Fig 3.12 Imagen de programación en Scratch. Definicion de la MAC a la que nos hemos de conectar.

Inicializamos las variables una vez se haya establecido el emparejamiento con el módulo bluetooth. (Fig 4.12)

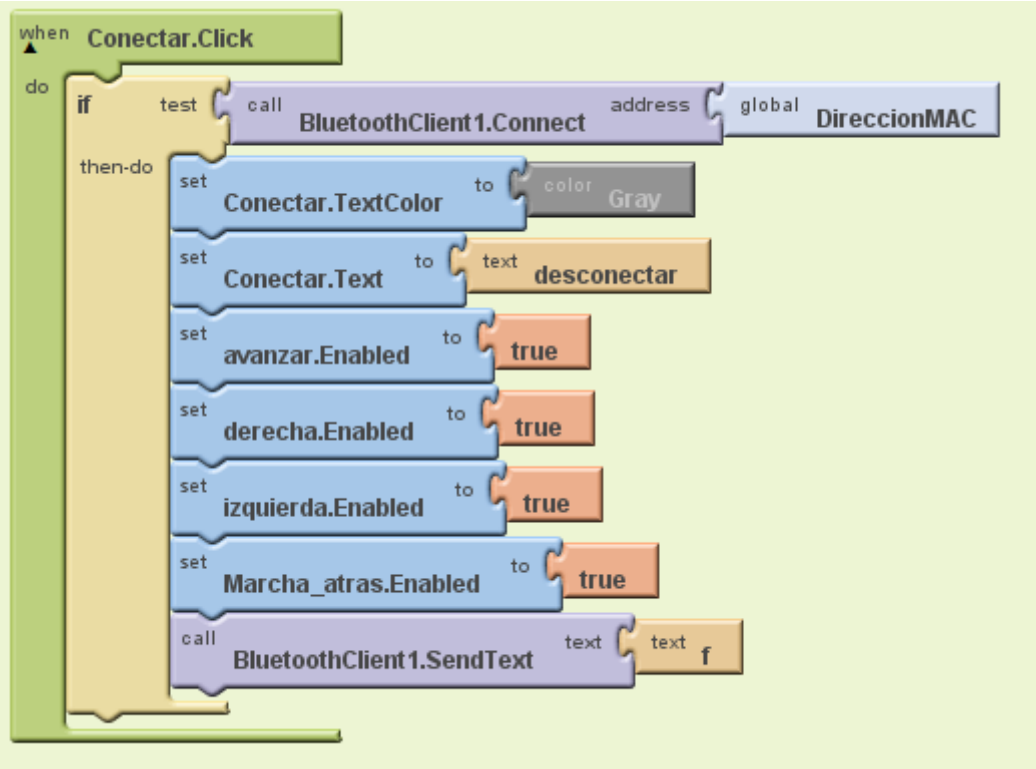


Fig 3.13 Inicialización de las variables sólo cuando se haya establecido el emparejamiento.

3.3.3.2 Desconexión BT

Necesitamos la desconexión para que el móvil no quede asociado a Arduino. Realmente no es necesario este punto ya que podríamos desemparejar los elementos cerrando la conexión BT desde el móvil, aunque de esta manera nos aseguramos que Arduino no queda asociado a la última conexión realizada.

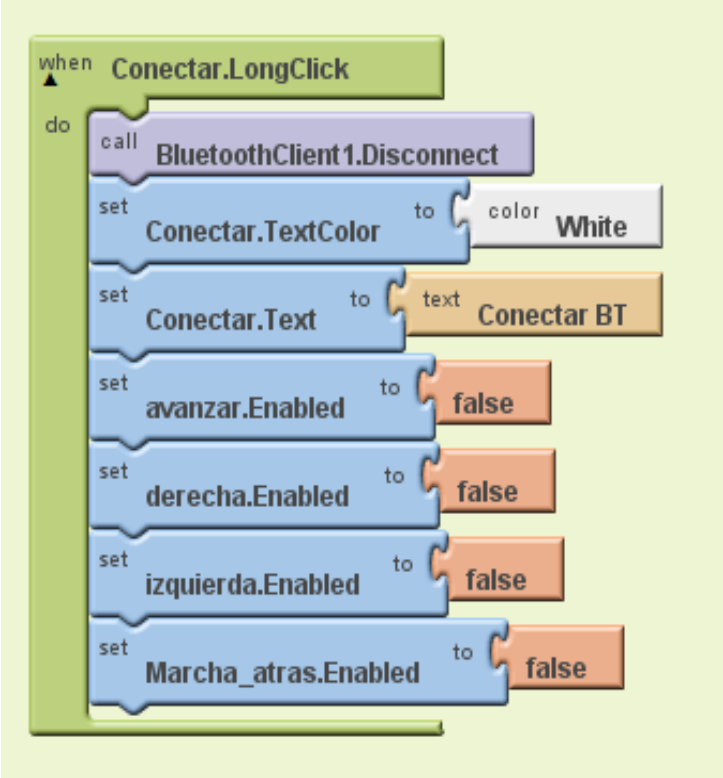


Fig 3.14 Cambio de condiciones al desconectar el modulo BT.

