Trabajo de Final de Grado

Grado en Ingeniería en Tecnologías Industriales

Diseño y construcción de un vehículo de tres ruedas seguidor de trayectoria seleccionable

MEMORIA

Autor: Silvia Muñoz Souweine
Director: Rosa Rodríguez Montañés

Convocatoria: Junio 2016

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resumen

Este proyecto consiste en el diseño y construcción de un sistema electrónico basado en un microcontrolador que maneje el movimiento de un vehículo de tres ruedas, dos de ellas movidas por motores de corriente continua.

La trayectoria y el tipo de desplazamiento podrán ser determinados de forma automática o manual. Si el vehículo se encuentra en modo de conducción manual, el usuario es el que dirige los movimientos escogiéndolos de entre una serie de opciones predeterminadas mediante una aplicación para *smartphone* Android. Por otro lado, si se encuentra en modo de conducción automática, el propio vehículo mediante un sensor digital identificará diferentes colores que se le presenten a tal efecto y ejecutará, para cada color, una acción predeterminada.

El microcontrolador de 8 bits que se utiliza es de la familia PIC del fabricante Microchip. La comunicación entre el usuario y el vehículo se realiza mediante tecnología *bluetooth* por vía de un teléfono con sistema operativo Android. Se han escogido estos componentes por su disponibilidad y bajo coste económico.

Se ha diseñado una aplicación para dispositivo Android que permite dar instrucciones al vehículo utilizando el software App Inventor 2 del MIT.

En este documento se detallan los componentes utilizados para la construcción de este vehículo así como los pasos que se han ido siguiendo para implementarlo. También se explica la aplicación de Android que se ha creado para manejar el vehículo y la programación del microcontrolador.

Sumario

1. Glosario

Microcontrolador: Circuito integrado en un solo chip que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada/salida y periféricos.

Microprocesador: Procesador o CPU de pequeñas dimensiones. Contiene todos los elementos, es decir la CPU, memoria de datos, memoria de programa y los puertos de entrada/salida integrados en un solo chip.

Computadora: Máquina electrónica capaz de almacenar información y tratarla automáticamente mediante operaciones matemáticas y lógicas controladas por programas informáticos.

PIC (del inglés *Peripheral Interface Controller*): Familia de microcontroladores tipo RISC fabricados por Microchip Technology Inc.

RISC (del inglés *Reduced Instruction Set Computer*): Tipo de diseño de CPU utilizado generalmente en microprocesadores o microcontroladores que tiene dos características fundamentales. La primera, las instrucciones son de tamaño fijo y están presentadas en un reducido número de formatos y, la segunda, sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos.

CPU (del inglés *Central Processing Unit*): Parte central de una computadora u otros dispositivos programables que interpreta instrucciones y ejecuta operaciones básicas aritméticas, lógicas y de entrada/ salida del sistema. Sirve para procesar datos.

RAM (del inglés *Random Access Memory*): Memoria de acceso aleatorio de escritura y lectura. Se utiliza como memoria de trabajo en las computadoras.

ROM (del inglés *Read Only Memory*): Memoria programable no volátil que almacena instrucciones y datos de forma permanente.

FLASH: Tipo de memoria capaz de mantener la información una vez que se retira la fuente de alimentación.

EEPROM (del inglés *Electrically Erasable Programmable Read Only Memory)*: Tipo de memoria permanente de solo lectura que puede ser borrada eléctricamente.

USART (del inglés *Universal Synchronous Asynchronous Receiver Transmitter*): Es un dispositivo que sirve para transmitir o recibir datos secuenciales de manera síncrona o asíncrona.

UART (del inglés *Universal Asynchronous Receiver-Transmitter*): USART que solo maneja el modo asíncrono.

PWM (del inglés *Pulse Width Modulation*): Tipo de modulación de una señal o fuente de energía en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

LED (del inglés *light-emitting diode)*: diodo emisor de luz

Diodo: Dispositivo electrónico semiconductor de dos electrodos que permite el paso de la corriente eléctrica en una sola dirección y presenta las mismas características que un interruptor.

RGB (de las siglas en inglés de *Red*, *Green and Blue*): Se trata de un modelo de color basado en la síntesis. Es posible la representación de un color mediante la mezcla por adición de los tres colores primarios de luz, el rojo el verde y el azul.

2. Prefacio

2.1. Origen del proyecto

La autora de este trabajo está cursando el último año del grado en Ingeniería en Tecnologías Industriales en la ETSEIB (UPC) y, después de estudiar muchas asignaturas diferentes, la electrónica junto con la informática quizás son las que más interés le han despertado. Por este motivo, a la hora de elegir el tema del proyecto ha intentado unir ambas asignaturas construyendo un sistema electrónico que permita el control del movimiento de un vehículo a través de un dispositivo móvil o de forma automática según unas reglas preestablecidas.

2.2. Motivación

Durante este último año la autora de este trabajo se ha interesado por el mundo de la aviación no tripulada. Además de obtener el título de piloto de drones y de formar parte de la plantilla de una empresa dedicada a la grabación de vídeos aéreos con RPAS *(Remotely Piloted Aircraf System)*, ha tenido la oportunidad de ver la amplia oferta que supone esta nueva tecnología. Actualmente los drones están en auge y tienen muchas posibilidades que ofrecer al usuario.

Por lo tanto, la primera intención en el momento de decidir qué trabajo de final de grado iba a hacer fue pensar en un proyecto relacionado con un vehículo volador no tripulado pero al solicitar a la tutora su opinión y, vista la complejidad para realizar un vehículo de estas características, se optó por construir un vehículo más sencillo y, sobretodo, terrestre.

Para la autora, este proyecto sigue siendo un gran reto ya que toda la parte electrónica del presente trabajo servirá en un futuro para ampliar sus conocimientos y así poder aplicarlos a cualquier otro vehículo sea volador o no.

2.3. Requerimientos previos

Para poder realizar este proyecto se requería una serie de conocimientos previos de programación y de electrónica ya que este proyecto combina ambas tecnologías.

En este caso la autora era conocedora del lenguaje de programación *python* que es relativamente similar al lenguaje C con el que se ha desarrollado el trabajo ya que ambos son lenguajes de alto nivel. Los conocimientos previos de electrónica se adquirieron

inicialmente en una optativa que consistió en la construcción de una figura luminosa con LED's y su control mediante un microcontrolador PIC, aunque no se utilizó el mismo lenguaje de programación, en ella se entró en contacto con algunos de los componentes que se han utilizado en este proyecto que, aunque no han sido exactamente los mismos, tenían funcionamientos similares. Por otro lado, la base más sólida en conceptos electrónicos se ha adquirido en la asignatura obligatoria "Electrónica", cursada en paralelo con el desarrollo de este proyecto.

Referente a la parte de programación en este proyecto, no solo ha sido necesario saber de informática para programar el microcontrolador sino que también se ha tenido que diseñar una aplicación para un *smartphone* Android.

3. Introducción

3.1. Objetivos del proyecto

El objetivo principal de este proyecto es diseñar y construir un sistema electrónico basado en un microcontrolador que permita mover un vehículo terrestre de tres ruedas según las ordenes que reciba del exterior.

Se pretende controlar el movimiento del vehículo mediante dos tipos de órdenes, las que decide el usuario en base a una serie de opciones predeterminadas, y las que imponen los colores de unos indicadores mostrados al sistema según unos códigos establecidos con anterioridad.

Estos objetivos principales dan lugar a una serie de objetivos secundarios que se listan a continuación:

- Conocer y aplicar los conocimientos de electrónica impartidos en la universidad así como los conocimientos de informática para programar un microcontrolador.
- Construir un vehículo con motores de corriente continua que pueda desplazarse cuando se le da una orden desde un *Smartphone* o de forma automática cuando un sensor capta unos determinados colores.
- Implementar un sistema electrónico que permita la realización de los movimientos escogidos para el vehículo.
- Conseguir una forma funcional de comunicación entre el usuario y el vehículo intentando que la aplicación sea fácil de utilizar para el mayor número de usuarios posibles.
- Confirmar la importancia que tienen las nuevas tecnologías para conseguir retos tales como mover objetos desde un móvil.

3.2. Alcance del proyecto

En este apartado se detallan los condicionantes y limitaciones impuestos al trabajo desarrollado.

Por un lado, el material utilizado es el disponible en el laboratorio de electrónica de la ETSEIB y puesto a disposición de la estudiante para la ejecución del trabajo.

Por otro lado, la aplicación para el teléfono móvil se ha diseñado para el SO Android por ser

el más usado en los dispositivos móviles y porque permite la creación gratuita de aplicaciones.

La comunicación entre el microcontrolador y el dispositivo Android se realiza mediante *bluetooth*, por ser un protocolo de comunicación extendido y al alcance de la mayoría de usuarios.

Por último, para la programación se ha utilizado el lenguaje C por ser el lenguaje más utilizado para el tipo de microcontrolador escogido. Para la programación de la App se ha usado el software *MIT App Inventor 2*, ya que permite una forma de programación muy intuitiva y visual mediante el uso de bloques.

También hay que reconocer diversos problemas y dificultades. Por una parte, es obvio que los conocimientos en informática y electrónica de la autora de este proyecto son limitados. Por lo tanto, al final se ha tenido que construir un vehículo terrestre con un chasis prediseñado bastante sencillo. No obstante se han podido realizar todas las conexiones eléctricas necesarias para que el vehículo se desplace y, además, también se ha podido conectar con el móvil a través de *bluetooth.*

En este proyecto los movimientos del vehículo son básicos pero más adelante sería muy interesante poder ampliar las órdenes para conseguir movimientos mucho más difíciles y fluidos. Este vehículo supone un buen inicio para aprender y en un futuro poder realizar proyectos de más envergadura.

4. Especificaciones del sistema

En este apartado se detallarán las características escogidas para el sistema construido.

Al tratarse de un vehículo que estará en movimiento requiere de un chasis y una estructura que pueda soportar todos los componentes que precisa para moverse de forma independiente.

El coche cuenta con **tres ruedas**, dos de ellas movidas por motores de corriente continua y una tercera libre que sirve para equilibrar el **chasis** que consiste en una placa plana.

4.1. Movimientos

El vehículo debe ser capaz de realizar los cinco movimientos básicos de avanzar, retroceder, girar hacia la derecha, girar hacia la izquierda y parar. La elección del movimiento a realizar es escogida por el usuario, ya sea en modo de funcionamiento manual o automático. En el modo manual, la elección se realiza mediante las opciones a tal efecto de la aplicación Android. En el modo automático, la elección se realiza en base al indicador de color correspondiente a cada movimiento según la relación predeterminada en la programación del sistema.

4.2. Motores

Para poder controlar el movimiento del vehículo éste ha de constar de mínimo dos motores independientes para así poder crear una diferencia de velocidades entres las ruedas que permita el giro. En este caso cuenta con dos motores de corriente continua que requieren de una corriente de unos 200mA que es superior a la que puede otorgar el PIC utilizado (unas decenas de mA) y, por lo tanto, es imprescindible el uso de un **amplificador de corriente.**

4.3. Microcontrolador

El microcontrolador escogido es el PIC16F690, de 8 bits, del fabricante Microchip. Cuenta con un módulo UART para posibilitar la comunicación con el módulo externo *bluetooth* y también cuenta con un módulo PWM que será utilizado para el control de la velocidad de los motores.

Este dispositivo es el encargado de enviar las órdenes recibidas a cada uno de los diferentes componentes que tiene conectado en sus pines para que, finalmente, en conjunto

realicen la serie de instrucciones que el usuario quiere y por otra parte poder ejecutar el programa que se le ha guardado. Por lo tanto este dispositivo es el que procesa la información que le llega del exterior y hace que el vehículo actúe en consonancia.

4.4. Opciones de conducción

El coche tiene dos modos de conducción (manual y automática), por lo tanto, se han utilizado dos códigos de comunicación entre el exterior y el vehículo.

En el caso del modo manual se requiere un **teléfono Android** con una aplicación diseñada específicamente para la conducción del vehículo y que es capaz de enviar la información necesaria hasta el microcontrolador para determinar qué movimiento se le requiere ejecutar.

En el caso de la conducción automática basada en colores, aunque se activa desde la aplicación previamente mencionada, requiere de un **sensor de color** y de unas tarjetas indicadoras de los colores RGB para que el coche realice un movimiento programado asociado al color que identifica.

En ambos tipos de conducción se necesita un **módulo de *Bluetooth*** que permita la transmisión de información entre el teléfono móvil y el microcontrolador.

5. Arquitectura del sistema

En este capítulo se detalla la elección, la relación y el conexionado entre los módulos y las señales que forman el sistema diseñado.

5.1. Esquema funcional

A continuación se explicarán brevemente los dos esquemas funcionales en los que se basa el funcionamiento del vehículo diseñado.

5.1.1. Esquema del vehículo dirigido manualmente

En la *Fig. 5.1* se muestra el conexionado conceptual entre los módulos implicados en el control del movimiento del vehículo en modo manual.

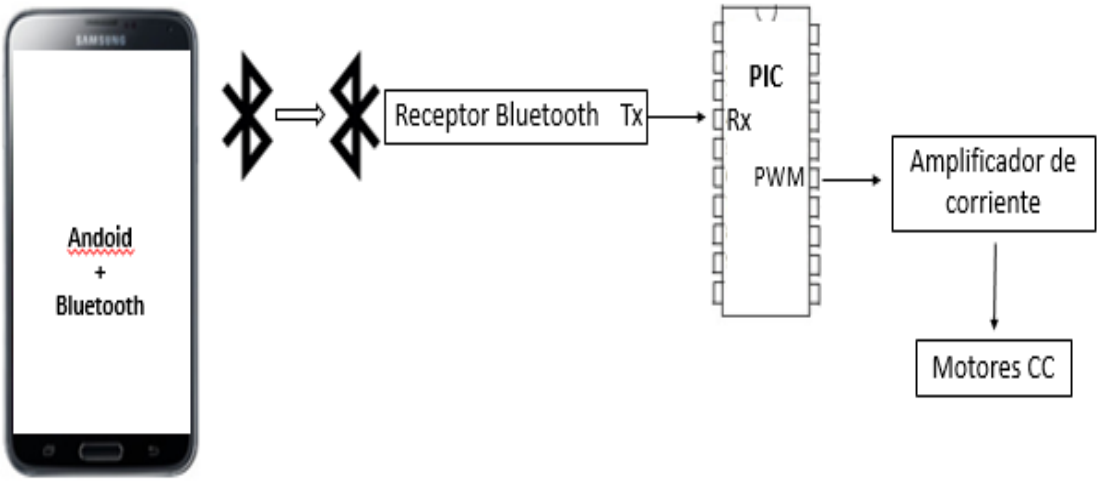

Fig. 5.1. Esquema funcional del vehículo dirigido en modo manual

Las acciones escogidas desde la aplicación de teléfono por el usuario son enviadas, vía señales, según protocolo y tecnología *bluetooth* al receptor situado sobre el vehículo. Este último transmite la información cambiándola a formato serie hacia el interior del PIC que contiene un programa que, al ejecutarlo, envía las órdenes pertinentes a los motores para que actúen en consecuencia.

5.1.2. Esquema del vehículo dirigido por colores

En la *Fig. 5.2* se muestra el conexionado entre los módulos implicados en el control del movimiento del vehículo en modo automático basado en colores.

