Plataforma robótica experimental

Juan Barea Lozano

Grado de Tecnologías de Telecomunicación

Arduino

Antoni Morell Pérez

Pere Tuset Peiró

08/01/2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Plataforma robótica experimental*
Nombre del autor:	*Barea lozano Juan*
Nombre del consultor/a:	*Antoni Morell Pérez*
Nombre del PRA:	*Pere Tuset Peiró*
Fecha de entrega (mm/aaaa):	*01/2019*
Titulación::	Grado de Tecnologías de Telecomunicación
Área del Trabajo Final:	*Arduino*
Idioma del trabajo:	*Castellano*
Palabras clave	*Robot, Sharp, omnidireccional*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objetivo del presente proyecto es el desarrollo de una plataforma robótica móvil que pueda ser dirigida a distancia mediante una aplicación para Android, usando un Smartphone o tableta. Se dotará al robot, además, de otras funcionalidades como son un sistema para no colisionar con obstáculos, un sistema de control de la alimentación y otras aplicaciones secundarias.

El proyecto engloba todo lo relacionado con el hardware necesario para diseñar y construir un prototipo donde poner a prueba todo el software que se ha debido desarrollar para controlar el robot. Para conseguirlo se ha adaptado un chasis con tres motores donde se han montado todos los sensores, actuadores y demás elementos necesarios para construir el prototipo. Se han utilizado dos tarjetas de micro-controlador, una como unidad principal de proceso y otra para hacer posible el control a distancia mediante su módulo *wifi.* Por su parte, el sistema detector de obstáculos se ha implementado con varios sensores de infrarrojos y un sensor láser para adaptar la velocidad del vehículo automáticamente.

La construcción del robot se ha realizado estructurándolo en distintos módulos, implementando Hardware y software simultáneamente e incorporándolos al conjunto a medida que se iban finalizando las tareas.

Al tratarse de un prototipo se puede emplear en distintas funciones, ya sea como plataforma base para construir un robot de limpieza o para desarrollar funciones de vigilancia y, además, para servir como medio de aprendizaje e iniciación a todos aquellos que tengan interés en el diseño y construcción de robots móviles.

Abstract (in English, 250 words or less):

The objective of the present project is the development of a mobile robotic platform that can be remotely addressed through an Android application, using a Smartphone or tablet. The robot will also be provided with other features such as a system to avoid collisions with obstacles, a power control system and other secondary applications.

The project includes everything related to the necessary hardware to design and build a prototype where to test all the software that has to be developed to control the robot. To achieve this, a chassis has been fitted with three motors where all the sensors, actuators and other elements necessary to build the prototype have been assembled. Two micro-controller cards have been used, one as the main processing unit and the other to make remote control possible through its Wi-Fi module. For its part, the obstacle detection system has been implemented with several infrared sensors and a laser sensor to adapt the speed of the vehicle automatically.

The construction of the robot has been done by structuring it in different modules, implementing hardware and software simultaneously and incorporating them to the set as the tasks were being finalized.

Being a prototype, it can be used in different functions, either as a base platform to build a cleaning robot or to develop surveillance functions and, also, to serve as a means of learning and initiation to all those who have an interest in the design and construction of mobile robots.

Índice

Índice de imágenes

Índice de tablas

1. Introducción

1.1 Motivación

La motivación para realizar este proyecto se fundamenta en la posibilidad de trabajar en distintas áreas de gran interés personal como son la electrónica, la robótica y las comunicaciones. En especial la posibilidad de poder programar micro controladores con los que controlar un pequeño robot.

El proyecto se basa en el desarrollo y aplicación de software libre, así como su implementación en micro controladores desarrollados para muy diversos propósitos (hardware libre). Se parte de una base donde existe una gran cantidad de librerías y proyectos realizados, así como una gran variedad de hardware que puede ser utilizada para poder realizar el proyecto.

En definitiva, en el presente proyecto se llevará a cabo la construcción de una plataforma robótica móvil realizada mediante la integración y validación de componentes, así como el control del dispositivo. El objetivo principal del proyecto es construir un prototipo de robot al que se pueda controlar a distancia (mediante *wifi*) a través de un Smartphone o Tableta y que tenga la capacidad de evadir obstáculos. Como función secundaria se podrá controlar el encendido de una cámara IP instalada en el dispositivo.