3.3.3.3 Comandos enviados a Arduino.

Enviamos diferentes comandos a Arduino para diferenciar las órdenes que recibe el coche:

En nuestro caso enviamos los siguientes comandos:

Avanzar: enviamos el carácter ”**f**” al Arduino.

Marcha atrás: enviamos el carácter ”**b**” al Arduino.

Izquierda: enviamos el carácter “**r**” al Arduino.

Derecha: enviamos el carácter “**l**” al Arduino.

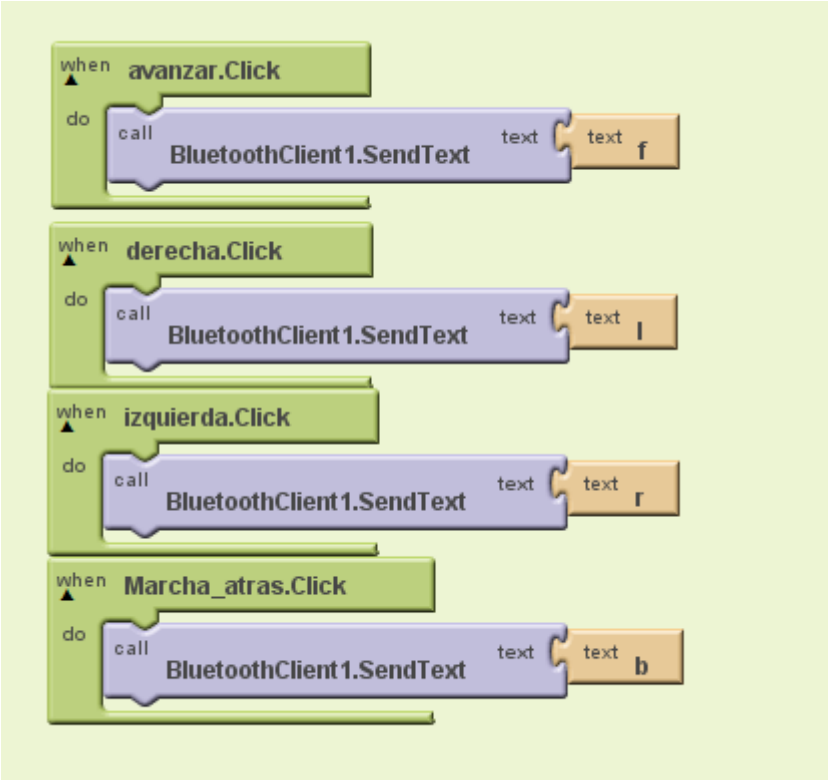


Fig 3.15 Comandos enviados a Arduino por cada botón.

3.3.4 Control remoto gestual mediante los acelerómetros del smartphone

En esta versión del programa hemos de declarar en la visión de elementos, los acelerómetros del Smartphone como elementos no visibles de nuestra apk.

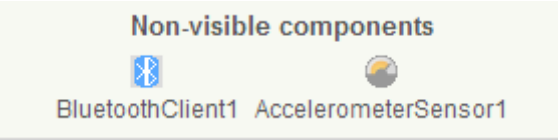


Fig 3.16 Componentes no visibles en el appinventor del MIT.

Usamos el mismo método de conexión y desconexión que en la aplicación anterior, simplemente definimos las condiciones iniciales ya que ahora, además de botones tenemos que inicializar los acelerómetros como desactivados.

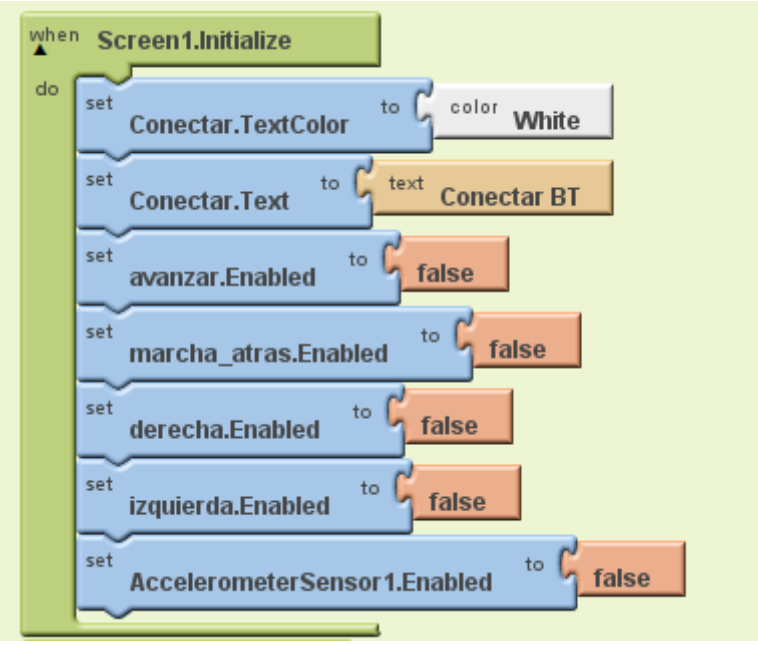


Fig 3.17 Cambio de condiciones al establecer la conexion BT.

A continuación mostramos el esquema para que nuestro Smartphone mande los comandos en función de la posición en la que se encuentra. Fig 4.17

Añadimos dos comandos adicionales respecto a la versión de botones que serán:

Stop giro; el carácter enviado será “z”.