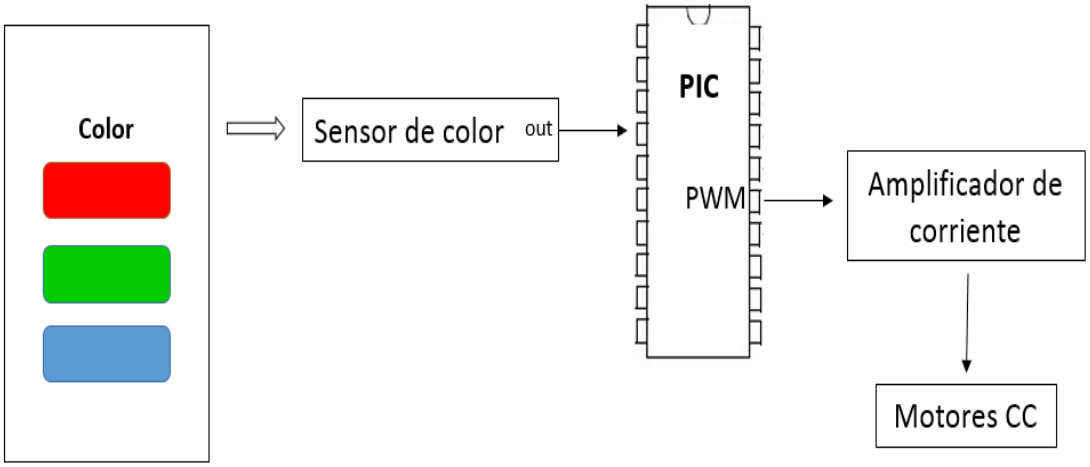

Fig. 5.2. Esquema funcional del vehículo dirigido en modo automático

En este tipo de conducción el usuario ha de escoger, en la aplicación, el modo de conducción automática. Una vez seleccionado este tipo de conducción el movimiento de los motores viene determinado por el color que detecta el sensor de colores y que transmite al PIC. El microcontrolador por el programa que tiene compilado ejecuta un conjunto de acciones en función del color recibido. Estas acciones son las que envía por medio de la corriente a los motores para que actúen en consecuencia.

5.2. Selección de componentes

Una vez visto el esquema funcional del sistema, se han decidido los componentes físicos que se usarán para construir el vehículo teledirigido. Se ha optado por estos componentes ya que, o bien se disponía de ellos con anterioridad como por ejemplo el teléfono utilizado, o porque estaban disponibles en la universidad y cumplían las características necesarias. Finalmente, los componentes que se han seleccionado son los que se pueden ver en la *Fig. 5.3.*

El usuario utiliza un Smartphone Samsung Galaxy S5 desde donde selecciona el tipo de conducción. Por una parte si se trata de conducción manual también selecciona los movimientos que desea que el vehículo realice. La información se envía vía *bluetooth* y se recibe en un dispositivo BlueCore4-Ext™ BC-417 que la modifica para que posteriormente

se procese mediante el PIC16F690. El PIC otorga un corriente insuficiente de salida que a continuación pasa por un amplificador de corriente L296D Motor Shield deek robot para poder poner en funcionamiento los motores de corriente continua.

Por otra parte si el usuario, a través de la aplicación, selecciona la conducción automática se activa un sensor de colores TCS3210. La información del sensor se procesa por el mismo PIC mencionado anteriormente.

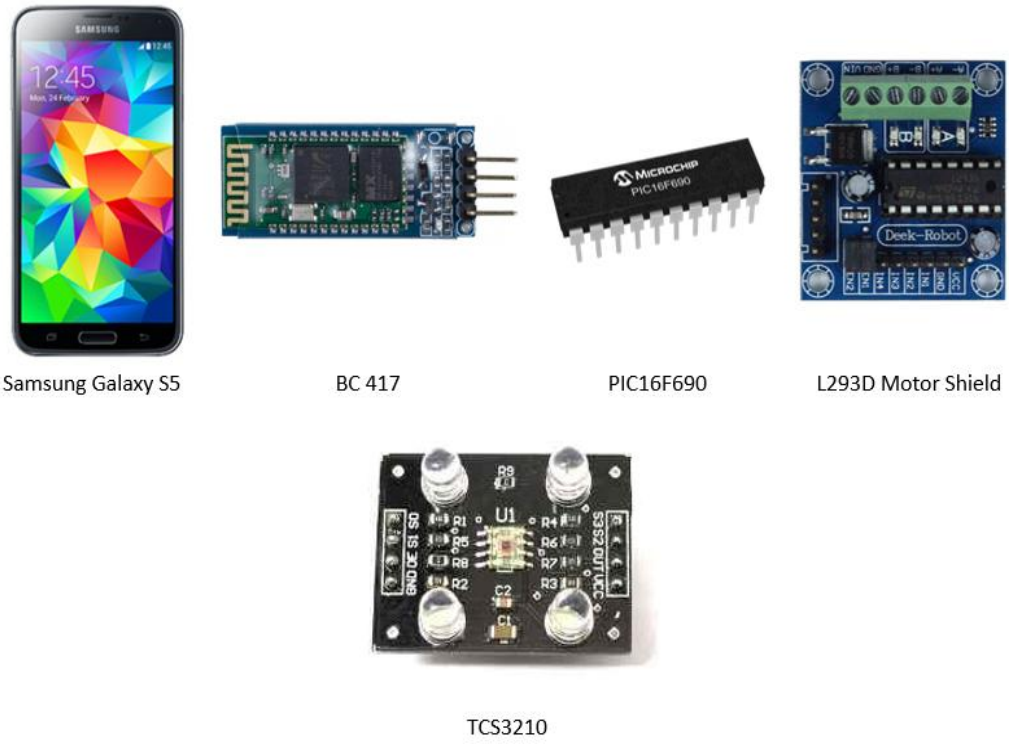

Fig. 5.3. Componentes principales utilizados

6. Componentes

En este capítulo se detallan las características básicas de los componentes utilizados para el desarrollo de este proyecto.

6.1. Microcontrolador PIC16F690

Este microcontrolador es de la familia PIC (del inglés *Peripheral Interface Controller*) de 8 bits de Microchip. Se trata de un dispositivo de gama media con 20 pines y de tecnología CMOS [1].

Se trata de un circuito integrado que en su interior contiene una CPU (del inglés *Central Processing Unit*), unidades de memoria, puertos de entrada/salida y periféricos. Todas estas partes se encuentran interconectadas entre sí dentro del dispositivo.

Fig. 6.1. PIC16F690

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	Comparators	Timers 8/16-bit	SSP	ECCP+	EUSART
	Flash (words)	SRAM (bytes)	EEPROM (bytes)							
PIC16F690	4096	256	256	18	12	2	2/1	Yes	Yes	Yes

Tabla 6.1. Características del PIC16F690

En esta *Tabla 6.1* se puede ver un resumen de los módulos más importantes y las dimensiones de éstos en el microcontrolador que se está estudiando.

6.1.1. Diagrama de bloques

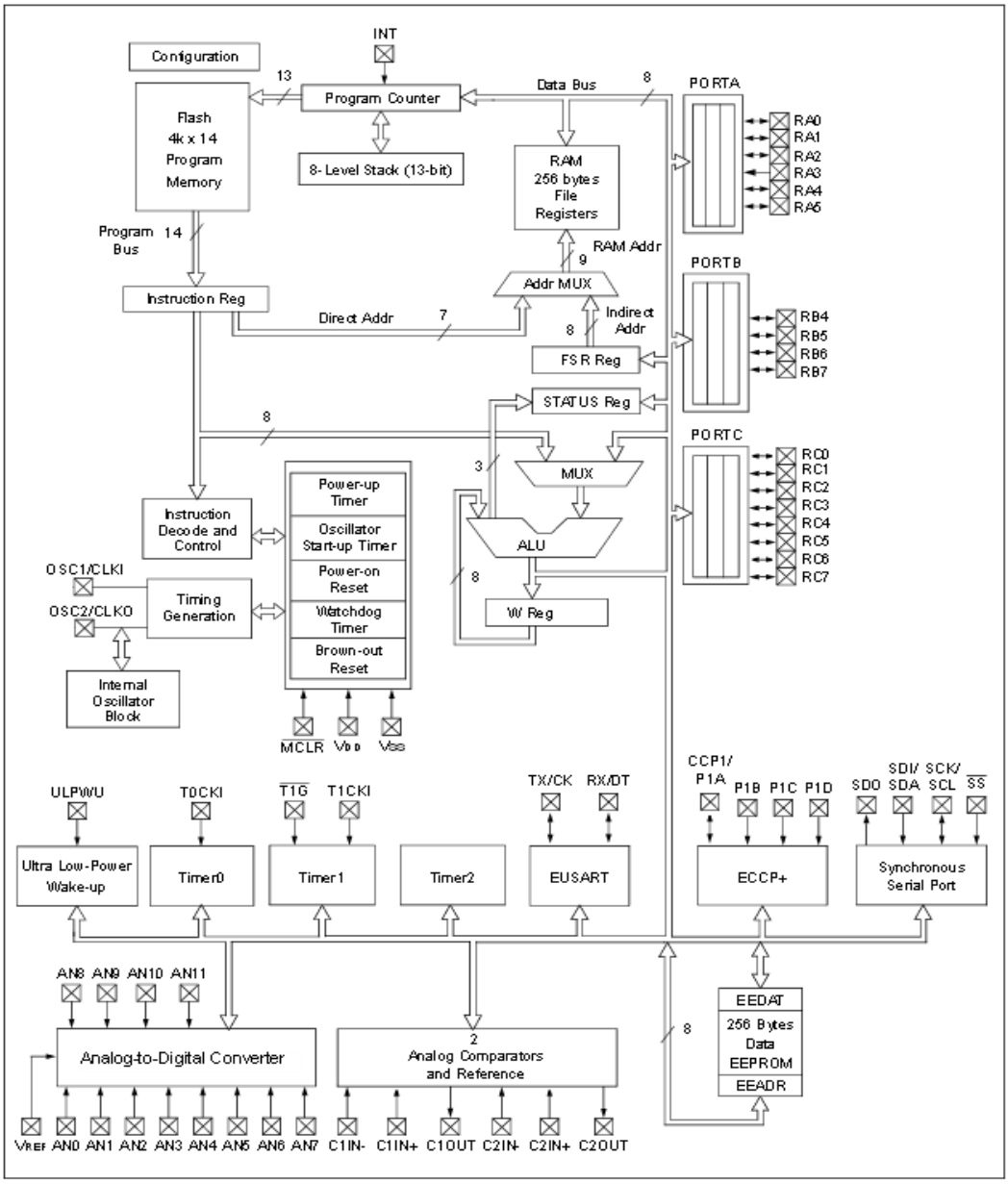

Fig. 6.2. Diagrama de bloques PIC16F690

Si se lee el diagrama de bloques *(Fig. 6.2)* de izquierda a derecha y de arriba hacia abajo primeramente se encuentra la memoria de programa Flash de 4k x 14 bits; el contador de programa PC; la pila subrutinas de 8 niveles; el bus de datos; la RAM; los puertos A,B y C; el bus de programa; la unidad de procesamiento aritmético/analógico ALU con sus registros asociados; el descodificador de instrucciones; el módulo de control; el oscilador interno; la unidad generadora de tiempos; los temporizadores 0, 1 y 2; la EUSART; el módulo de captura/comparación mejorado ECCP+; el puerto de comunicaciones síncronas SSI; el

módulo conversor A/D; el módulo de comparadores y por último la memoria EEPROM.

A continuación se explicarán los módulos de este microcontrolador que se han utilizado para la realización de este proyecto.

6.1.2. Memoria

Debido a que los datos que manejan los programas varían continuamente se necesita que la memoria que los contenga sea de escritura y lectura y por este motivo posee una memoria de datos RAM (del inglés *Random Access Memory*) que, en este microcontrolador, es de 256 bytes.

La memoria RAM es de tipo de celdas estáticas SRAM ("*StaticRAM*"). Las SRAM permiten almacenar la información un tiempo indefinido siempre que se mantengan alimentadas ya que son volátiles, es decir, que cuando se desconecta de la alimentación pierden la información que contenían.

El acceso a la memoria RAM está determinado por la memoria programable no volátil ROM (acrónimo en inglés *Read Only Memory*) de tipo Flash de 4K que permite la lectura y la escritura de múltiples posiciones de memoria en la misma operación. En esta memoria se graba el programa que consiste en una serie de instrucciones que se ejecutarán cuando el chip esté alimentado. Se utiliza este tipo de memoria ya que es reprogramable por el usuario que, mediante una orden de ordenador, puede borrar eléctricamente el contenido o modificarlo. Además la falta de alimentación no ocasiona la pérdida de información. Debido a que esta memoria de programa está integrada en el chip se accede a ella a través del bus de datos.

También posee una memoria permanente de datos EEPROM (acrónimo en inglés de *Electrically Erasable Programmable Read Only Memory*) de 256 bytes en la que se pueden almacenar datos durante la ejecución.

6.1.3. Procesador

Consta de un procesador RISC (del inglés *Reduced Instruction Set Computer*) que es un tipo de diseño de CPU y que se caracteriza por su conjunto de instrucciones reducido y sencillo (33 instrucciones en el caso del PIC16F690). Tiene una arquitectura de buses Harvard, es decir, consta de dos buses, uno de 14 bits que se usa para las instrucciones de programa y un segundo bus de 8 bits para los datos. Este tipo de arquitectura permite una mayor velocidad de ejecución ya que la lectura de su próxima instrucción se realiza mientras está ejecutando la instrucción actual. Cada instrucción necesita 4 ciclos de reloj para ejecutarse aunque las instrucciones de salto requieren el doble de tiempo ya que no se

solapan acciones. El reloj usado para la sincronización del sistema es el disponible internamente de 4 MHz.

6.1.4. Diagrama de pines del dispositivo

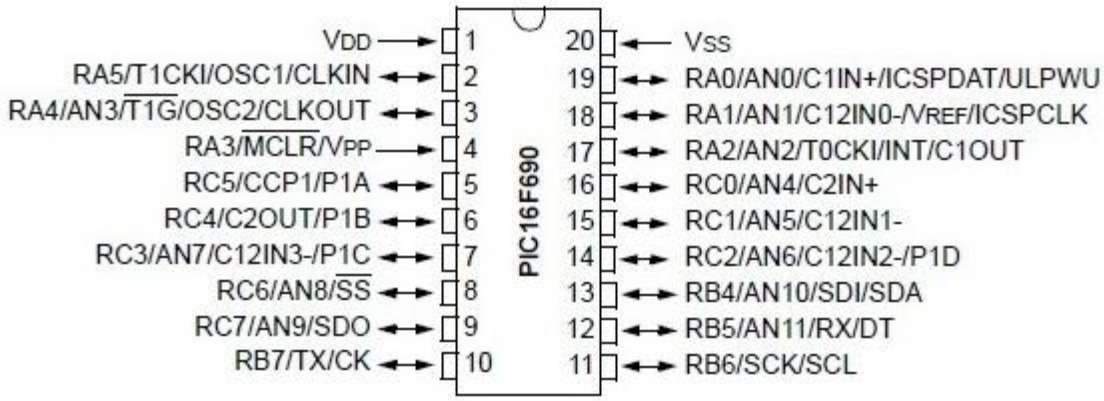

Fig. 6.3. Diagrama de pines del PIC16F690

Como se puede apreciar en la *Fig. 6.3* este PIC consta de 20 pines. Pero tiene más de 20 funciones distintas y esto conlleva a que algunos pines puedan incorporar más de una función pero no de forma simultánea. Así pues, en la configuración del dispositivo se ha de indicar qué función va a ejecutar el pin que se está conectando.

Este microcontrolador dispone de 3 puertos paralelos de entrada/salida denominados PORTA, PORTB y PORTC cada uno de ellos de 8 bits aunque no todos están disponibles. Estos puertos pueden ser de entrada cuando el microcontrolador lee datos del exterior, de salida si éste envía los datos hacia el exterior o bidireccional si la dirección del flujo de datos se puede configurar de salida o entrada a elección durante la ejecución del programa.

Los puertos también pueden ser digitales, analógicos o mixtos en función de los datos que intercambian. Se consideran puertos analógicos aquellos en los que se intercambian datos analógicos y requieren de un conversor analógico digital. Los puertos mixtos se configuran bit a bit por software para intercambiar datos analógicos o digitales. En este proyecto se han intercambiado datos digitales, por lo tanto, se ha trabajado con puertos configurados de tipo digital.

Este microcontrolador también posee una serie de pines para módulos específicos como, por ejemplo, el pin 5 donde se encuentra la salida del PWM (*Pulse Width Modulation*). En el pin 12 se encuentra la señal Rx de recepción de datos del módulo UART y en el pin 10 se encuentra la señal Tx que es el pin transmisor de datos del mismo módulo UART.

Para el funcionamiento del microcontrolador es necesario subministrarle una alimentación de entre 2,0V y 5,5V y también debe conectarse a 0V (GND).