1.2 Objetivos

Los objetivos principales son los siguientes:

- Elegir el micro controlador adecuado para poder llevar a cabo todas las funciones con las que se dotaran al robot.
- Dotar de movilidad a la plataforma robótica.
- Realizar un sistema evasor de obstáculos capaz de detectar paredes, esquinas, cortinas, muebles, etc. habituales en las distintas estancias de cualquier casa, mediante el uso de sensores de infrarrojos tipo Sharp.
- Implementar un sistema detector de falta de suelo, o piso, mediante el uso de sensores infrarrojos del tipo seguidores de línea.
- Crear un sistema para poder regular la velocidad de desplazamiento del robot mediante un sensor laser cuando aquel se encuentre en el modo evasor de obstáculos (el robot tendrá dos estados básicos: modo de control manual y modo evasor de obstáculos).
- Crear un software específico para poder controlar el robot a distancia mediante un módulo *wifi*, que incorpore todos los programas creados en los puntos anteriores.
- Desarrollar una APP para el control a distancia del robot, a través de un dispositivo móvil (Smartphone o Tableta).
- Poder seleccionar dos modos de conducción mediante la APP: conducción manual mediante los botones de dirección y otro modo evasor de obstáculos. Además se podrá seleccionar la velocidad de desplazamiento mediante el correspondiente control deslizante, así como el encendido y apagado de dos relés, uno de ellos para activar/desactivar la alimentación de una cámara IP.
- Realizar un sistema de control del nivel de carga de la batería que avise cuando el nivel sea insuficiente.
- Diseño y construcción de un prototipo de robot donde comprobar el buen funcionamiento del software desarrollado para todas las tareas expuestas en los puntos anteriores.

Objetivos secundarios:

- Implementar un sistema de aviso acústico mediante un zumbador o buzzer.
- Instalar en el chasis del robot una pequeña pantalla donde poder visualizar cualquier dato como la velocidad de desplazamiento, nivel de carga de la batería o cualquier otro que se decida.
- Poder medir la distancia y velocidad de desplazamiento, así como el ángulo de giro de las ruedas.

El proyecto habrá finalizado cuando se hayan conseguido realizar con éxito los distintos objetivos establecidos en la relación anterior.