Stop tracción; el carácter enviado será “v”.

La limitación en los sensores de 0,5 puede ampliarse hasta 9,8 que es el valor máximo q pueden tener estos sensores. Podemos subir el valor limite a 3 para quitar sensibilidad a los sensores ya que dependiendo del modelo de Smartphone que se use tendremos más o menos sensibilidad y puede provocar envíos de comandos a causa de una leve vibración del mismo.

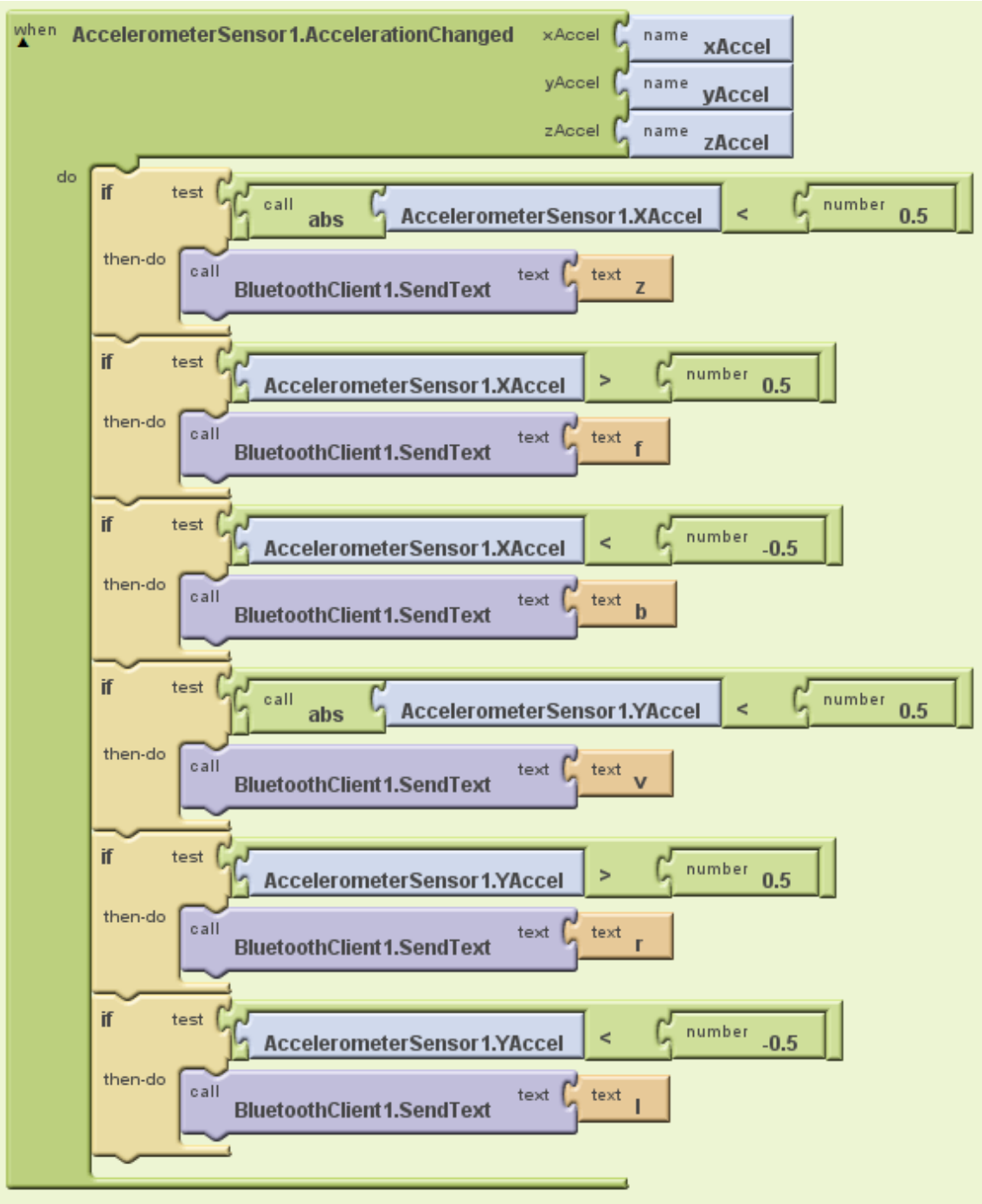


Fig 3.18 Envio de comandos a Arduino en función de la posición del smartphone.

4 Cuarto prototipo: Conducción mediante un segundo smartphone embarcado

El cuarto prototipo es el más ambicioso de todos. Para la realización de este prototipo ha sido necesario unir dos proyectos de final de carrera.

Uno de ellos se ha centrado en la plataforma física, coche, motores, driver de potencia y en su activación-desactivación en función de las órdenes recibidas (esta parte se refiere a este trabajo final de carrera).