6.1.5. Módulo PWM

El PWM (del inglés *Pulse Width Modulation*) es un módulo de control incorporado en el microcontrolador que genera una onda cuadrada con una frecuencia determinada por la siguiente fórmula *(Ec. 6.1)*

$$f_{PWM} = \frac{f_{osc}}{4xPREx(PR2+1)} \qquad \text{Ec. 6.1}$$

Donde:

f_{osc} ≡ Frecuencia del oscilador principal

PRE ≡ Divisor previo del TMR2, puede tomar los valores 1:1, 1:4 o 1:16

$PR2$ ≡ Registro asociado al TIMR2 puede ser un valor entre 0 y 255

Como el módulo PWM en este proyecto se utiliza para mover los motores se ha optado por el divisor de 16 y el registro de 255 para obtener una frecuencia de unos 250Hz, un valor favorable para los motores.

La onda generada por el pin de salida del PWM es periódica y se puede modificar su ciclo de trabajo (*Duty Cycle* en inglés) variando t_{on} *(Fig. 6.4)*. La variación de este tiempo nos permitirá darle mayor o menor potencia eléctrica, y por lo tanto, velocidad a los motores. En el apartado de implementación 8.1.2 se explica de forma detallada el PWM utilizado en este proyecto.

El *Duty Cycle* es un valor entre 0 y 1 que normalmente se da en porcentaje y que se obtiene de la siguiente operación *(Ec. 6.2)*.

$$D = \frac{t_{on}}{Period} \qquad \text{Ec. 6.2}$$

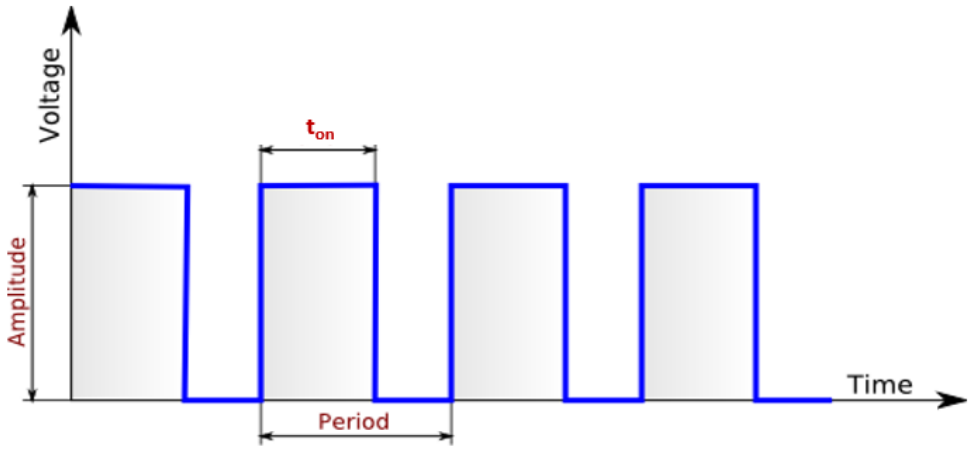

Fig. 6.4.Onda cuadrada PWM

6.1.6. UART

La UART (del inglés *Universal Asynchronous Receiver-Transmitter*) denominada en el caso del PIC16F690 como EUSART (del inglés *Enhanced Universal Synchronous Asyncronous Reciver-Transmitter*) tiene como función principal convertir los datos entre formato paralelo y formato serie para que puedan ser transmitidos/recibidos por los puertos hacia/desde el exterior. Toma bytes de datos internos y transmite los bits de forma secuencial hacia el exterior, o bien recibe los bits de forma secuencial del exterior y los agrupa al completar cada byte y los transmite internamente de forma paralela por el bus de datos En este caso se ha utilizado la UART como receptor de datos.

El patrón de funcionamiento se puede ver en la *Fig. 6.5,* la señal está, en su estado de reposo, a 5V hasta que empieza la transmisión, en ese momento se pone a 0V durante un cierto período de tiempo creando así el bit de inicio y el que informará al PIC que una nueva palabra de 8 bits llega a continuación. Seguido del bit de inicio llegan los 8 bits de la palabra de datos donde los 1's de la palabra dan una señal de 5V y los 0's de 0V. Así, estudiando la señal de entrada, se puede saber la combinación de 0's y 1's ordenados de menor a mayor peso que forman la palabra. Finalmente hay un bit de finalización donde la señal está a 5V durante un período de tiempo indicando así que se ha acabado la transmisión. Vemos en la *Fig. 6.5.* que también puede haber un bit de paridad que indica si el número de 1's que forman la palabra es par o impar, esto sirve como método de detección de errores.

Los períodos de tiempo que ocupan cada bit son aproximadamente unos 100 µs si la velocidad de comunicación es de 9600 baudios.

La palabra de 8 bits que se recibe se trata del número en forma binaria del valor numérico que se ha enviado por *bluetooth*, este número es de un valor entre 0 y 255.

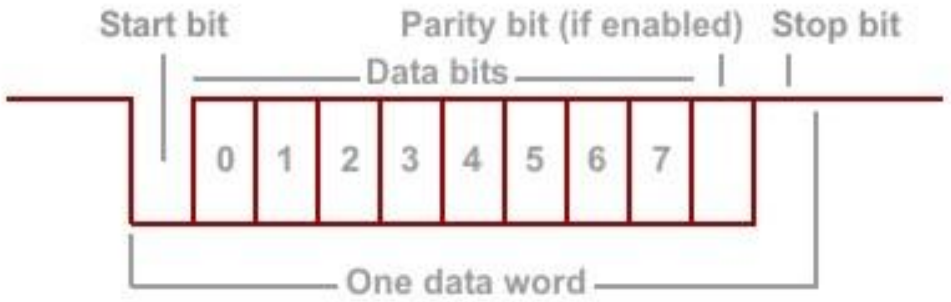

Fig. 6.5. Patrón de funcionamiento de la lectura de bits por la UART

6.1.7. Interrupciones

Las interrupciones son activadas por un módulo del microcontrolador que, una vez solicitadas, interrumpen la ejecución del programa principal. Una vez finalizada la respectiva rutina de interrupción, hay una instrucción especial que permite salir del subprograma y retomar el programa principal en el punto donde fue interrumpido. De la misma forma hay que habilitar las interrupciones globales y las de periféricos para poder utilizarlas mediante los registros y bits INTCONbits.GIE e INTCONbits.PEIE, respectivamente.

En el caso de este trabajo se ha creado una subrutina de interrupción para los puertos serie que se activa cada vez que se recibe un dato por el puerto Rx ya que, cada vez que el usuario desea cambiar la trayectoria del vehículo y selecciona una nueva acción en el teléfono, el vehículo ha de realizar el movimiento requerido en el mínimo tiempo posible.

Como se puede ver en la *Fig. 6.6,* tras la recepción del primer byte, se activa el Flag de interrupción y éste se desactiva una vez recibido el octavo bit del último dato recibido en la figura.

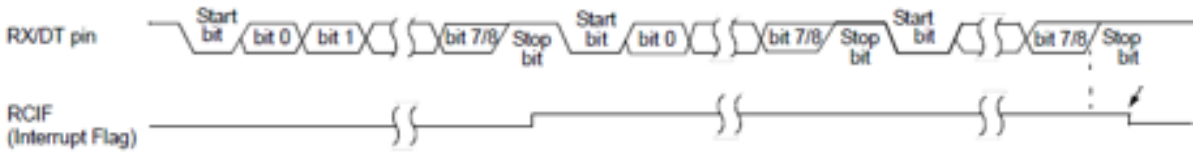

Fig. 6.6. Patrón de interrupciones PIC16F690

6.1.8. Oscilador

Todo microcontrolador requiere de un circuito, el oscilador de frecuencia, que le indique la velocidad a la que debe trabajar. El circuito oscilador es el que se encarga de generar una onda cuadrada de alta frecuencia que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema.

En el caso del PIC16F690 el circuito de reloj puede estar configurado por un oscilador externo o por uno interno. En este proyecto se ha configurado uno de los osciladores internos vía software escogiendo una frecuencia de 4MHz ya que posee un oscilador interno de este valor.

Como se ha comentado previamente, las instrucciones necesitan 4 ciclos de reloj para ejecutarse. De esta forma, si el microcontrolador funciona a 4 MHz, la frecuencia efectiva será de 1MHz aproximadamente.

Aumentar la frecuencia de reloj supone disminuir el tiempo en el que se ejecutan las instrucciones.

6.1.9. Temporizadores

El microcontrolador utilizado tiene tres temporizadores denominados TMR0, TMR1 y TMR2. Las principales aplicaciones para las que se utilizan en este proyecto son para generar retardos temporales, para medir la frecuencia de señales periódicas o bien para generar el PWM, respectivamente.

El TMR0 es un temporizador de 8 bits con *pre-scaler* de este microcontrolador que queda utilizado por la librería de los *delay*. El TMR2 de 8 bits con pre y *post-scaler* se utiliza para controlar el generador de PWM y el TMR1 de 16 bits, que puede trabajar con *pre-scaler* y oscilador independiente, se ha utilizado para el sensor de color.

6.1.10. Herramientas de programación MPLAB y PICkit 2

Para poder desarrollar un programa e introducirlo en el microcontrolador se han de utilizar una serie de herramientas y programas informáticos para poder escribir en la memoria del dispositivo. En el caso de los PIC se utiliza el software MPLAB que es el programa ensamblador/compilador que ofrece *Microchip Tecnology* de forma gratuita y también se hace uso del PICkit 2 que se ha utilizado como programador [3][6].

Fig. 6.7. Logo MPLAB e imagen del PICkit 2

El software MPLAB es un editor IDE (*Integrated Developement Enviroment*) gratuito para escribir y desarrollar código en lenguaje ensamblador o en el caso de este proyecto en C, lenguaje de alto nivel. La programación en C permite que el programa escrito pueda funcionar con mínimas modificaciones para otro microcontrolador de diferente arquitectura, por lo tanto se trata de un lenguaje reutilizable y portable. Además su escritura facilita el entendimiento.

El programa incluye varios módulos que permiten llevar a cabo las distintas etapas del proyecto.

La **edición** consiste en la escritura del programa en un archivo con la extensión *.c* y que permite la realización de correcciones.

El **preprocesador** realiza manipulaciones al programa denominadas directivas del preprocesador antes de compilarlo. Las directivas deben empezar por # para que el preprocesador las reconozca.

El **compilador** se encarga de pasar el programa C a código lenguaje de máquina que el microcontrolador pueda entender ya que están diseñados para interpretar y procesar datos e instrucciones en forma binaria, es decir patrones de 0's y 1's.

El **enlazador** enlaza el programa escrito con las diferentes librerías y datos referenciados en el código y crea una imagen ejecutable del programa archivo con la extensión *.hex* que se cargará en la memoria del microcontrolador.

Para cargar el programa en el microcontrolador se ha utilizado el PICkit 2. Este dispositivo se conecta al ordenador por vía USB y al PIC mediante la conexión a unos pines concretos detallados en la *Tabla 6.2 y la Fig.6.8.*

Pin PICkit	1	2	3	4	5	6
Pin PIC16F690	4	1	20	19	18	-
Función	$V_{pp}/\overline{MCLR}$	V_{DD}	GND	ICSPDAT/PGD	ICSPCLK/PGC	Auxiliary

Tabla 6.2. Conexión de pines del PIC con el PICkit 2

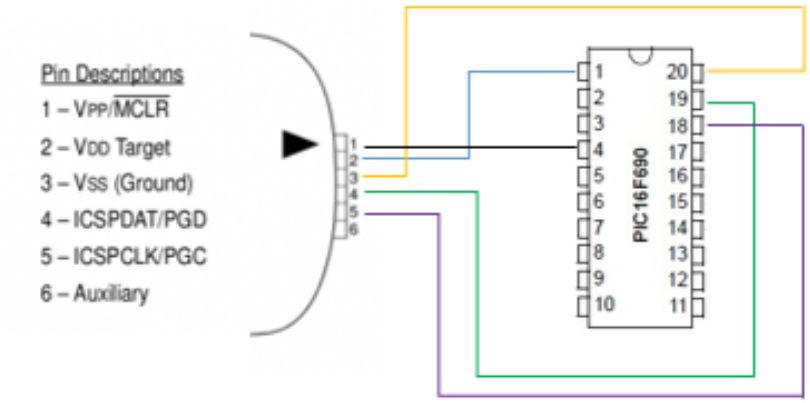

Fig. 6.8. Circuito de conexión del PIC16F690 con el PICkit 2

6.2. Bluetooth BC417

El BlueCore4-Ext™ BC417 permite enviar y recibir datos por vía Bluetooth [5]. En este proyecto se utiliza como receptor de los datos enviados desde el Smartphone y que, posteriormente, transmite al microcontrolador. Esto conlleva que se tenga que conectar el pin TX del BC417 con el pin 12 del PIC que es el receptor Rx. Además, el módulo Bluetooth se ha de alimentar a 5V para su funcionamiento y conectar a 0V (GND).

Dispone de otros 3 pines que no se han utilizado, el de Rx para cuando actúa de receptor de información, otro pin para alimentarlo a 3.3V y por último el pin Key que permite cambiar propiedades de configuración del módulo no necesarias para la realización de este proyecto.

Este dispositivo es el encargado de transformar el número en base decimal (entre 0 y 255) que recibe vía *bluetooth* y transformarlo a código binario para posteriormente comunicarlo al PIC.

6.3. L293D Motor Shield

Este dispositivo se ha utilizado para amplificar la corriente de salida del microchip adecuándola a un valor que permita mover los motores del vehículo (unos 200 mA).

Está compuesto por 4 módulos amplificadores diferenciados *(Fig. 6.9),* los cuales se pueden usar de manera independiente para controlar cargas de todo tipo y cualquiera de ellos sirve para configurar la mitad de un puente en H. De esta forma al tener 4 circuitos se pueden configurar 2 puentes en H completos que permiten el control de dos motores por sus salidas.

El chip tiene dos fuentes de alimentación separadas. Por un lado se encuentra la alimentación de la lógica es decir una conexión a 5V que alimenta al propio circuito integrado. Por otro lado se encuentra la alimentación de la carga que puede ser de un valor entre 4,5V y 36V.

Está diseñado para recibir niveles lógicos dónde los 1's corresponden a una tensión de 5V y los 0's a 0V. Con la señal recibida alimenta cargas inductivas, por ello consta de diodos de protección incorporados por donde se pueden descargar las bobinas. En el caso de este proyecto alimenta a dos motores de corriente continua.

La máxima corriente de salida que puede dar el L293D es de 600 mA por canal y puede soportar picos de hasta 1,2 A [4].

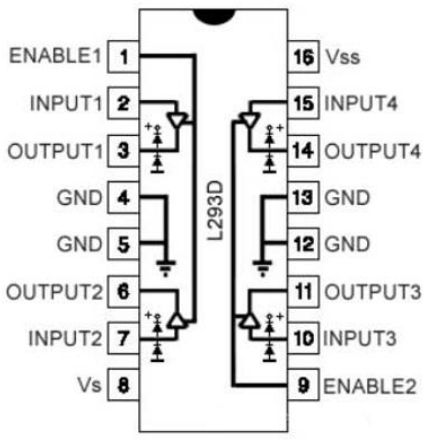

Fig. 6.9. Pines y circuito del L293D

6.4. TCS3210

El TCS3210 es un sensor de color programable que convierte la luz en frecuencia. Es capaz de filtrar los datos RGB de la luz y convertirlos a una onda cuadrada con frecuencia directamente proporcional a la intensidad de luz irradiada (*Fig. 6.11*). Tiene una matriz de 8x8 de fotodiodos donde 16 tienen filtros azules, otros 16 filtros verdes y 16 más tienen filtros rojos.

El dispositivo cuenta con dos pines, el S0 y S1, por los que se puede escalar la frecuencia de salida escogiendo entre 3 opciones: el 2%, el 20% o el 100 % de la frecuencia. También hay otros dos pines, el S2 y S3 por los que se controla el filtro RGB (*Fig. 6.10*).

También tiene un pin que permite habilitar y deshabilitar el sensor y por último tiene el pin de alimentación (5V) y el de tierra (GND) [2].