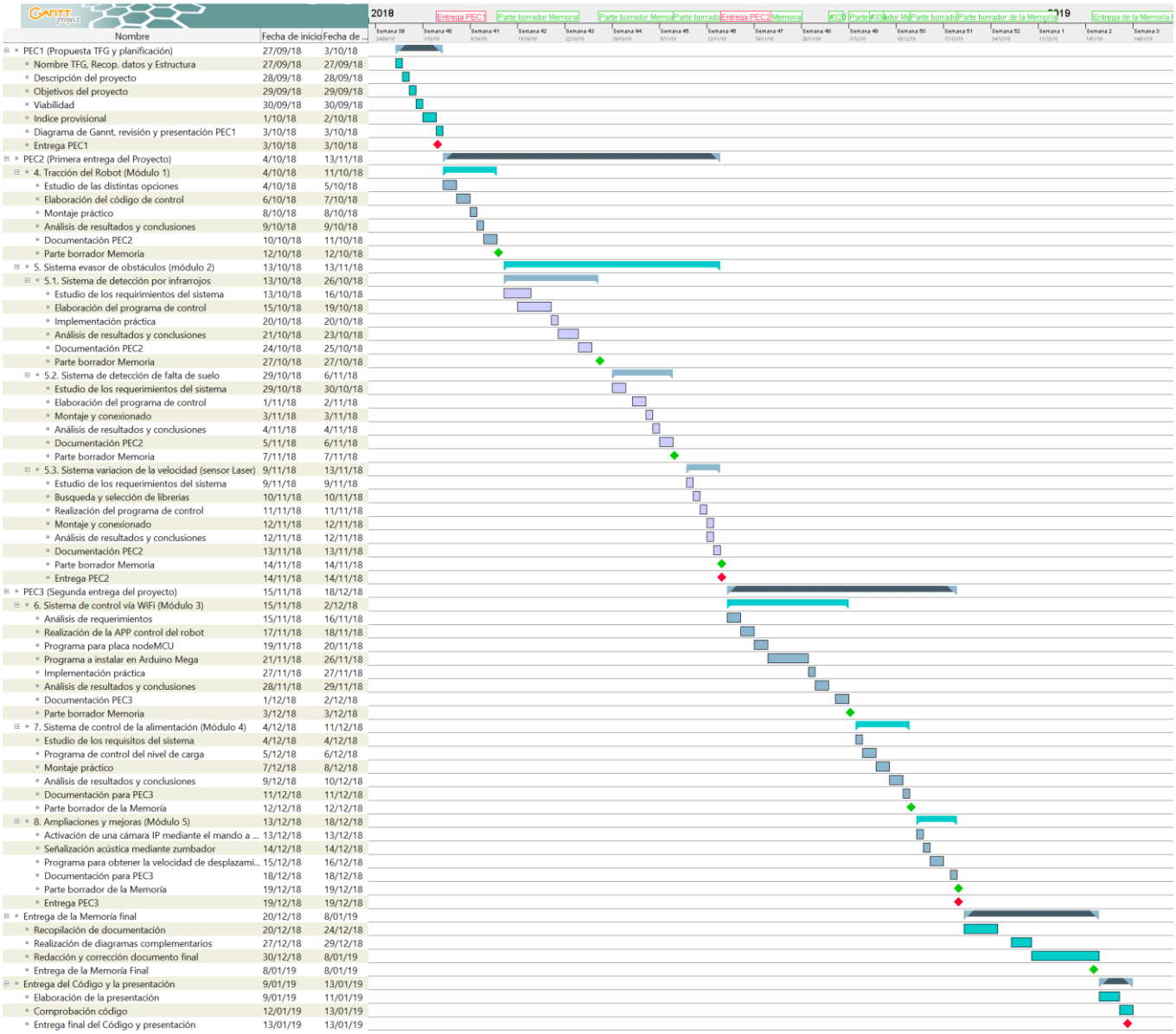
1.3 Enfoque y método seguido

Para realizar la plataforma robótica experimental se combinará una parte de hardware, representada principalmente por actuadores y sensores, y otra parte de software compuesta por los programas de control de las distintas funcionalidades y la aplicación (APP) para el control a distancia.

El método seguido para lograr los objetivos marcados se basa en realizar un proyecto de forma eminentemente práctica, construyendo un prototipo que simule de forma real las funciones típicas descritas en el apartado anterior. Para su construcción se ha decidido realizar el proyecto sin depender de ningún kit comercial para montar, ya que que se ha realizado el proyecto adquiriendo las distintas partes de Hardware necesarias a distintos proveedores.

1.4 Planificación del trabajo

Para la realización de las distintas tareas de este proyecto se ha decidido estructurarlo en módulos. Cada módulo engloba una serie de trabajos de similar naturaleza que han de realizarse en los plazos de tiempo establecidos. A continuación se muestra el diagrama de Gantt donde se fijan los plazos de comienzo y finalización de cada uno de los módulos y las tareas que los conforman.



1.4.1 Entregables a nivel de producto

Tarea	Inicio	Fin
4. Control de tracción (Módulo 1) - Estudio de las distintas opciones y recopilación de librerías - Elaboración del código - Pruebas y montaje práctico	04/10/18	12/10/18
5. Sistema evasor de obstáculos (Módulo 2)	13/10/18	13/11/18
5.1 Sistema de detección por infrarrojos: - Requerimientos, estudio de las distintas opciones, recopilación de librerías, elaboración del código y montaje práctico.	13/10/18	26/10/18

- Análisis de resultados y conclusiones.		
5.2 Sistema de detección de falta de suelo: - Requerimientos, estudio de las distintas opciones y recopilación de librerías, elaboración del código y montaje práctico. - Análisis de resultados y conclusiones.	29/10/18	06/11/18
5.3 Sistema de adaptación de la velocidad por sensor laser: - Requerimientos, estudio de las distintas opciones, recopilación de librerías, elaboración del código y montaje práctico. - Análisis de resultados y conclusiones.	09/11/18	13/11/18
6. Sistema control vía *wifi* (Módulo 3) - Requerimientos, estudio de las distintas opciones, recopilación de librerías, elaboración del código y montaje práctico. - Análisis de resultados y conclusiones.	15/11/18	02/12/18
7. Sistema alimentación del robot (Módulo 4) - Requerimientos, estudio de las distintas opciones, recopilación de librerías, elaboración del código y montaje práctico. - Análisis de resultados y conclusiones.	04/12/18	11/12/18
8. Ampliaciones y mejoras (Módulo 5)	13/12/18	18/12/18
8.1 Activación de una cámara IP mediante el mando a distancia. **8.2 Señalización acústica mediante zumbador.** **8.3 Programa para obtener la velocidad de desplazamiento del robot.**	13/12/18 14/12/18 15/12/18	13/12/18 14/12/18 16/12/18