El otro proyecto se ha centrado en el desarrollo de la aplicación para android en la que se comunican dos smartphones entre ellos; el primero es el embarcado que envía los comandos a Arduino y también envía los datos de ubicación y streaming de video al smartphone de control, y el segundo es el de control que envía los comandos de control del coche al smartphone embarcado (que el smartphone embarcado enviará a Arduino), y recibirá el streaming de video y ubicación desde el smartphone embarcado. El proyecto es “control remoto de un coche con Android I” documento disponible en UPCommons. [ref 6]

4.1 Objetivo

El objetivo principal del cuarto prototipo es aprovechar la potencia de los smartphones y utilizar la cámara, el GPS y acelerómetros del Smartphone embarcado y transmitir estos datos al Smartphone de control, en manos del usuario. De esta manera podremos conducir el vehículo sin la necesidad de tenerlo en nuestra línea de visión y tener datos de la telemetría del coche en todo momento haciendo uso de la aplicación de Google Maps.

En el esquema de la figura 5.1 se aclara el funcionamiento de los 2 móviles; el embarcado y el de control.

Fig 4.1 Esquema de conexiones del cuarto prototipo

Una de las múltiples ventajas que nos aportará esta aplicación es que podemos conectarnos al móvil embarcado de dos maneras diferentes:

1. Creando un “Access Point” con el móvil embarcado al que nos conectaremos con la tablet. Esta conexión nos obliga a estar en un radio cercano al coche.
2. Conexión al móvil embarcado mediante redes móviles. Esta conexión nos permitirá conectarnos al coche desde cualquier lugar que tenga cobertura móvil siempre y cuando en el móvil embarcado esté ejecutándose el programa con la opción de móvil embarcado mediante “internet” que es la opción que se refiere a redes móviles.

4.2 Descripción de materiales y herramientas

Este prototipo necesita de 2 smartphones:

- El primer Smartphone irá embarcado en vehículo y estará emparejado mediante BT al Arduino que se encuentra en el coche. Nos referiremos a este Smartphone como “móvil embarcado” o “Smartphone embarcado”.
- El segundo Smartphone lo utilizaremos como mando del coche, este Smartphone o tablet se conectará al Smartphone embarcado y accederá

a su ubicación, tanto por redes móviles como por GPS como por geolocalización de ip. Además accederá a la cámara del Smartphone embarcado y será lo que nos muestre principalmente la pantalla de nuestro Smartphone de control o tablet.

La única herramienta de desarrollo en este prototipo ha sido el entorno de programación Eclipse, que nos permite hacer la aplicación soportada por el sistema operativo Android.

4.3 Integración del prototipo con otro proyecto

La integración de los dos proyectos ha requerido el trabajo conjunto, de equipo, entre los dos proyectistas. Para ello se planificaron 3 reuniones de coordinación, durante la primera reunión se discutió sobre los comandos que debe recibir Arduino para cambiar los estados de los pines, es decir, activar y desactivar motores, sobre cuando y como deben activarse, y también sobre el emparejamiento bluetooth-Arduino con el Smartphone.

En esta primera reunión establecimos que: la conexión bluetooth es serie y unidireccional (Arduino no envía datos sólo los recibe).

Los comandos que ha recibir Arduino establecidos durante esta primera reunión son los seis siguientes:

- Marcha adelante: el comando que espera Arduino es “**f**”.
- Marcha atrás: el comando que espera Arduino es “**b**”.
- Parar tracción: el comando que espera Arduino es “**v**”.
- Giro izquierda: el comando que espera Arduino es “**l**”.
- Giro derecha: el comando que espera Arduino es “**r**”.
- Ruedas de dirección rectas: el comando que espera Arduino es “**z**”.

En la segunda reunión se realizaron las pruebas de verificación.

En la tercera reunión realizamos el test final de conducción, se hicieron pruebas sobre el vehículo obteniendo los resultados deseados en lo que a la integración de proyectos se refiere.

En la figura 5.2 observamos las pruebas realizadas durante la tercera reunión y también el funcionamiento del programa:

A la izquierda de la imagen vemos el Smartphone de control o Tablet. En ella vemos 2 pedales (esquinas inferiores izquierda y derecha) que nos dará control sobre la tracción del coche, marcha atrás y hacia adelante cada pedal respectivamente. En la esquina superior derecha vemos el mapa de Google maps donde se encuentra el Smartphone embarcado y como fondo de pantalla tenemos el video de lo que está captando la cámara del móvil embarcado.

A la derecha de la imagen vemos el vehículo que utilizamos en nuestro primer prototipo que contiene: Arduino con antena BT y el móvil embarcado en el frontal del coche.

En el portátil se reproduce un video de prueba para poder testear el retardo en la recepción del video que hemos de ver en el Smartphone o en la tablet de control.