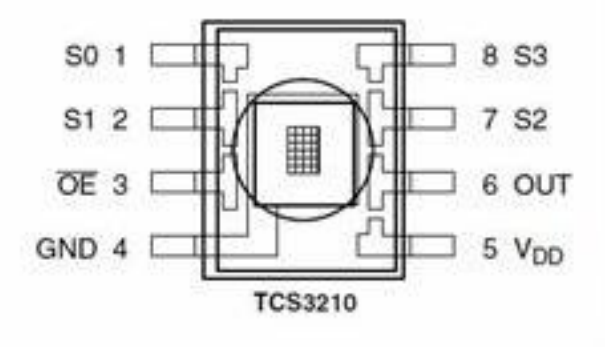

Fig. 6.10. Diagrama de pines del TCS3210

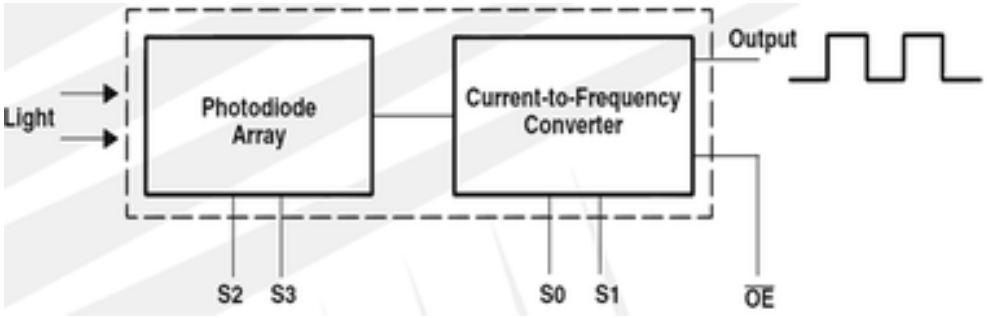

Fig. 6.11.Esquema simplificado de funcionamiento del TCS3210

6.4.1. Modelo RGB

El RGB (siglas en inglés de *Red, Green, Blue*) es un modelo de definición de colores usado en trabajos digitales tomando valores de 0 a 255. El rojo en RGB es el (255, 0, 0), el verde (0, 255, 0) y el azul (0, 0, 255) que son la composición del color en términos de la intensidad de los colores primarios de la luz. Es un modelo de color basado en la síntesis aditiva del color como se puede apreciar en la imagen (*Fig.6.12*) el blanco es la mezcla del rojo verde y azul.

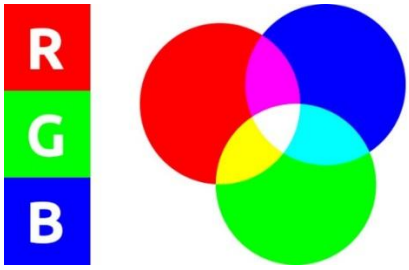

Fig. 6.12. Síntesis aditiva y colores RGB

7. Aplicación Android

La función de la aplicación creada consiste en establecer una vía de comunicación entre el usuario y el vehículo ya que mediante este soporte el usuario puede decidir los futuros movimientos que realizará el coche entre unos movimientos ya predeterminados y también escoger el modo de conducción entre automático o manual.

Se ha utilizado el software App Inventor 2 diseñado por MIT para desarrollar la aplicación con las diferentes opciones de movimiento ya que este software permite la creación de aplicaciones de forma fácil y gratuita.

MIT App inventor 2 es una herramienta de programación dividida en dos partes, por un lado se encuentra la parte de diseño y por otro lado la parte de los bloques.

7.1. Diseño

La **pantalla de diseño** *(Fig.7.1)* es donde se ubican y se escogen los diferentes elementos que compondrán el programa. Estos elementos pueden ser botones, cuadros de texto, imágenes y elementos de conexión tipo el *bluetooth*. En esta pantalla también se han de distribuir los elementos de la forma que se desea visualizar ya que, a excepción de algunos específicos que no se visualizarán, esta pantalla será la misma que el usuario obtendrá en su dispositivo al abrir la aplicación. También es en esta pantalla donde se define el formato de la letra y el aspecto de los botones.

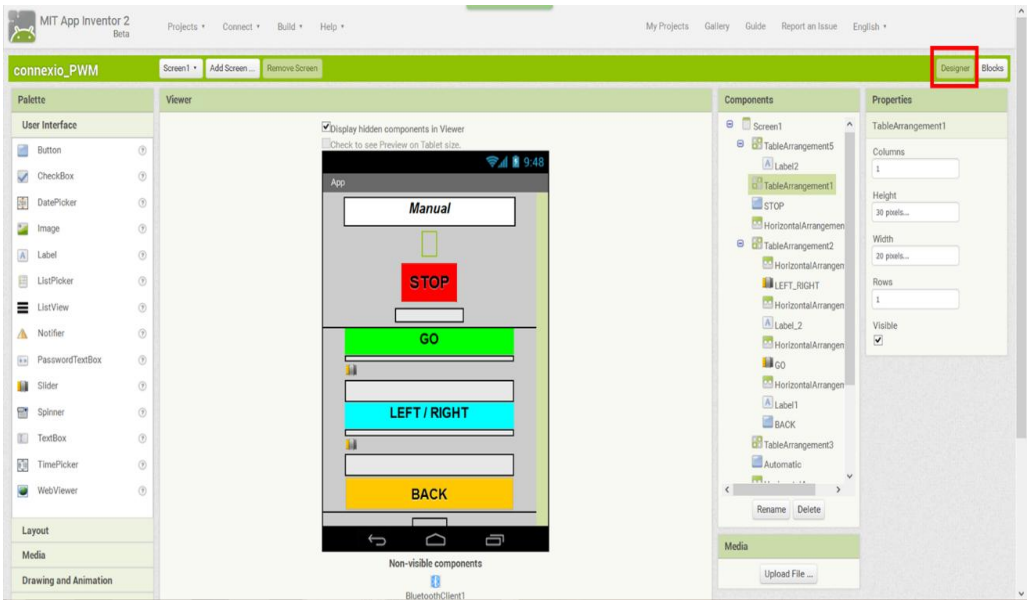

Fig. 7.1. Pantalla de diseño del software App Inventor 2

7.1.1. La aplicación en el teléfono

En la aplicación que se ha creado se han hecho diferentes pruebas para que sea un diseño lo más intuitivo posible para el usuario ya que ha de manipular la aplicación al mismo tiempo que el vehículo está en movimiento. También se intentado reducir al máximo el número de errores que puedan ocasionar la mala disposición de los botones. Finalmente se ha optado por la distribución que se puede ver en la *Fig. 7.2.*

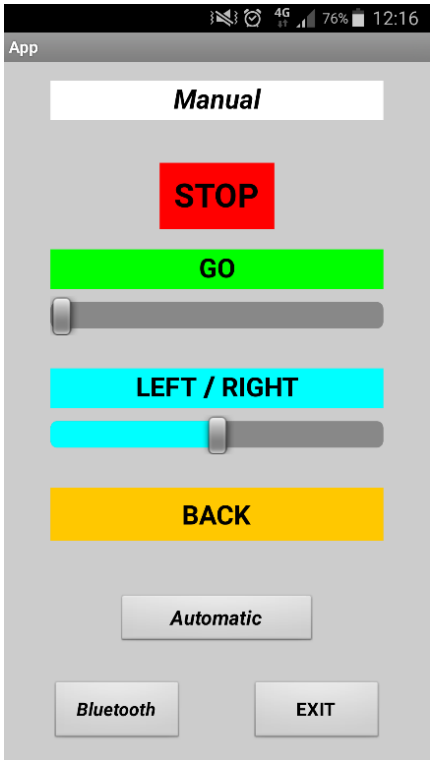

Fig. 7.2. Pantalla de la aplicación vista desde el Smartphone

Como se puede ver en la (Fig. 7.2) hay 5 botones y 2 barras deslizadoras. Las barras deslizadoras permiten aumentar gradualmente el valor. En esta aplicación se puede aumentar gradualmente la velocidad, en el caso de GO, o el grado y el sentido de giro en el caso del LEFT/RIGHT. El botón AUTOMATIC permite escoger el tipo de conducción automática. El botón de STOP detiene cualquier tipo de movimiento y desactiva la conducción automática, el botón BACK permite hacer marcha atrás a una velocidad determinada y sin posibilidad de giro. El botón de BLUETOOTH permite acceder a una lista de dispositivos disponibles y conectar con el módulo *bluetooth* del coche teledirigido y, finalmente, el botón EXIT permite salir de la aplicación.

7.2. Bloques

En la *pantalla de bloques (Fig. 7.3)* es donde se definen las funciones que realiza cada botón. El software tiene una serie de bloques prediseñados que realizan unas acciones predeterminadas. Mediante la unión de diferentes bloques se asocia una acción a un botón.

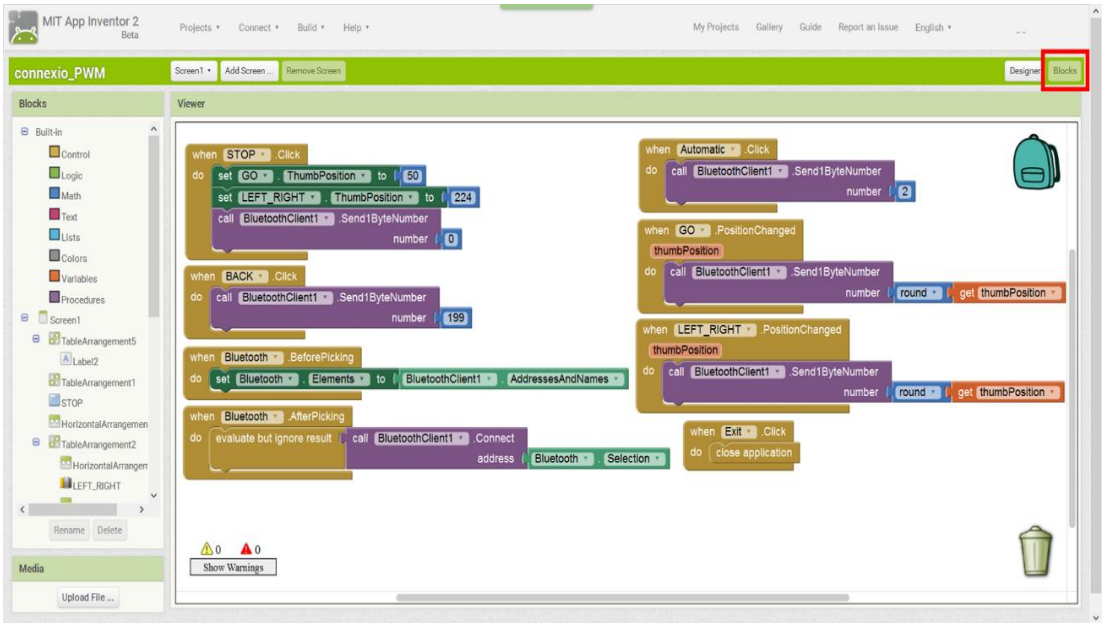

Fig. 7.3. Pantalla de bloques del software App Inventor 2

Los bloques marrones son bloques de control. Si se activa alguno de los botones al que están asociados se pasa a realizar la acción del bloque que contienen.

Si el bloque interior es de color verde es un bloque lógico, si es de color lila es un bloque de procedimiento que en este caso está asociado al *bluetooth.*

7.2.1. La aplicación creada

A continuación se va a explicar el procedimiento interno que realiza la aplicación al clicar cada uno de los botones o barras deslizadoras.

STOP

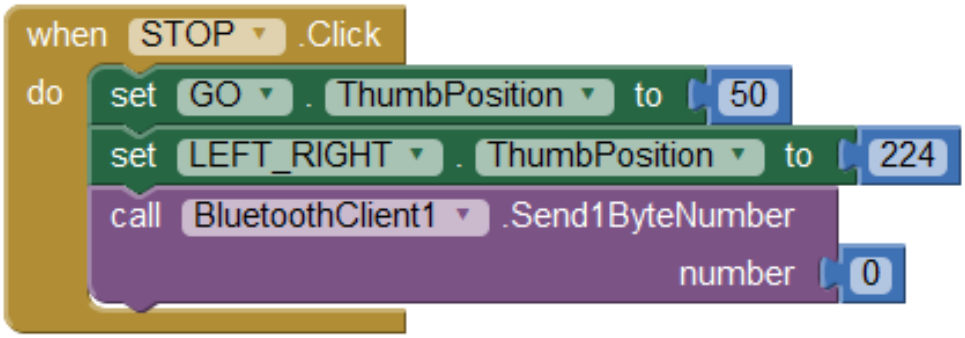

Fig. 7.4. Disposición de bloques para el botón STOP

Cuando se aprieta el botón de STOP los bloques lógicos que se representan de color verde son lo que dan la orden de poner las barras deslizadoras de la aplicación en su posición inicial es decir, se pone la barra de GO a cero, posición que se hace corresponder con el número 50 ya que es su valor mínimo para la funcionalidad de la aplicación, y la barra de LEFT/RIGHT en el centro (número 224) para que el coche no empiece el movimiento girando hacia ninguno de los dos lados. El porqué de la elección de estos valores numéricos se explica en capítulos siguientes.

Al apretar este botón también se manda por *bluetooth* el número 0 al microcontrolador que corresponde parar cualquier movimiento que se esté realizando y, también, a salir del modo de conducción automática. Acción que está representada por el bloque de color lila.

BACK

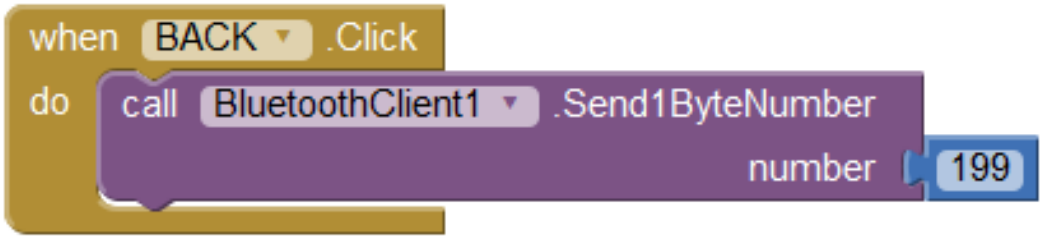

Fig. 7.5. Disposición de bloques para el botón BACK

Al apretar este botón se manda por *bluetooth* el número 199 al microcontrolador que corresponde a ir marcha atrás a una velocidad predeterminada sin posibilidad de giro.

AUTOMATIC

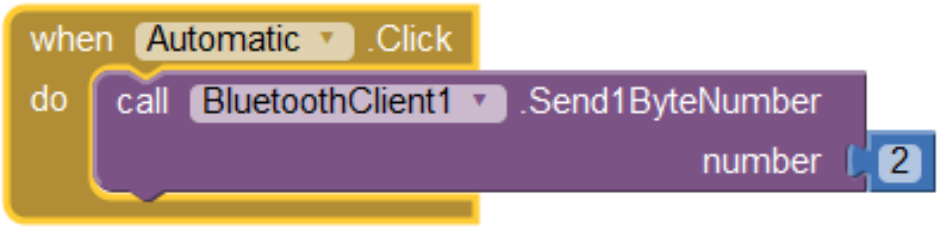

Fig. 7.6. Disposición de bloques para el botón AUTOMATIC

Al clicar este botón se manda por *bluetooth* el número 2 al microcontrolador que corresponde a entrar en conducción automática y habilitar el sensor. Procedimiento que se explicará más adelante en el apartado de programación 9.2.2.

BLUETOOTH

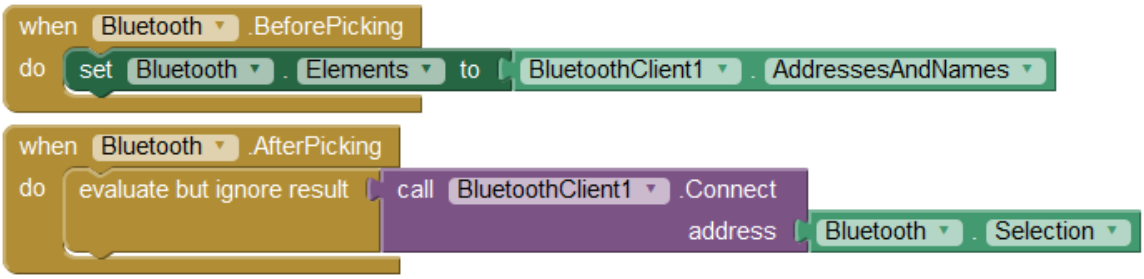

Fig. 7.7. Disposición de bloques para el botón BLUETOOTH

Hay dos bloques asociados a este módulo. El primero (*BeforePicking*) es antes de escoger el dispositivo *bluetooth* al que conectarse y el otro una vez escogido (*AfterPicking*). Antes se buscan los dispositivos cercanos y se obtiene un listado y una vez escogido el del vehículo se establece la conexión.