Tabla 1. Entregables a nivel de producto

1.5 Lista de materiales

Seguidamente se relacionan los componentes de hardware y software que se han utilizado durante la realización del proyecto:

- Chasis metálico triangular con tres motores y ruedas omnidireccionales.
- Puente H298N (x2).
- Planchas redondas de madera utilizadas para ampliar la superficie útil del prototipo.
- Placa de micro controlador Arduino Mega 2500.
- Placa de alimentación de sensores para Arduino Mega (Shield sensor Mega).
- Placa nodeMCU (acceso WiFi).
- Placa de alimentación de sensores para nodeMCU (Shield sensor nodeMCU).
- Batería de 7.2 voltios (x2).
- Sensores infrarrojo SHARP 2Y0A21 F 84 (x4).
- Sensor láser (x1).
- Conversor DC/DC (x2)
- Módulo buzzer pasivo.
- Pantalla LCD (16X2).
- Interruptor de palanca.
- Cables con conexiones y otros materiales.

Equipos y herramientas:

- Fuente alimentación para pruebas.
- Osciloscopio.
- Polímetro de banco.
- Estación de soldadura.
- Ordenador.

Por otra parte, hay que mencionar los distintos programas que se han necesitado para realizar la parte del Software:

- Arduino IDE.
- MIT APP Inventor.
- Draw Io.

1.6 Descripción de los capítulos de la memoria

Para finalizar este capítulo introductorio, a continuación se hará una relación del contenido de los capítulos restantes que se expondrán a lo largo de la memoria.

El capítulo 2 se dedicará al estado del arte, haciendo una breve descripción de los sistemas de locomoción utilizados en los robots con ruedas, de las familias de micro controladores y tipos, de los sensores más comunes utilizados en un sistema evasor de obstáculos y, finalmente, una relación de productos en los que se podría aplicar el dispositivo motivo del presente proyecto.

En el capítulo 3 se describirán, por una parte, las placas de micro controladores que se van a utilizar. Por otro lado, se estudiará el entorno de programación utilizado y otro software adicional.

El capítulo 4 se dedicará, en su primer apartado, a la representación del diagrama de bloques del dispositivo y una descripción del producto obtenido, en su totalidad. El resto de apartados describirán el desarrollo del sistema de locomoción del robot.

El capítulo 5 englobará el estudio e implementación del sistema de detección de obstáculos, de suelo y regulación de velocidad mediante un sensor láser.

El capítulo 6 se empleará en explicar el modo en que se ha llevado a cabo la conexión *wifi* del dispositivo y la realización de la APP de control, así como la implementación de los distintos códigos en las dos tarjetas utilizadas.

El capitulo 7 se encargará de describir el desarrollo de un sistema de control de la alimentación del robot.

El capítulo 8 servirá para describir otras aplicaciones y utilidades implementadas en el robot.

El capítulo 9 tratará del estudio económico del proyecto.

Finalmente, el último capítulo se dedicará a las conclusiones donde se incluyen posibles aplicaciones del proyecto y mejoras que podrían añadírsele.

Al final de la memoria se relacionarán una serie de anexos donde se incluyen esquemas y otra información relacionada con el material utilizado en este proyecto.

2. Estado del arte de los robots móviles mediante tracción por ruedas

En este apartado se describirán el estado del arte de los robots móviles propulsados mediante ruedas. Para ello se hará una breve descripción de los distintos modelos locomotrices

existentes, de los sistemas embebidos utilizados para su control y de los distintos tipos de sensores empleados en la realización de un sistema evasor de obstáculos. Por último, al ser el motivo de este proyecto el diseño y construcción de un prototipo de robot con capacidad de evadir obstáculos, se hará una breve relación de modelos actuales de una de las principales aplicaciones que pueden derivarse de este tipo de diseños, los robots de limpieza autónomos.