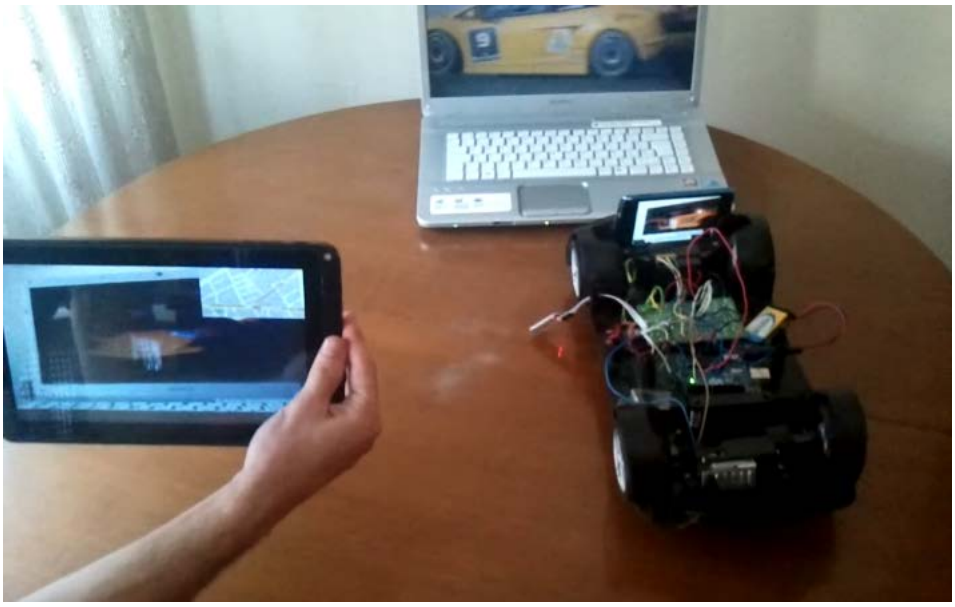


Fig 4.2 Imagen del cuarto prototipo funcionando durante una sesión de test.

5 Conclusiones

5.1 Valoración de los resultados

El objetivo de este trabajo ha sido explorar y presentar distintas formas de modificar un vehículo teledirigido por radiocontrol para que pueda ser controlado desde un dispositivo android. La finalidad es introducir en el mundo de la robótica a personas con diferentes niveles de conocimientos tanto en electrónica como en programación. Teniendo esto en cuenta, la realización de los 4 tutoriales ha sido satisfactoria ya que todo ha funcionado y ha dado resultados de una manera rápida.

El uso de Arduino como “autómata” ha facilitado mucho la realización de los prototipos ya que el lenguaje de programación para esta plataforma se basa en C/C++, un lenguaje de alto nivel sencillo de comprender.

Por otro lado, la programación en android no es tan sencilla, aunque usando el programa AppInventor del MIT nos simplifica mucho la programación de Android tiene bastantes limitaciones, como por ejemplo: no permite hacer un programa que se comunique vía serie con Arduino, nos obliga a saber la MAC del módulo BT para poder establecer la conexión Arduino-smartphone, …

Gracias al trabajo final de carrera de Víctor Nieto, en el que diseñó un programa para android, nuestro cuarto prototipo cumple con los requisitos que queríamos obtener al final de este trabajo final de carrera.

5.2 Propuesta de trabajos futuros

En futuros proyectos en los que se logre conectar vía serie Arduino con Android se pueden usar los sensores que tiene el Smartphone, acelerómetros, brújula y GPS y que los datos de éstos los procese Arduino y poder realizar algoritmos de rutas y mejoras en seguridad para el vehículo usando los acelerómetros.

La seguridad ha sido un problema, ya que cuando se pierde la conexión Android-Arduino, Arduino se queda en el último estado que ha recibido, normalmente “marcha hacia adelante” cosa que provoca que el vehículo siga recto hasta que encuentre un obstáculo. Este problema se habría resuelto con sensores de distancia que se pueden adquirir en el mercado y que debido a la falta de fondos no se han podido implementar haciendo que la seguridad del vehículo mejorase mucho ya que cuando se detectase un obstáculo lo suficientemente cerca Arduino entrase en el estado “parado”.

Y finalmente, destacar que en el cuarto prototipo, dependiendo de los modelos de Smartphone que se utilicen el video que se recibe en el Smartphone de control tiene un retardo que varía según los estos modelos. Este retardo hace que la “conducción” del vehículo mirando solamente la tablet tenga que ser muy cuidadosa ya que la imagen presenta cierto retardo. Sería interesante estudiar más profundamente mejoras relacionadas con streaming de video.

6 Referencias

1. Definición de Arduino de http://www.arduino.cc/es/
2. MOBOT BTCar, http://www.mobot.es/MobotBTCar.html
3. Arduino PWM http://arduino.cc/es/Reference/AnalogWrite?from=AnalogWrite.PWM
4. APPinventor http://appinventor.mit.edu/
5. Esquema del circuito extraído de; http://letsmakerobots.com/node/2074
6. Trabajo final de Carrera “Control remoto de un coche con Android I” https://upcommons.upc.edu/pfc/handle/2099.1/16277

Otras fuentes de información:

- Prácticas 3 y 4 con Arduino: control de un motor y un servomotor http://www.tecnosalva.com/pr%C3%A1cticas-3-y-4-arduino-control-motor-y-servomotor
- DC Motor Control Using an H-Bridge, http://itp.nyu.edu/physcomp/Labs/DCMotorControl#toc8
- Arduino Labs, http://labs.arduino.cc/ADK/GettingStarted
- Arduino Forum, http://arduino.cc/forum/index.php/topic,5859.0.html
- Amarino, Android metes Arduino, http://www.amarino-toolkit.net/index.php/home.html
- SkyBot, http://www.mobot.es/Skybot.html
- Fritzing beta projects, http://fritzing.org/projects/
- L293B datasheet, http://www.datasheetcatalog.org/datasheet/SGSThomsonMicroelectronics/mXurruu.pdf

Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANEXOS

TÍTULO DEL TFC: Introducción al control y telemetría de vehículos R/C mediante dispositivos Android