GO y LEFT/RIGHT

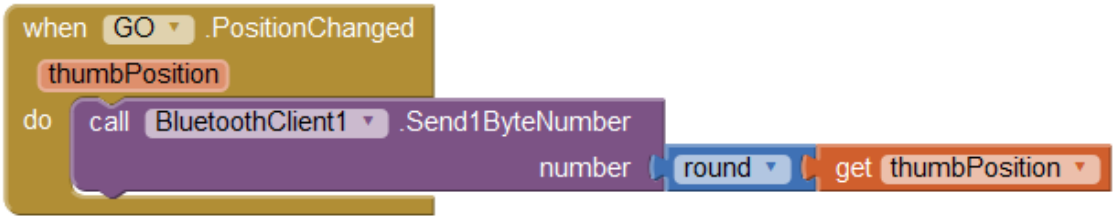

Fig. 7.8. Disposición de bloques para la barra deslizadora GO

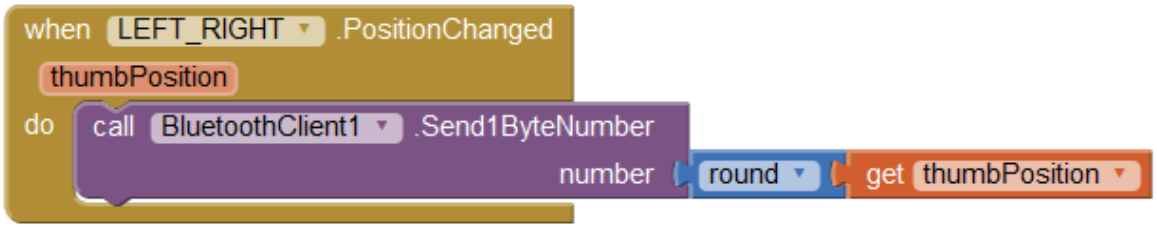

Fig. 7.9. Disposición de bloques para la barra deslizadora LEFT/RIGHT

Cuando se desliza el dedo por la barra se van obteniendo los valores que ésta comprende entre su máximo y su mínimo. Los valores aumentan gradualmente si se mueve el pulgar hacia la derecha y disminuyen si se mueve hacia la izquierda. El valor que se obtiene de la posición del pulgar ha de ser redondeado ya que el microcontrolador solo puede recibir números enteros de un valor, en este caso, de entre 0 y 255. Este valor numérico entero se envía por *bluetooth*.

Cada una de las barras tiene un valor mínimo y máximo que se corresponden a los que se han definido en el programa que está grabado en el microcontrolador.

EXIT

Fig. 7.10. Disposición de bloques para el botón EXIT

Cuando se aprieta este botón se cierra la aplicación.

8. Implementación

En este apartado de la memoria se explicará la metodología seguida para la construcción paso a paso del vehículo teledirigido.

El proyecto se ha ido realizando por fases y creando pasos intermedios para comprobar que cada componente funcione bien y realice correctamente las tareas que le son designadas. Para cada prueba se han ido creando distintos programas en código C, algunos de ellos se han podido reutilizar para el programa final.

Se ha dividido la implementación en dos partes:

La primera parte alcanza hasta que se ha implementado todo lo necesario para que el vehículo pueda moverse siguiendo las instrucciones indicadas por el usuario manualmente mediante la aplicación.

La segunda parte consiste en la implementación de todo lo relacionado con la conducción automática, es decir, con el sensor de colores.

8.1. Primera parte. Pruebas previas. Conducción manual

Primeramente, para entrar en contacto con el lenguaje de programación y la conexión de elementos en un *protoboard,* se optó por crear un programa simple de encendido de LED's con el que se aprendió a programar y a configurar el PIC utilizado.

También se aprendió a utilizar el MPLAB y se tomó un primer contacto con el lenguaje de programación C y con los pasos a seguir para compilar el programa y grabar el fichero ejecutable en el microcontrolador. Por otro lado se crearon los primeros circuitos físicos de conexión con el PICkit2 y los correspondientes LED's *(Fig. 8.1)*

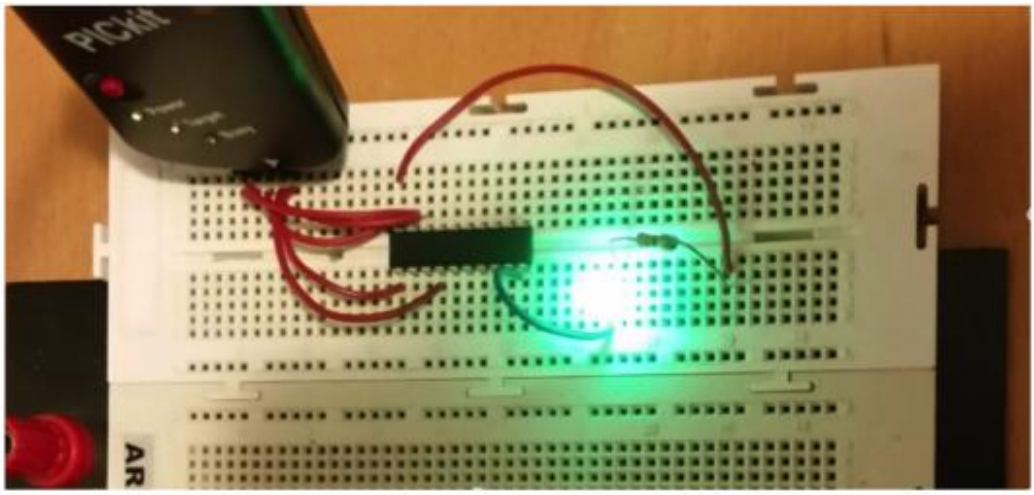

Fig. 8.1. Circuito físico de encendido de un LED y conexión con el PICkit 2

8.1.1. Comprobación de la conexión vía Bluetooth

Posteriormente, se creó un circuito con 8 LED's donde cada uno de ellos estaba conectado a un pin distinto del microcontrolador. En concreto, se ocupó el Puerto C que se configuró como salida. Al pin Rx del PIC se conectó el dispositivo *Bluetooth* BC417 mediante su pin Tx y éste se alimentó a una tensión de 5V y se conectó también a GND *(Fig.8.2 - A).*

Para la correcta comunicación entre la UART del PIC y el BC417 se han tenido que configurar ambos a una misma velocidad que, en este caso, ha sido de 9600 baudios, es decir, que el período aproximado que ocupa cada bit es de 100 µs. Se ha escogido esta velocidad de transmisión por ser ampliamente utilizada y adecuada para las necesidades de este proyecto.

Para poder enviar información al dispositivo *Bluetooth* BC417 se creó una primera aplicación con el software App Inventor 2, previamente mencionado, que permitía enviar valores numéricos seleccionados de entre un rango disponible.

El BC417 tiene como función recibir datos de 8 bits, de un valor de entre 0 y 255, vía protocolo *bluetooth* y transmitirlo al PIC en formato binario secuencial. Para comprobar que se estuviese recibiendo y transmitiendo de forma correcta la información se utilizaron los 8 LED's dispuestos como se ha mencionado previamente ya que de esta forma cada LED representaba un bit del número en binario que se estaba enviando. Para poder asociar los LED's al número en binario se tuvo en cuenta su disposición ya que se pusieron por orden el LED de más a la izquierda es el que representa el bit de mayor peso (msb) y el de más a la derecha el de menor (lsb) *(Fig.8.2 - A).* En las imágenes B y C *(Fig.8.2– B y C)* se pueden ver numerados del 0 al 7 los LED's donde el 7 es el bit de mayor peso y van disminuyendo hasta el 0 que es el bit de menor peso.

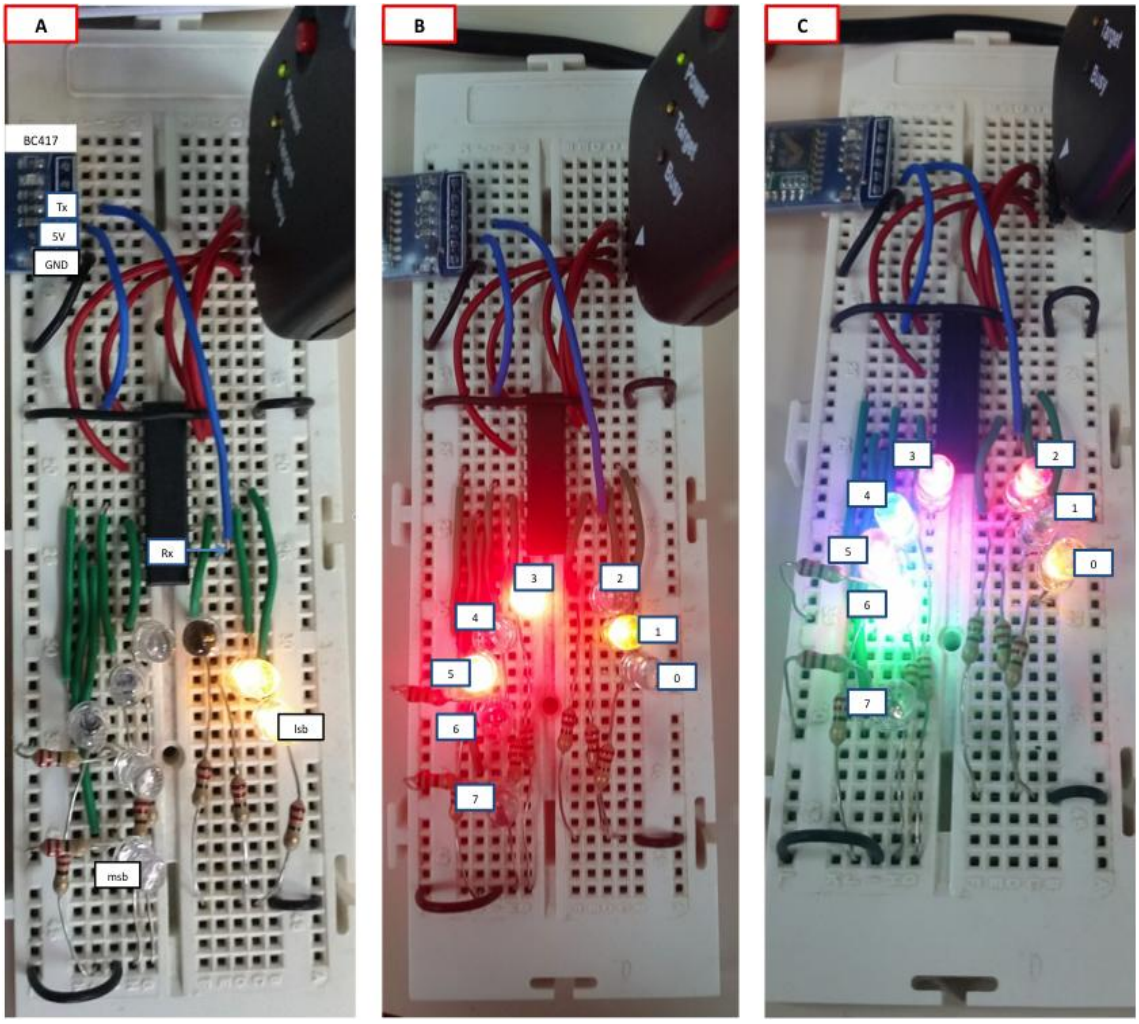

Fig. 8.2.Circuito de comprobación de la conexión bluetooth

En la imagen 8.2-A se puede visualizar el número 3, que en binario corresponde al 00000011.

En la imagen 8.2-B se puede visualizar el número 42, que en binario corresponde al 00101010.

En la imagen 8.2-C se puede visualizar el número 125, que en binario corresponde al 01111101.

En la siguiente tabla *(Tabla 8.1)* se recoge el estado de los LED's para los valores previamente mencionados.

Si el valor es un 0, el LED se encuentra apagado y si es un 1, encendido.

Número decimal	Bit/LED 7 (msb)	Bit/LED 6	Bit/LED 5	Bit/LED 4	Bit/LED 3	Bit/LED 2	Bit/LED 1	Bit/LED 0 (lsb)
3	0	0	0	0	0	0	1	1
42	0	0	1	0	1	0	1	0
125	0	1	1	1	1	1	0	1

Tabla 8.1. Valores en binario.

8.1.2. Pruebas módulo PWM y función PWM de creación propia

Como se ha explicado previamente el PWM se trata de un módulo del PIC que, en este caso, controla la velocidad de los motores mediante la variación del valor del ciclo de trabajo de una onda cuadrada.

En este proyecto, debido a la necesidad de controlar dos motores por separado y a la existencia de un único pin con esta función en el microcontrolador, se ha tenido que crear un PWM a mano o propio. Éste se ha hecho mediante una función programada, a la que se le han ido ajustando los valores de forma experimental (debido a la pérdida de información temporal introducida por el compilador).

Este segundo PWM ha de devolver una onda con unas características prácticamente iguales o lo más parecidas posibles a la que crea el PWM del PIC para que los motores tengan la misma velocidad en todos los valores que se envíen posteriormente vía Bluetooth.

Por otro lado, la velocidad de giro dependerá de la tensión media que irá variando según el valor que se reciba como instrucción. De esta forma, como los motores trabajan correctamente a una frecuencia de 250 Hz, se ha buscado este valor en ambos PWM.

Como ya se ha visto, el ciclo de trabajo de este módulo se calcula como el tiempo en el que se alimenta a tensión alta (t_{on}) dividido por el período total de trabajo *(Period) (ver Fig. 8.3).* De esta forma se ha optado por asociar el valor enviado vía *bluetooth* con el t_{on} y así, a mayor valor introducido en esta función, mayor potencia conseguida, es decir, mayor velocidad de giro para ambos motores.

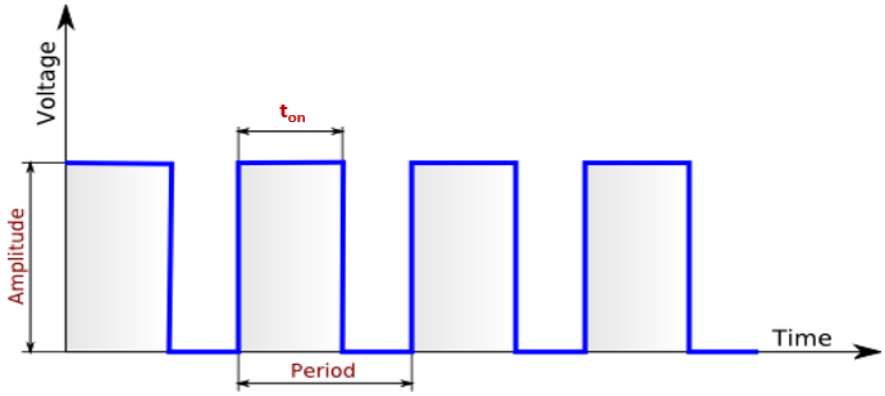

Fig. 8.3. Onda cuadrada PWM

Se podría decir que se ha dividido un período en 255 intervalos. La onda se encontrará en su valor alto tantos intervalos como número se introduzca en la función (t_{on}), y en su valor bajo la diferencia de este valor a 255.

Se han realizado diferentes medidas en el laboratorio con el osciloscopio para poder ajustar al máximo el parecido de las dos ondas, es decir que tengan un valor de tensión medio y de frecuencia parecido.