TITULACIÓN: Ingeniería Técnica Aeronáutica, esp. aeronavegación

AUTOR: Domingo Margareto Sanz

DIRECTOR: Marcos Quílez Figuerola

1 Anexo I

1.1 Código Arduino del primer prototipo extraído de la página de Mobot BTCar

```
/*
  MOBOT BTCar
  Author: Daniel Garrote
  Project: MOBOT
  Project URL: www.mobot.es
*/


int forward = 12;      // Pin 12 - Forward
int reverse = 11;      // Pin 11 - Reverse
int left = 10;         // Pin 10 - Left
int right = 9;         // Pin 9  - Right
int turbo = 8;         // Pin 8  - Turbo
int short_lights = 7;  // Pin 7  - Short Lights
int long_lights = 6;   // Pin 6  - Long Lights
int back_lights = 5;   // Pin 5  - Back Lights
int reverse_lights = 4;        // Pin 4  - Reverse Lights

char val;  // Variable to receive data from the serial port

void setup() {

  // initialize the digital pins as output
  pinMode(forward, OUTPUT);
  pinMode(reverse, OUTPUT);
  pinMode(left, OUTPUT);
  pinMode(right, OUTPUT);
  pinMode(turbo, OUTPUT);
  pinMode(short_lights, OUTPUT);
  pinMode(long_lights, OUTPUT);
  pinMode(back_lights, OUTPUT);
  pinMode(reverse_lights, OUTPUT);

  Serial.begin(9600);  // Start serial communication at 9600bps
}


// Fordward action
void go_forward() {
  digitalWrite(forward, HIGH);
  digitalWrite(turbo, LOW);
  digitalWrite(reverse, LOW);
}

// Stop Forward action
void stop_go_forward() {
  digitalWrite(forward, LOW);
}

// Reverse action
void go_reverse() {
  digitalWrite(reverse, HIGH);
  digitalWrite(forward, LOW);
  digitalWrite(turbo, LOW);
  digitalWrite(reverse_lights, HIGH);
```

```
}

// Stop Reverse action
void stop_go_reverse() {
  digitalWrite(reverse, LOW);
  digitalWrite(reverse_lights, LOW);
}

// Turbo action
void go_turbo() {
  digitalWrite(turbo, HIGH);
  digitalWrite(forward, LOW);
  digitalWrite(reverse, LOW);
}

// Stop Turbo action
void stop_go_turbo() {
  digitalWrite(turbo, LOW);
}

// Left action
void go_left() {
  digitalWrite(left, HIGH);
  digitalWrite(right, LOW);
}

// Right action
void go_right() {
  digitalWrite(right, HIGH);
  digitalWrite(left, LOW);
}

// Stop turn action
void stop_turn() {
  digitalWrite(right, LOW);
  digitalWrite(left, LOW);
}

// Stop car
void stop_car() {
  digitalWrite(forward, LOW);
  digitalWrite(reverse, LOW);
  digitalWrite(turbo, LOW);
  digitalWrite(right, LOW);
  digitalWrite(left, LOW);
  digitalWrite(reverse_lights, LOW);
}

// Short Lights ON
void lights_on() {
  digitalWrite(short_lights, HIGH);
  digitalWrite(back_lights, HIGH);
}

// Short Lights OFF
void lights_off() {
  digitalWrite(short_lights, LOW);
  digitalWrite(back_lights, LOW);
}

// Long Lights ON
```

```
void long_lights_on() {
  digitalWrite(long_lights, HIGH);
}

// Long Lights OFF
void long_lights_off() {
  digitalWrite(long_lights, LOW);
}

// Reverse Lights ON
void back_lights_on() {
  digitalWrite(reverse_lights, HIGH);
}

// Reverse Lights OFF
void back_lights_off() {
  digitalWrite(reverse_lights, LOW);
}

// Read serial port and perform command
void performCommand() {
  if (Serial.available()) {
    val = Serial.read();
  }
    if (val == 'f') { // Forward
      go_forward();
    } else if (val == 'z') { // Stop Forward
      stop_go_forward();
    } else if (val == 'b') { // Backward
      go_reverse();
    } else if (val == 'y') { // Stop Backward
      stop_go_reverse();
    } else if (val == 't') { // Turbo
      go_turbo();
    } else if (val == 'x') { // Stop Turbo
      stop_go_turbo();
    } else if (val == 'l') { // Right
      go_right();
    } else if (val == 'r') { // Left
      go_left();
    } else if (val == 'v') { // Stop Turn
      stop_turn();
    } else if (val == 's') { // Stop
      stop_car();
    } else if (val == 'a') { // Short Lights
      lights_on();
    } else if (val == 'c') { // Stop Short Lights
      lights_off();
    } else if (val == 'd') { // Long Lights
      long_lights_on();
    } else if (val == 'e') { // Stop Long Lights
      long_lights_off();
    }

}


void loop() {
  performCommand();
}
```

2 Anexo II

2.1 Código Arduino modificado.

Código introducido en Arduino para el primer prototipo. Los comandos de encendido y apagado de luces los usaremos para controlar la velocidad.

```
int forward = 2;       // Pin 2 - Adelante
int reverse = 3;       // Pin 3 - Atrás
int left = 4;          // Pin 5 - Izquierda
int right = 7;         // Pin 7  - Derecha
int velocidad = 5;     // Pin 5  - Velocidad


char val;  // Variable recibida del puerto serie

void setup() {

  // declaración de los pines como salidas digitales
  pinMode(forward, OUTPUT);
  pinMode(reverse, OUTPUT);
  pinMode(left, OUTPUT);
  pinMode(right, OUTPUT);
  pinMode(velocidad, OUTPUT);

  Serial.begin(9600);  // Start serial communication at 9600bps
}


// avanzar recto
void avanzar_recto() {
  analogWrite(velocidad, 200);
  digitalWrite(forward, HIGH);
  digitalWrite(reverse, LOW);
  digitalWrite(right, LOW);
  digitalWrite(left, LOW);
}

// Avanzar y girar derecha
void avanzar_derecha() {
  analogWrite(velocidad, 200);
  digitalWrite(forward, HIGH);
  digitalWrite(reverse, LOW);
  digitalWrite(right, HIGH);
  digitalWrite(left, LOW);
}

// Avanzar y girar izquierda
void avanzar_izquierda() {
  analogWrite(velocidad, 200);
  digitalWrite(forward, HIGH);
  digitalWrite(reverse, LOW);
  digitalWrite(right, LOW);
  digitalWrite(left, HIGH);
```

```
}

// atrás recto
void atras_recto() {
  analogWrite(velocidad, 200);
  digitalWrite(reverse, HIGH);
  digitalWrite(forward, LOW);
  digitalWrite(right, LOW);
  digitalWrite(left, LOW);
}

// atrás y giro izquierda
void atras_izquierda() {
  analogWrite(velocidad, 200);
  digitalWrite(reverse, HIGH);
  digitalWrite(forward, LOW);
  digitalWrite(left, HIGH);
  digitalWrite(right, LOW);
}

// atrás y girar derecha
void atras_derecha() {
  analogWrite(velocidad, 200);
  digitalWrite(reverse, HIGH);
  digitalWrite(forward, LOW);
  digitalWrite(right, HIGH);
  digitalWrite(left, LOW);
}

// giro de ruedas a derecha
void girar_derecha() {
  digitalWrite(right, HIGH);
  digitalWrite(left, LOW);
  digitalWrite(forward, LOW);
  digitalWrite(reverse, LOW);
}

//giro ruedas a izquierda
void girar_izquierda() {
  digitalWrite(right, LOW);
  digitalWrite(left, HIGH);
  digitalWrite(forward, LOW);
  digitalWrite(reverse, LOW);
}

// Frenar, activando los dos polos del motor de tracción
void freno() {
  analogWrite(velocidad, 255);
  digitalWrite(forward, HIGH);
  digitalWrite(reverse, HIGH);
  digitalWrite(right, LOW);
  digitalWrite(left, LOW);
}

//parar coche
void stop_coche() {
  analogWrite(velocidad, 0);
  digitalWrite(forward, LOW);
  digitalWrite(reverse, LOW);
}
```

```
//centrar ruedas
void stop_giro() {
  analogWrite(velocidad, 0);
  digitalWrite(right, LOW);
  digitalWrite(left, LOW);
  digitalWrite(forward, LOW);
  digitalWrite(reverse, LOW);
}

// lectura del puerto serie
void performCommand() {
  if (Serial.available()) {
    val = Serial.read();
  }
    if (val == 'a') {
      avanzar_recto();
    } else if (val == 'd') {
      avanzar_derecha();
    } else if (val == 'c') {
      avanzar_izquierda();
    } else if (val == 'e') {
      atras_recto();
    } else if (val == 'g') {
      atras_derecha();
    } else if (val == 'h') {
      atras_izquierda();
    } else if (val == 'r') {
      girar_derecha();
    } else if (val == 'l') {
      girar_izquierda();
    } else if (val == 's') {
      stop_coche();
    }
    else if (val == 'v') {
      stop_giro();
    }
}

void loop() {
  performCommand();

}
```

2.2 Código final de Arduino

Este es el código básico final utilizado en el tercer y cuarto prototipos.

```
int forward = 2;       // Pin 2 - Adelante
int reverse = 3;       // Pin 3 - Atrás
int left = 4;          // Pin 5 - Izquierda
int right = 7;         // Pin 7  - Derecha
int velocidad = 5;     // Pin 5  - Velocidad


char val;  // Variable recibida del puerto serie

void setup() {

  // declaración de los pines como salidas digitales
  pinMode(forward, OUTPUT);
  pinMode(reverse, OUTPUT);
```

```
  pinMode(left, OUTPUT);
  pinMode(right, OUTPUT);
  pinMode(velocidad, OUTPUT);

  Serial.begin(9600);  // Start serial communication at 9600bps
}


// avanzar recto
void avanzar_recto() {
  analogWrite(velocidad, 200);
  digitalWrite(forward, HIGH);
  digitalWrite(reverse, LOW);
  digitalWrite(right, LOW);
  digitalWrite(left, LOW);
}

// atrás recto
void atras_recto() {
  analogWrite(velocidad, 200);
  digitalWrite(reverse, HIGH);
  digitalWrite(forward, LOW);
  digitalWrite(right, LOW);
  digitalWrite(left, LOW);
}

// giro de ruedas a derecha
void girar_derecha() {
  digitalWrite(right, HIGH);
  digitalWrite(left, LOW);
  digitalWrite(forward, LOW);
  digitalWrite(reverse, LOW);
}

//giro ruedas a izquierda
void girar_izquierda() {
  digitalWrite(right, LOW);
  digitalWrite(left, HIGH);
  digitalWrite(forward, LOW);
  digitalWrite(reverse, LOW);
}

//parar coche
void stop_coche() {
  analogWrite(velocidad, 0);
  digitalWrite(forward, LOW);
  digitalWrite(reverse, LOW);
}

//centrar ruedas
void stop_giro() {
  digitalWrite(right, LOW);
  digitalWrite(left, LOW);
  digitalWrite(forward, LOW);
  digitalWrite(reverse, LOW);
}

// lectura del puerto serie
void performCommand() {
  if (Serial.available()) {
    val = Serial.read();
```