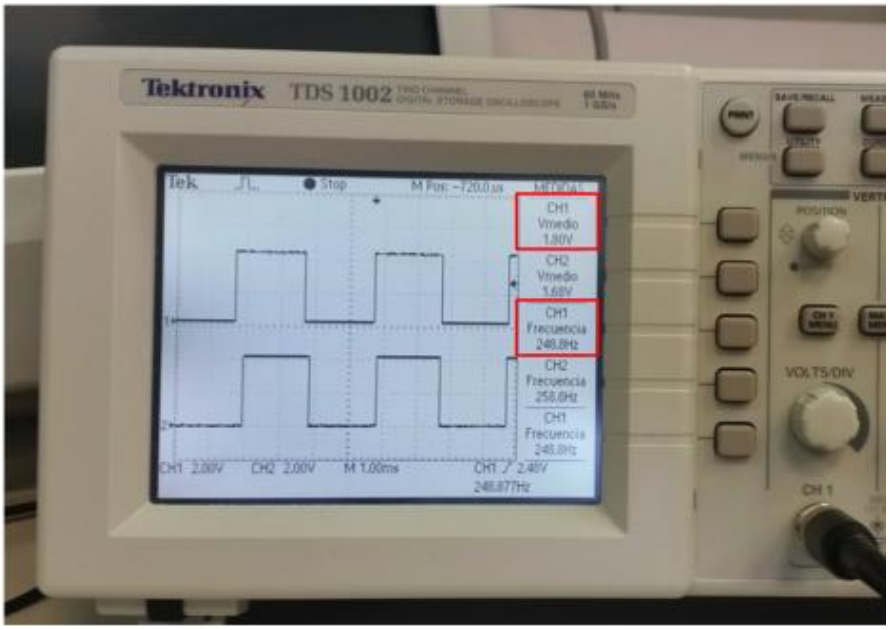

Fig. 8.4. Ondas del PWM y valores del PWM generado por el PIC

En la *Fig. 8.4* se pueden ver las dos ondas que son prácticamente iguales. En la parte superior se encuentra la onda que genera el PWM del propio PIC y sus valores resaltados en rojo. También se puede ver la frecuencia que es aproximadamente el valor buscado de 250 Hz y la tensión media es de 1,8V.

En la siguiente imagen *(Fig. 8.5)* se pueden apreciar de nuevo las dos ondas y en rojo los valores del PWM que se ha creado por *software* para este proyecto. Se presentan los valores numéricos en dos imágenes distintas ya que el osciloscopio solo permite el cálculo con mejor precisión de la tensión media de una de las dos ondas cada vez.

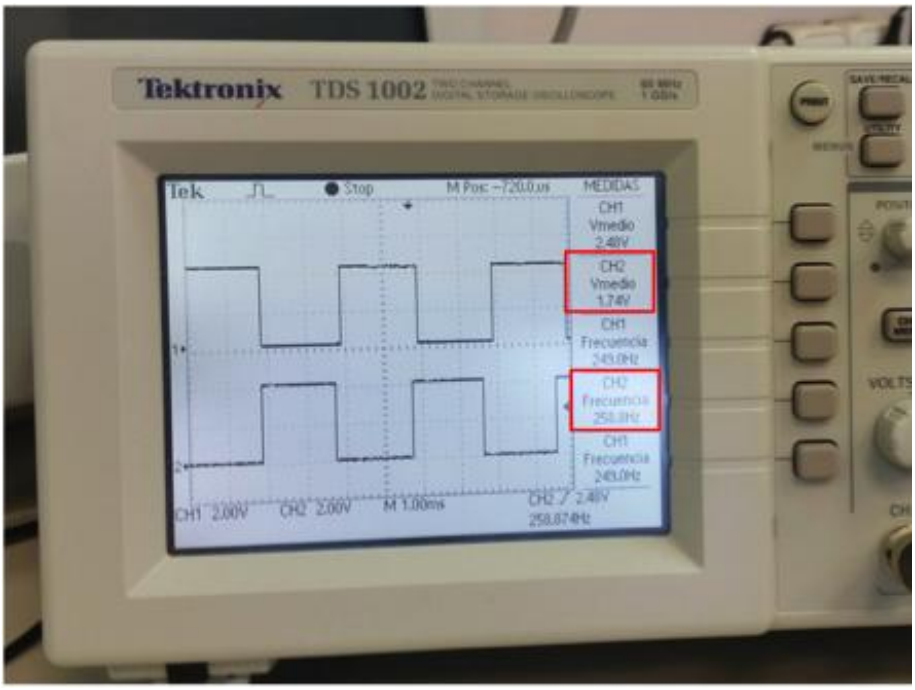

Fig. 8.5. Ondas del PWM y valores del PWM propio

En esta segunda imagen se puede comprobar que, aunque hay una cierta diferencia con respecto a la onda generada por el módulo PWM del PIC, la tensión media es de 1,74V y la frecuencia de 258,8 Hz que son valores muy próximos y suficientes para la finalidad del trabajo.

Como en este proyecto es necesario que los motores giren en ambas direcciones, para que el coche pueda circular hacia delante y hacia atrás, se han habilitado 4 pines del microcontrolador para el control de estos. De esta forma, dos pines (el 5 y 15) corresponden a la salida de los PWMs, el propio del PIC y del que se ha creado. La tensión de los otros dos pines sirve para escoger el sentido de la marcha, es decir que si se quiere que el coche circule hacia delante los pines (pines 6 y 16) están a 0V. En cambio, cuando se requiere la marcha hacia atrás, estos pines otorgan 5V y de esta manera el efecto de las señales de los PWMs queda invertido generando el sentido de giro de los motores en dirección contraria.

Otro aspecto a tener en cuenta es que el microcontrolador no puede generar corrientes de valores elevadas, por ello, se requiere un amplificador de corriente para poder alimentar los motores de forma correcta.

8.1.3. Amplificación de la corriente

Para el correcto funcionamiento de los motores estos requieren de una corriente de alimentación de 200mA. Como el PIC solo es capaz de otorgar 50 mA se ha conectado a la salida de los pines previamente mencionados el amplificador L293D detallado en el apartado de componentes *(apartado 6.3).*

El amplificador necesita dos alimentaciones separadas. Una de ellas es de 5V y es la tensión de trabajo de la parte lógica del propio chip. La otra tensión que recibe es la que directamente controla los motores y tiene un valor entre 4,5V y 36V que se obtiene de las pilas. En el proyecto se ha escogido 6V para esta tensión, procedente de 4 pilas AA. Las características de estas pilas permiten el suministro de los 2x200mA requeridos por los motores.

Por lo tanto se necesitan las conexiones que se pueden apreciar en la imagen *(Fig. 8.6).*

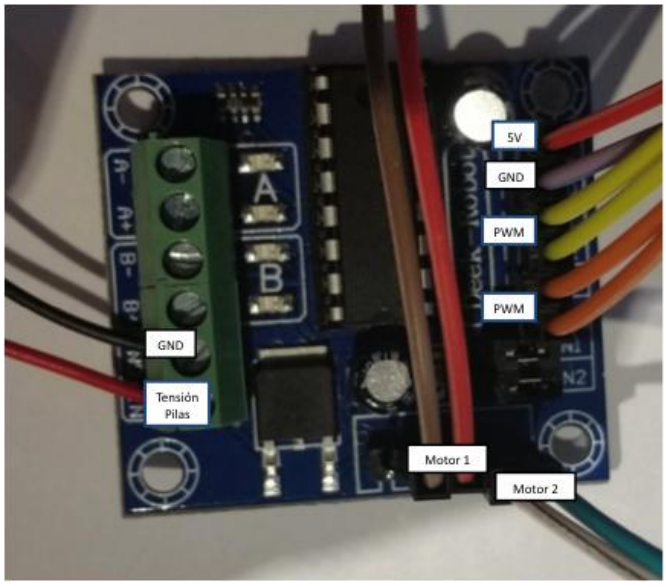

Fig. 8.6. Conexiones físicas del amplificador L293D

Llegados a este punto de la implementación, el coche ya puede moverse mediante conducción manual. Se ha de tener en cuenta que requiere del programa y la aplicación creados para que el microcontrolador actúe en consecuencia a las órdenes dadas.

Todos estos componentes se han implementado sobre un chasis que se ha montado previamente y que está habilitado con soportes para los motores, las ruedas y la cajetilla de las pilas *(Fig.8.7).* En el chasis también se han sujetado dos *protoboard* donde se encuentran los circuitos de conexiones.

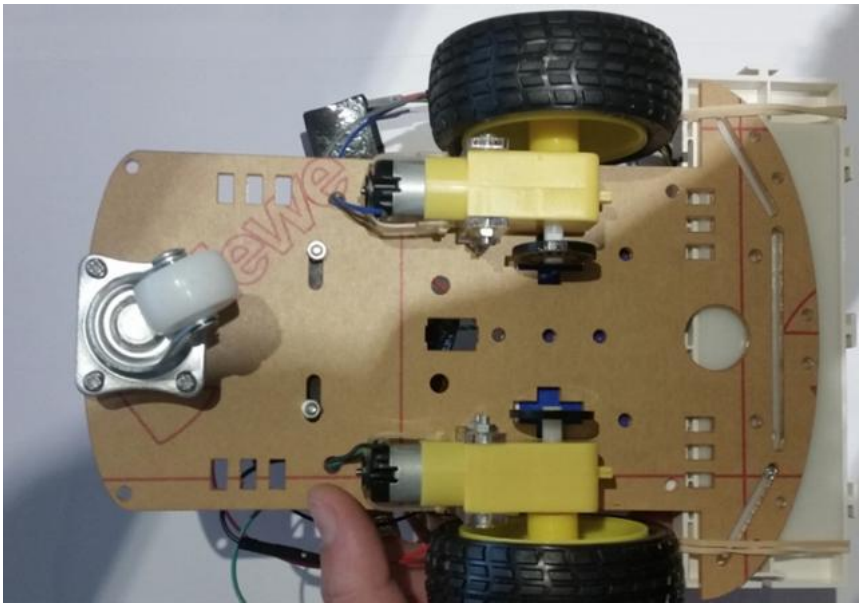

Fig. 8.7. Chasis con las ruedas y los motores

8.2. Segunda parte. Diseño definitivo.

Para esta segunda parte de la implementación se ha requerido de una serie de material del laboratorio para poder estudiar el comportamiento del sensor de color que, posteriormente, no se ha utilizado en el coche como por ejemplo una pantalla LCD.

8.2.1. Conexión del sensor de color

Primeramente se realizaron las conexiones del sensor con el microcontrolador. El sensor cuenta con 8 pines *(Fig.8.8.).* Primero se conectó el dispositivo a la alimentación de 5V y a GND. Posteriormente se habilitaron 1 pin del microcontrolador como entrada para recibir la información del sensor y 5 pines como salidas para activar controles del sensor.

Dos de los 5 pines configurados de salida van conectados a las señales S0 y S1 del sensor, por los que se escala la frecuencia de salida. En este caso, se ha escogido, mediante el programa creado, la opción del 2% de la frecuencia máxima posible (600kHz) por ser suficiente para los requerimientos del proyecto. Dos de los otros pines de salida van conectados a las entradas S2 y S3 del sensor por las que se controla el filtro RGB. El último pin requerido es el que habilita el sensor, el Enable (OE), que solo estará activado cuando se entra en conducción automática y permitirá que el sensor vaya realizando las diferentes medidas de frecuencia de los colores. Finalmente, el último pin es el de la salida de la onda cuadrada de la frecuencia (OUT) que actúa como entrada del PIC.

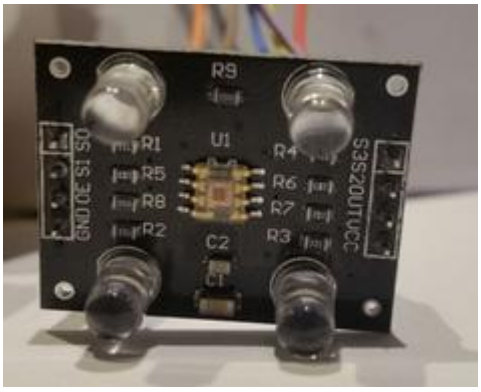

Fig. 8.8. Sensor TCS3210

8.2.2. Pruebas del sensor de color

Para poder extraer resultados de las pruebas se montó un primer circuito, conectando una pantalla LCD, que permitía leer el valor de la frecuencia con la que se detectaba el color verde, rojo y azul. Esta frecuencia se obtenía como respuesta a un objeto de color que se ponía enfrente del sensor. De esta forma se pudo estudiar el comportamiento del sensor delante de diferentes colores y, así, plantear diferentes formas para programar esta parte del proyecto.

En las siguientes imágenes se puede ver el estudio de la frecuencia del sensor para el caso del filtro de color rojo seleccionado. Concretamente, en la imagen que aparece a continuación *(Fig. 8.9)* se observa la onda cuadrada de la frecuencia del sensor cuando se le pone un color verde enfrente. La onda que muestra la pantalla del osciloscopio corresponde a la frecuencia que se puede leer en la pantalla LCD de la *Fig. 8.9* y al objeto de color verde que también se observa. La frecuencia de respuesta al color rojo que detecta el sensor se puede ver que es un valor relativamente bajo, de 3732 Hz, y por eso la onda oscila a velocidad baja.

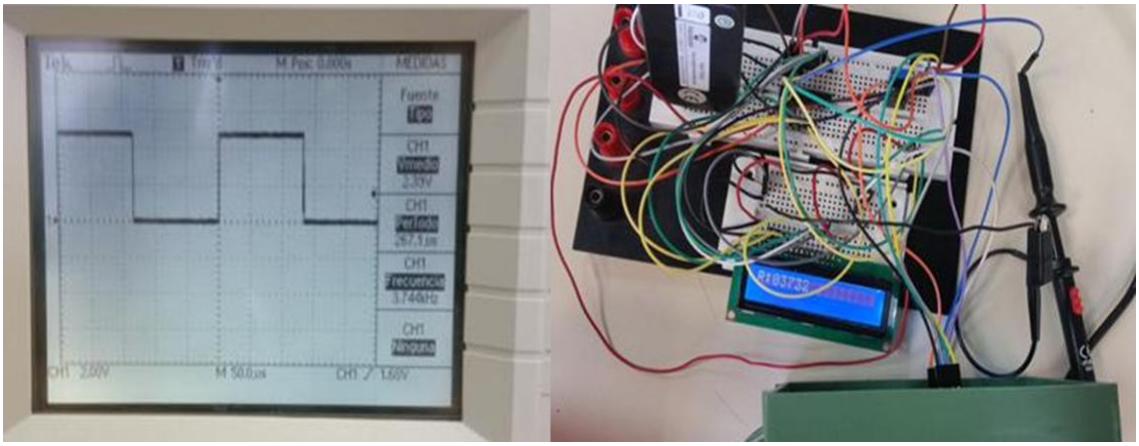

Fig. 8.9. Circuito y resultados para la detección de un color verde

En el siguiente conjunto de imágenes *(Fig. 8.10)* se puede ver la onda que genera el sensor para la frecuencia del rojo al poner delante de éste un color rojizo. Por ello se ve que la onda oscila a una mayor velocidad y el valor que se lee por pantalla es más elevado de 12104 Hz.

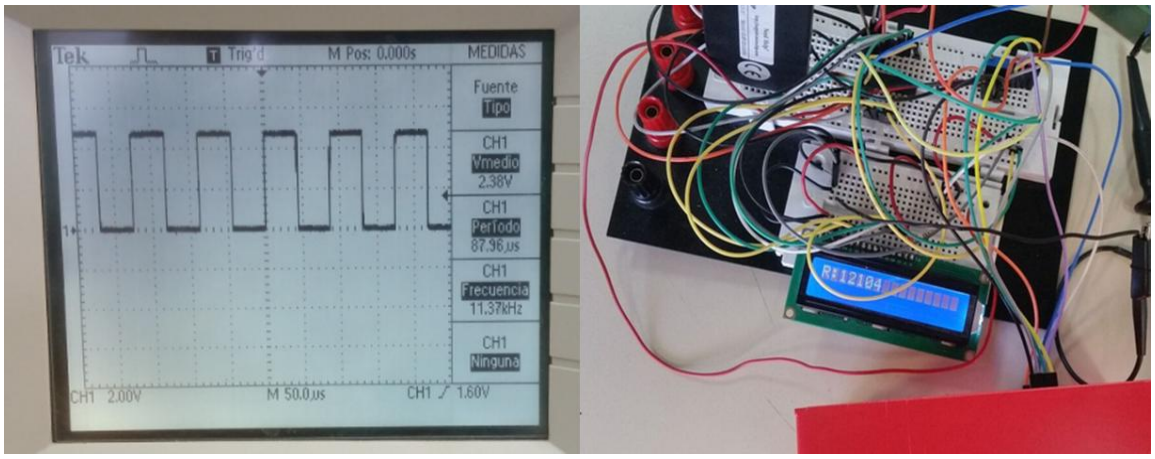

Fig. 8.10. Circuito y resultados para la detección de un color rojo

Al realizar este estudio se pudo ver que las lecturas de un mismo color se podían ver muy afectada por la iluminación ambiente debido a que el sensor capta toda su luz incidente y los colores que lo rodean. Por este motivo los valores podían oscilar demasiado como para poder tomar un rango de frecuencias concretas en un período de tiempo necesario para el procesamiento.

8.2.3. Pruebas RGB

Para poder realizar las lecturas con mayor precisión se creó una caja negra para envolver el sensor y que no tuviese tanta influencia del entorno. Este recurso ha sido posible ya que el propio sensor funciona constantemente con 4 LED's blancos encendidos.

Como el sensor detecta los colores RGB se ha creado para el proyecto unas tarjetas con estos colores. Se han ido realizando pruebas con distintas tonalidades de rojo, verde y azul hasta encontrar los que detectaba mejor el sensor y, así, reducir posibles errores de lectura. Las tarjetas de colores se insertan en una ranura de la caja negra para facilitar al sensor la detección del color y no impedir al coche el libre movimiento.

Fig. 8.11. Colores RGB

Pasando a la parte experimental, se han realizado diferentes pruebas con el osciloscopio y la pantalla LCD. En las siguientes imágenes *(Fig. 8.12, 8.13 y 8.14)* se puede apreciar el valor numérico de la frecuencia al ir poniendo los colores RGB en frente del sensor.

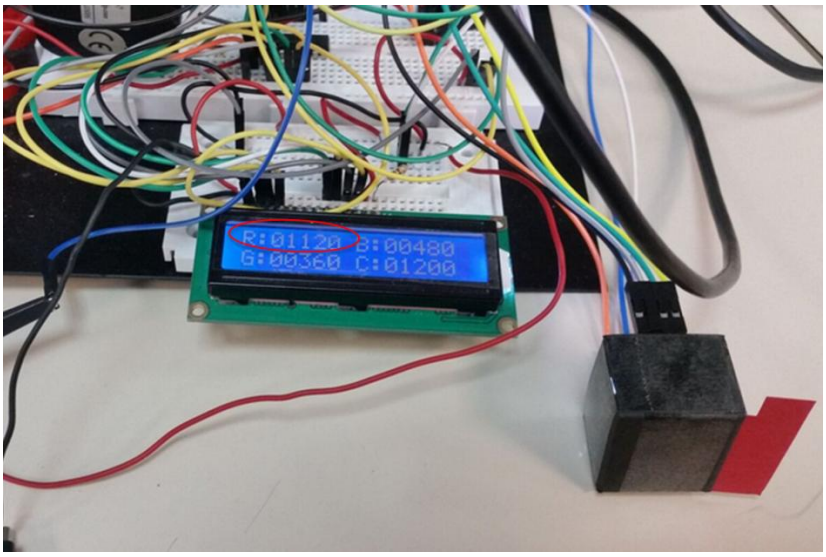

Fig. 8.12. Circuito y valor de la frecuencia para la detección del color rojo

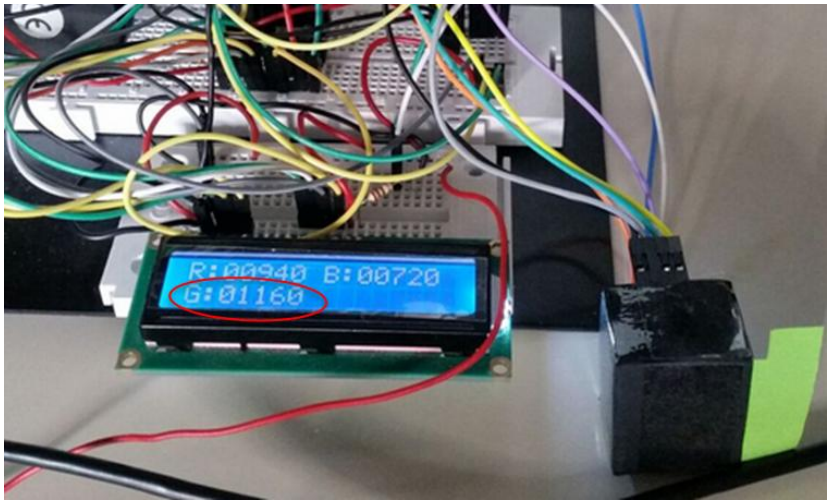

Fig. 8.13. Circuito y valor de la frecuencia para la detección del color verde

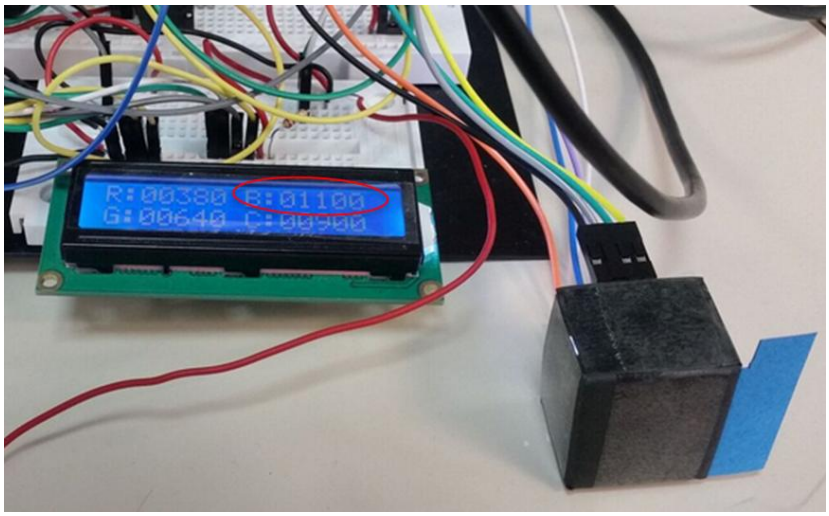

Fig. 8.14. Circuito y valor de la frecuencia para la detección del color azul

Al realizar estas pruebas experimentales se pudieron comprobar diferentes evidencias.

Primeramente que el sensor con la caja negra no otorgaba un valor mayor a 300 en ninguno de los tres colores RGB. Este dato ha resultado muy útil a la hora de la programación para poder restringir las condiciones.

Nótese que entre las Figuras 8.9-10 y las 8.12-14 hay un cambio de escala en los valores de frecuencia ya que se cambió la ventana de tiempo de medida. En las primeras imágenes se calculó la frecuencia contando el número de ciclos obtenidos durante 1s para hacer coincidir el resultado con el que mostraba el osciloscopio y, en las últimas imágenes *(Fig. 8.12-14),* se redujo la ventana de tiempo una cuarta parte.

Se comprobó también que el color blanco, que en la pantalla LCD es el valor que acompaña la letra C *(Clear),* siempre es el valor más elevado ya que, como se ha explicado previamente, el color blanco es la suma de los tres colores RGB. En este proyecto no se ha tenido en cuenta este color.

Como se puede ver en las imágenes previas, el color introducido en la caja negra siempre tiene un valor mayor a los otros dos colores RGB pero en el caso del verde, éste tiene un valor superior pero no tan evidente como cuando se introduce el rojo o el azul. Esto ha conllevado al ajuste de tiempos de medida y a una serie de funciones iterativas para reducir al máximo el error mediante programación.

Una vez se obtuvieron unos valores para poder programar, se tuvo que crear un circuito de detección de los colores para comprobar que el programa funcionase correctamente antes de implementarlo en el vehículo teledirigido. De esta forma se optó por 3 LED's, cada uno de un color RGB y, entonces, si el sensor detectaba el color rojo, el LED rojo recibía una tensión de 5V y quedaba encendido *(Fig.8.15)*. De forma similar ocurría con los colores verde y azul.

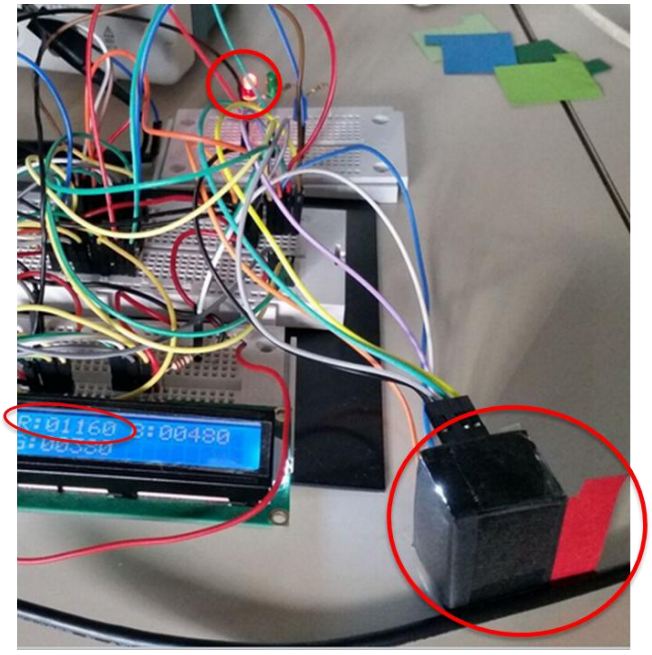

Fig. 8.15. Circuito de detección del rojo con LED y valor de la frecuencia

Una vez el programa respondía de la forma requerida se prosiguió a implementar el sensor en el coche. Se utilizaron las mismas conexiones descritas con anterioridad y se mantuvo el circuito de los LED's que indica el color que está detectando el sensor en cada momento a modo informativo. Posteriormente, mediante programación, se ha asociado a cada color un movimiento.

8.2.4. Ocupación final pines PIC 16F690

En la Fig. 8.16 se muestra la asignación final de los pines del PIC y su relación con los módulos externos.

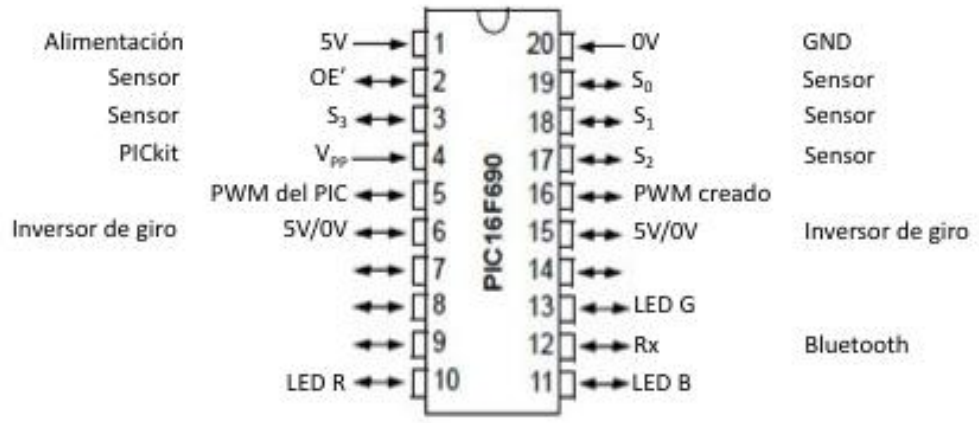

Fig. 8.16. Diagrama de pines del PIC16F690 ocupados

9. Programación del microcontrolador

En este apartado se explicará el funcionamiento del programa que está compilado y grabado en el microcontrolador. Se describirán las diferentes funciones que se han tenido que ir creando para que el coche pueda realizar todas las acciones requeridas.

En el Anexo se puede consultar el código de todos los archivos descritos en este apartado.

9.1. Introducción

Antes de empezar a explicar el código que se ha escrito, es necesario mencionar algunos conceptos previos. Se ha escrito una función principal que controla todos los movimientos del coche pero que necesita solicitar un conjunto de funciones auxiliares que realizan cálculos para poder ejecutar todas las acciones requeridas. El lenguaje utilizado, como ya se ha mencionado previamente, es el lenguaje de alto nivel C.

Además de programar las funciones del proyecto, se deben incluir los archivos de librería .h utilizados que, en este caso, han sido 2: el *timer.h* y la *uart.h*. Además se debe configurar el PIC y definir todas las variables y funciones que se van a utilizar a lo largo del programa.

El programa que se ha creado es el que debe ejecutar el PIC16F690 y tiene como función que el microcontrolador se comunique con el resto de componentes para lograr los movimientos que se le indican al coche.

9.2. Función principal

En esta función es donde se decide las instrucciones que se envían a los motores. Esta decisión se toma en base a las instrucciones enviadas desde el *Smartphone*. Cada instrucción (marcha adelante, atrás, giro y stop) viene codificada por un valor numérico (variable DADA) entre 0 y 255 que es el transmitido vía *bluetooth*. En la *Tabla 9.3* se detalla esta relación.

En la función principal también se define la configuración de las entradas y salidas de los pines del microcontrolador así como su carácter digital. Al inicio de esta función además se configuran los Timers 0, 1, 2 y también la USART. Por último, como paso previo, se han de activar las interrupciones periféricas y globales e inicializar las variables que se usarán en los valores que se requiere.

Se divide en dos partes diferenciadas, por un lado las acciones a realizar para cualquier valor recibido que sea un valor entre 0 y 255 excluyendo el 2 y, por otro lado, las acciones a realizar cuando se recibe el número 2 que es el valor asignado para entrar en modo de conducción automática.

9.2.1. Conducción manual (Si DADA no es 2)

El valor de la variable DADA se obtiene al leer la UART y contiene información relativa al tipo de movimiento y velocidad del vehículo que harán que en la función principal se activen ambos PWM asignando a cada módulo su valor apropiado. Este valor se calculará mediante la operación matemática correspondiente. Si el valor de velocidad (SPEED) es superior a 150 se utilizarán las operaciones *Ec. 9.1* y *9.2* y si la velocidad es inferior, las operaciones *Ec. 9.3 y 9.4.*

CCPR1L = SPEED + 2 * VAR_RIGHT;

Ec. 9.1

PWM_homemade(SPEED + 2 * VAR_LEFT);

Ec. 9.2

CCPR1L = SPEED + 4 * VAR_RIGHT;

Ec. 9.3

PWM_homemade(SPEED + 4 * VAR_LEFT);

Ec. 9.4

La variable CCPR1L será el valor que se le asigne al PWM que genera el propio microcontrolador y el *PWM_homemade* será solicitado introduciendo el valor que proporciona la operación que tiene entre paréntesis.

9.2.2. Conducción automática (Si DADA es 2)

El valor 2 se ha escogido para el caso de conducción automática y esto requiere, para movilizar los motores, detectar el color de la tarjeta que se introduce en la caja negra. Por lo tanto, primeramente se ha de habilitar el sensor y esta tarea se consigue poniendo el ENABLE a 0 ya que tiene entrada por lógica negada.

A continuación, en el programa principal, se configura las entradas S2 y S3 con el filtro rojo y se solicita la función de *''mesura frequencia''* de donde se consigue el valor de la frecuencia del rojo obtenida del color que está enfrente del sensor. A continuación se realizan los mismos pasos para los otros dos colores, el verde y el azul.

Una vez se tienen estos 3 valores de frecuencia, se crea la variable NEGRO que es la suma de los tres valores. Se puede realizar los pasos que siguen a continuación ya que el sensor está rodeado por una caja negra y el color negro devuelve frecuencias muy bajas para cualquiera de los 3 colores RGB. De esta manera si la variable NEGRO no supera el valor umbral de 60, el programa asimila que no se ha introducido ninguna tarjeta y vuelve a hacer el barrido de las frecuencias sin realizar ningún movimiento de motores. La inactivación de los motores se registra en la variable ACCIÓN mediante el valor 0.

Si por el contrario, sí que se supera el valor umbral de 60 y la variable ACCIÓN es 0, para lograr reducir al máximo el error de detección, se calcula la variable COLOR_DETECTADO (es el resultado de solicitar la función *máximo* explicada en el *apartado 9.3.2*) y 500 milisegundos después se vuelve a calcular y, solo en caso de que coincidan ambos valores, se calcula una tercera vez. Si este tercer valor no coincide con el anterior, se vuelve a hacer el barrido de frecuencias. La indicación del color detectado se ha implementado en base a los números 1, 2, 3 indicando rojo, verde y azul, respectivamente.

Así, pues, en caso de coincidan ambos colores detectados consecutivamente, se compara el COLOR_DETECTADO con 1, 2 y 3 y dependiendo del valor con el que coincida, se enciende el LED de dicho color y se otorga a los PWM unos valores concretos. Una vez realizadas todas las acciones, la variable ACCIÓN se pone a 1 y a los PWM se les otorga un valor de 0 para que el coche se pare. En la siguiente *Tabla 9.1* se recogen las acciones en función del color detectado.

Resumiendo, la variable ACCIÓN a 0 significa que aún no se ha movido el coche con la tarjeta que se ha introducido, cuando pasa a valor 1 el coche ya ha realizado el valor requerido.