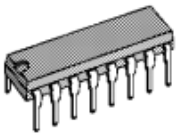
```
  }
    if (val == 'f') {
      avanzar_recto();
    } else if (val == 'b') {
      atras_recto();
    } else if (val == 'r') {
      girar_derecha();
    } else if (val == 'l') {
      girar_izquierda();
    } else if (val == 's') {
      stop_coche();
    } else if (val == 'v') {
      stop_giro();
    }
}

void loop() {
  performCommand();

}
```

2.3

3 Anexo III

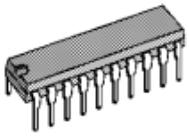
3.1 Datasheet del driver L293B

- 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

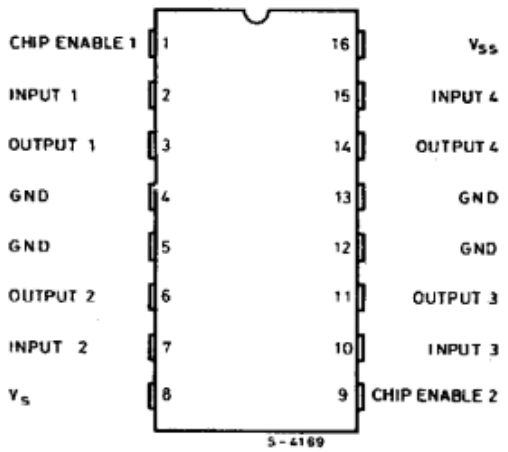
The L293B and L293E are quad push-pull drivers capable of delivering output currents to 1A per channel. Each channel is controlled by a TTL-compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off all four transistors. A separate supply input is 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.

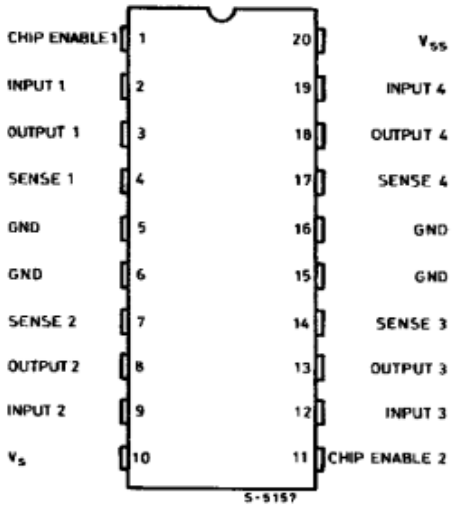
The L293B and L293E are package in 16 and 20-pin plastic DIPs respectively ; both use the four center pins to conduct heat to the printed circuit board.

PIN CONNECTIONS

DIP16 - L293B

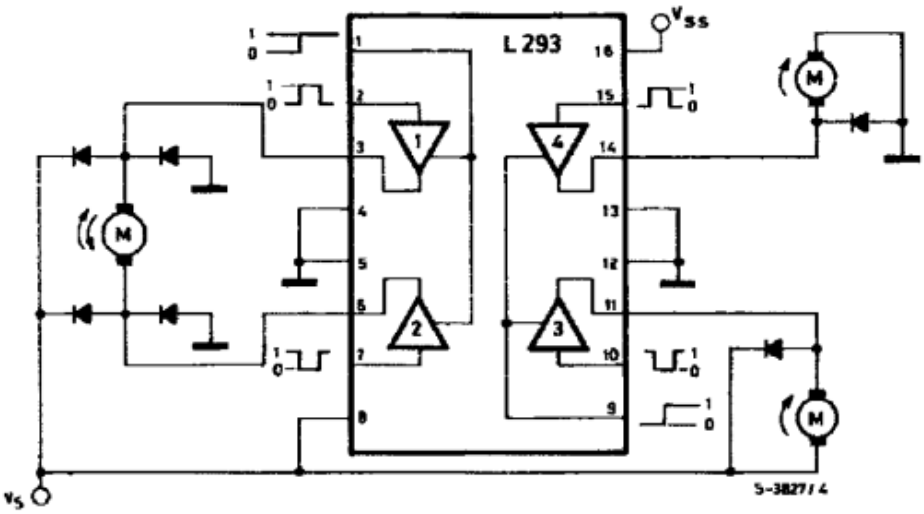


POWERDIP (16+2+2) - L293E



Block Diagrams

DIP16 - L293B



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 $T_{ground\text{-}pins}$ = 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_{SENS}	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