COLOR_DETECTADO	Acciones
1	LED_R = 1; LED_G = 0; LED_B = 0; CCPR1L = SPEED; PWM_homemade(SPEED); CCPR1L = 0 ; PWM_homemade(0); ACCION = 1;
2	LED_R = 0; LED_G = 1; LED_B = 0; CCPR1L = SPEED + 100 ; PWM_homemade(SPEED); __delay_ms (2000); CCPR1L = 0 ; PWM_homemade(0); ACCION = 1;
3	LED_R = 0; LED_G = 0; LED_B = 1; CCPR1L = SPEED; PWM_homemade(SPEED + 100); CCPR1L = 0 ; PWM_homemade(0); ACCION = 1;

Tabla 9.1. Acciones en función del color detectado

Así, si el color detectado es el rojo el coche girará a la izquierda, si el color detectado es el azul girará a la derecha y si se trata del verde avanzará hacia delante.

9.3. Funciones específicas para el sensor

Para poder adecuar el sensor a los requerimientos del sistema se han tenido que crear una serie de funciones que se explican a continuación.

9.3.1. Función "Mesura Frequencia''

El sensor utilizado detecta los colores según sus componentes RGB. Cuando se quiere saber la composición de un color, mediante el sensor, se ha de hacer un barrido de la frecuencia correspondiente a cada uno de los tres colores RGB por separado y, una vez obtenidos los valores, en función de éstos, se puede saber el color debido al dominio del rojo, verde o azul en éste. Cuanto más alta es la frecuencia de uno de los tres colores, significa que ese color se encuentra en mayor proporción.

La función *"mesura frequencia"* tiene como objetivo devolver el valor de la frecuencia del color rojo, verde o azul, dependiendo cuál se esté consultando, y aprovecha el Timer 1 del propio microcontrolador para hacerlo. Concretamente, se conecta la señal, de la que se desea conocer la frecuencia, a la entrada de reloj del Timer1. La lectura del valor del Timer1 es una indicación de los ciclos transcurridos en un intervalo de tiempo.

La tarea de esta función consiste en inicializar el Timer1 a cero e ir consultando su valor cada 50 ms durante un período de 200 ms e ir guardando los diferentes valores. De esta manera, los valores corresponderán a la frecuencia de un mismo color medida en diferentes intervalos de tiempo pero de la misma duración con la finalidad de realizar un posterior promedio para evitar distorsiones por transitorios indeseados. A continuación se detiene el Timer 1 y se reestablece a 0 para poder realizar otras medidas en un futuro. También hace la media de los 4 valores obtenidos para así reducir al máximo el error de lectura por si en uno de los periodos de 50 ms el sensor hubiese leído un valor erróneo. De esta forma esta función retorna el valor de la frecuencia con la que se está detectando en ese momento el color RGB requerido.

9.3.2. Función Máximo

El sensor, para poder detectar el color que tiene enfrente, ha de ir haciendo barridos de los tres colores RGB, es decir necesita solicitar la función "*mesura frequencia"* para cada uno de los tres colores y, en base a los resultados obtenidos, quedarse con el máximo valor de los tres. El color con el número más elevado de la frecuencia es el que se considera detectado.

Esta función compara las frecuencias de los tres colores, unas con otras, y retorna un número distinto en función del color del que se obtiene el valor más alto. Los valores que retorna en función del color, coincidiendo con el código establecido para la variable

COLOR_DETECTADO. En la *Tabla 9.2.* ilustra este código.

Número que retorna	Color detectado
1	Rojo
2	Verde
3	Azul

Tabla 9.2. Equivalencia de los números devueltos con los colores detectados

9.4. Función PWM homemade

Como se ha explicado en el apartado de implementación (apartado 8.1.2) ha sido necesaria la creación de un PWM propio para poder controlar dos motores por separado (para la realización de giros) ya que el microcontrolador utilizado solo tiene un módulo PWM.

Básicamente esta función retorna una onda cuadrada de las mismas características que la que genera el PIC. Esto se ha conseguido mediante el uso principalmente de la función *delay*, que utiliza el Timer 0.

El valor que se introduce en la función, que ha de ser un valor comprendido entre el 0 y el 255, será el número de microsegundos que se quiere tener la onda a su nivel alto. A mayor tiempo, mayor velocidad tendrán los motores. De esta forma, mientras el valor de los microsegundos sea mayor que 0, solicitará un delay de 8 microsegundos e irá restando los microsegundos de uno en uno hasta que no cumpla la condición previamente mencionada. Así se obtiene una función iterativa que mantiene la onda en su nivel alto tantos microsegundos como valor se ha introducido en la función.

El valor de 8 microsegundos se ha encontrado de forma experimental. Se han ido probando valores hasta que finalmente este valor era el que retornaba una onda lo más parecida posible a la que general el PWM del PIC. La utilización de lenguaje ensamblador hubiese dado una información más precisa respecto al tiempo de ejecución del programa y se hubiese podido evitar el ajuste experimental.

9.5. Servicio de Interrupción

En el programa creado se recurre constantemente a esta función ya que cada vez que el usuario envía una nueva orden, vía *bluetooth*, es necesario interrumpir cualquier acción que se esté ejecutando y proceder a analizar la nueva orden que se está recibiendo.

Esta función/servicio toma como dato el valor que lee la UART, es decir el número que envía el usuario, y que corresponde a una acción que se definirá en la función principal.

El *hardware* implicado en la interrupción de la UART activa la bandera de interrupción cuando se recibe un dato por el puerto Rx del PIC y pasa a realizar un conjunto de acciones programadas. Una vez obtenido los valores necesarios se desactiva la bandera de interrupción de recepción en el buffer de entrada de la UART y se retoma el programa principal donde se había dejado antes de la interrupción.

Esta función toma como DADA el valor recibido en la UART y, dependiendo de éste, retorna unos valores de las variables SPEED, VAR_LEFT y VAR_RIGHT que se utilizarán en la función principal para manejar de la forma que se quiera los motores.

En el caso del número 199 también modifica correspondientemente el valor de los pines reservados a tal efecto, consiguiendo hacer girar los motores en sentido contrario y creando así el movimiento hacia atrás.

A continuación se detallará en la *Tabla. 9.3* las variables y sus valores que se extraen de esta función para cada dato recibido.

Número = DADA	Valores asignados	Acción
0	VAR_RIGHT = 0; VAR_LEFT = 0; SPEED = DADA;	STOP: el coche se para o sale de conducción automática
199	BACK_RIGHT = ON; BACK_LEFT = ON; SPEED = 70; VAR_RIGHT = 0; VAR_LEFT = 0;	BACK: marcha atrás a velocidad constante predeterminada y sin posibilidad de giro

50-149	SPEED = 50 + DADA;	GO: marcha adelante. A mayor valor, más rapidez
200-223	VAR_RIGHT = 224 - DADA;	RIGHT: giro a la derecha. A mayor número, giro menos pronunciado
224-249	VAR_LEFT = DADA - 224;	LEFT: giro a la izquierda. A mayor número, giro menos pronunciado
2	SPEED = DADA + 123;	AUTOMATIC: el coche entra en conducción automática

Tabla 9.3. Variables, valores y acciones en función del número recibido

10. Presupuesto

A continuación se desglosarán los costes de este proyecto dividiéndolos en dos categorías diferentes: costes de recursos materiales y costes de recursos humanos donde se detalla únicamente las horas de trabajo de la autora del proyecto.

Como se podrá ver en el *apartado 10.3* hay una tercera tabla (*Tabla 9.3)* de costes que se ha denominado como costes varios donde se ha incluido el coste de la tutora y la amortización del ordenador con el que se ha trabajado durante todo el semestre para programar y realizar la memoria. Finalmente se dará el valor del coste total *(Tabla 9.4).*

10.1. Coste recursos materiales

En la *Tabla 10.1* se recogen los costes del material indicando el precio unitario y el total de cada componente así como el número consumido de cada uno.

Material	Cantidad	Precio Unitario	Coste Total [€]
Chasis + ruedas + motores	1	23,28	23,28
Protoboard	1	4,1	4,1
PIC16F690	1	2,81	2,81
L293D motor shield	1	3,5	3,5
TCS3210	1	4,5	4,5
PICkit2	1	35	35
Bluetooth BC417	1	5,95	5,95
Pack de Cables	1	1,74	1,74
LED rojo	1	0,35	0,35

LED verde	1	0,35	0,35
LED azul	1	0,39	0,39
Resistencia	3	0,04	0,12
Cartulina	4	0,25	1
Pack 4 Pilas	1	3,6	3,6
COSTE [€]			**86,69**

Tabla 10.1. Coste de los recursos materiales

10.2. Coste recursos humanos

En la tabla 10.2 se recoge el coste del trabajo de la autora de este proyecto separándolo por horas dedicadas a cada parte del trabajo realizado.

Trabajo	**horas**	**€/hora**	**Coste [€]**
Estudio	40	15	600
Montaje	20	15	300
Programación	150	15	2250
Pruebas	70	15	1050
Memoria	80	15	1200
COSTE [€]			**5400**

Tabla 10.2. Costes recursos humanos. Coste del trabajo de la autora del proyecto.

10.3. Costes varios

En este apartado se detalla el coste de la tutora que ha dirigido este proyecto y el coste del ordenador con el que se ha realizado el trabajo *(Tabla 10.3).*

Tipo			**Coste [€]**
Dedicación tutora	20 horas	50 €/hora	1000€

Tipo	**Valor**	**Amortización**	**Uso**	**Coste [€]**
Ordenador	1000€	A 5 años	0,5 años	100€
COSTE [€]				**1100€**

Tabla 10.3. Coste trabajo tutora y ordenador

10.4. Costes totales del proyecto

Finalmente en este apartado se muestran los costes totales del proyecto, el precio total sin IVA, el coste del IVA y por último el precio total con el IVA incluido. *(Tabla 10.4).*

TOTAL sin IVA	6586,69 €
IVA (21%)	1383,205 €
TOTAL	**7969,895€**

Tabla 10.4. Costes totales

11. Contaminación y medioambiente

Los aparatos eléctricos están diseñados para durar un tiempo determinado y cuando se rompen se tiran a la basura sin pensar en arreglarlos o incluso en recuperar y aprovechar sus componentes para construir otros nuevos. Por lo tanto la mayoría de aparatos eléctricos o electrónicos no deberían depositarse en los contenedores de basura normal y cuando se acaba su vida útil deberían ser tratados de manera especial ya que muchos de ellos son tóxicos y peligrosos para el medio ambiente.

En la actualidad los aparatos eléctricos son cada vez más abundantes. De hecho vivimos rodeados de estos aparatos tales como los electrodomésticos, los ordenadores, los móviles o simplemente la electricidad. Todos estos objetos son cosas tan habituales hoy en día y a las que no les damos importancia pero que suponen un gran problema para el medio ambiente. La mayoría de las personas no se plantea qué pasa con todos estos aparatos cuando dejan de funcionar y, desafortunadamente, cada año aumenta la cantidad de residuos de piezas eléctricas y electrónicas en el mundo. Estos aparatos tienen sustancias que son perjudiciales para el medio ambiente y para la salud de las personas. Por ejemplo con el fósforo que contiene una televisión se puede contaminar 80.000 litros de agua.

Lo ideal sería que los ciudadanos intentaran reducir el uso de estos aparatos eléctricos, reutilizarlos si es posible y, sino, pues reciclar sus piezas. De hecho estos componentes, una vez llegan a las plantas de reciclaje, se les retiran los elementos contaminantes como las pilas y se les separa el resto de componentes tipo plástico, aluminio, cobre… y se procesan para de esta forma poder fabricar nuevos productos. Este proceso de reciclaje de los dispositivos electrónicos es relativamente sencillo en comparación con otras industrias. Por este motivo es muy importante tener consciencia social y cuando un objeto ya no tenga utilidad intentar reciclar el máximo de piezas posibles.

El vehículo terrestre teledirigido que se ha construido para este trabajo de final de grado tiene muchos componentes electrónicos. Por lo tanto es importante saber qué pasará con todos estos componentes cuando el vehículo deje de tener utilidad. Se sabe que cualquier objeto de consumo genera residuos y el vehículo que se ha construido lleva muchas piezas eléctricas que sino se reciclan van a causar residuos que podrían tener impacto medio ambiental, directo o indirecto. Por lo tanto la despreocupación de tirar cualquier objeto a la basura sin intentar recuperar las piezas útiles es un gran daño para nuestro ecosistema. Incluso cuando el aparato es tan pequeño como el vehículo que nos ocupa.

El volumen de la chatarra electrónica crece cada año de forma exponencial. De hecho ha sido en esta última década cuando muchos ciudadanos se han dado cuenta del gravísimo problema que supone tantas toneladas de basura electrónica. Cuando estos aparatos se

desechan se convierten en unos residuos muy contaminantes y muchos de ellos contienen sustancias como el bromo, cadmio, fósforo o mercurios que son muy dañinos para la salud y para el medio ambiente. La legislación ambiental, afortunadamente, cada vez es más exigente y las personas estamos cada vez más concienciadas. Existen puntos de recogida de estos materiales y de la misma forma que se puede construir un sistema eléctrico para que un vehículo pueda desplazarse también se ha de poder desmontar cada una de sus piezas y reciclarse cuando el objeto deje de tener la utilidad que se pretendía darle.

Conclusiones

Este trabajo de final de Grado ha resultado una experiencia muy enriquecedora. Partiendo de la base que la elección del proyecto ha sido construir un vehículo terrestre electrónico y programarlo para que realice cinco movimientos básicos demandados por dos tipos diferentes de conducción.

Se puede afirmar que se han logrado todos los objetivos propuestos al inicio. Diseñar el coche ha sido la parte más fácil ya que de alguna manera los componentes estaban claros desde el inicio. La dificultad ha surgido al programar el microcontrolador ya que era necesario adaptar el funcionamiento de todos los componentes, uno a uno, para que en conjunto realizasen las acciones requeridas sobre los motores.

Se ha conseguido controlar el movimiento del vehículo mediante las órdenes que decide dar el usuario en función de una serie de opciones predeterminadas desde una aplicación de teléfono Android y, también, las órdenes de los colores según los códigos establecidos. Todo esto ha supuesto un gran aprendizaje en cuanto a conocimientos de electrónica e informática.

Es importante confirmar el auge de las nuevas tecnologías que siempre permiten avanzar. En el caso de la autora de este trabajo este coche supone la base para, en un futuro, poder realizar experimentos tecnológicos con otro tipo de vehículos consiguiendo circuitos electrónicos más sofisticados en los que se puedan demandar acciones más complejas.

Para acabar, aunque el resultado del trabajo ha sido satisfactorio, cabe mencionar que al vehículo se le podrían hacer un conjunto de mejoras de cara al futuro. Como por ejemplo mejorar la detección de colores y crear una secuencia de movimientos más fluidos para cada color. Por otro lado también se podría crear una aplicación más sofisticada para manejar el coche.

Agradecimientos

Estoy muy agradecida mi profesora y tutora Rosa Rodríguez Montañés, del departamento de electrónica de la ETSEIB, que me ha brindado todo su apoyo dedicándome su tiempo para poder llevar a cabo este trabajo.

Todo mi agradecimiento para la ETSEIB y sobretodo al departamento de electrónica que ha puesto a mi disposición el laboratorio y me ha proporcionado todo el material necesario.

También mi agradecimiento a mi familia por su paciencia y comprensión durante estos cuatro años de estudio intensivo y en especial a mi madre por su gran apoyo.

Por último agradecer a mi hermano el haberme transmitido su gran pasión por las nuevas tecnologías y a mi hermana por haberme animado a estudiar una ingeniería.

Bibliografía

En este apartado se incluye la información de las fuentes que se han ido citando a lo largo de este documento.

Referencias bibliográficas

[1] Microchip Technology Inc. *PIC16F631/677/685/687/689/690 Data Sheet*. 2006.

[2] TAOS Inc. *TCS3200, TCS3210 Programmable color Light-to-frequency converter.* July 2009

[3] Microchip Technology Inc. *MPLAB XC8 C COMPILER User's Guide*. 2012.

[4] Texas Instrument. *L293, L293D. Data Sheet*. January 2016.

[5] CSR. *BC417 BlueCore4-Ext™. Data Sheet.* 2005.

[6] Microchip Technology Inc. *PICkitTM 2 Programmer/Debugger User's Guide.* 2008.