2.1 Desplazamiento mediante ruedas

Los vehículos con ruedas son la solución más simple para conseguir la movilidad en terrenos suficientemente duros y libres de obstáculos, permitiendo conseguir velocidades relativamente altas. La limitación más importante lo constituye el deslizamiento que se produce durante la marcha, que dependiendo de las características del terreno puede ocasionar que el robot pierda tracción y se causen vibraciones en la estructura, siendo muy poco eficiente en terrenos blandos.

Los robos móviles pueden implementar distintos tipos de locomoción (configuraciones cinemáticas) mediante ruedas, que les confieren diferentes cualidades en lo que respecta al consumo energético, dimensiones, maniobrabilidad… En lo que respecta a la maniobrabilidad, los dispositivos con tracción omnidireccional son los que consiguen mayores capacidades, ya que pueden desplazarse simultánea e independientemente en cada eje del sistema de coordenadas y rotar según el eje perpendicular.

2.2 Características más significativas de las configuraciones cinemáticas más comunes

- **Configuración diferencial de las ruedas:** no dispone de ruedas directrices, lo que implica que el cambio de dirección se realiza modificando la velocidad de las ruedas de un lado con respecto a las del otro lado. Su principal ventaja es la facilidad de su implementación y su principal desventaja la falta de precisión y difícil control. En las figuras siguientes (imagen 1) puede verse un ejemplo de este tipo de configuración, con dos y cuatro ruedas, respectivamente.

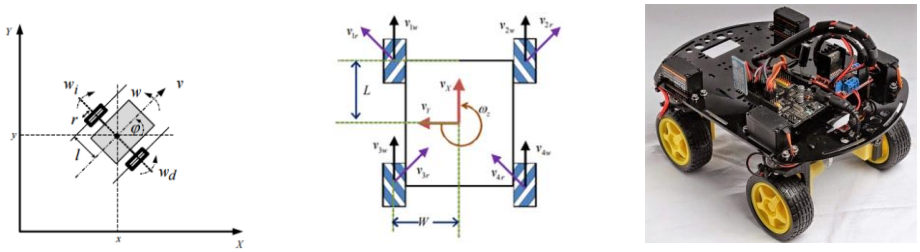


Imagen 1. Configuración diferencial con 2 y 4 ruedas

- **Configuración triciclo:** Su principal ventaja es la facilidad de diseño además de que el deslizamiento se reduce significativamente, y la principal desventaja de este sistema es la limitación en los movimientos y la inestabilidad. En las siguientes ilustraciones (imagen 2) se muestra un ejemplo de este tipo de configuración/implementación.

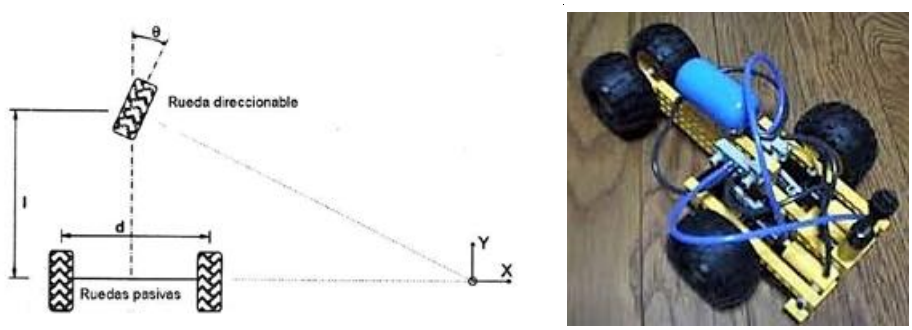


Imagen 2. Configuración triciclo

- **Configuración síncrona:** en esta configuración hay un único motor que controla la velocidad de todas las ruedas de modo síncrono y otro motor se encarga de la orientación de cada una de las ruedas. Se garantiza el desplazamiento en línea recta. Aunque el control de los motores es fácil de implementar, siendo esta su principal ventaja, en cambio su diseño y construcción son bastante complejos. En las figuras siguientes (imagen 3) puede verse el esquema de su diseño.

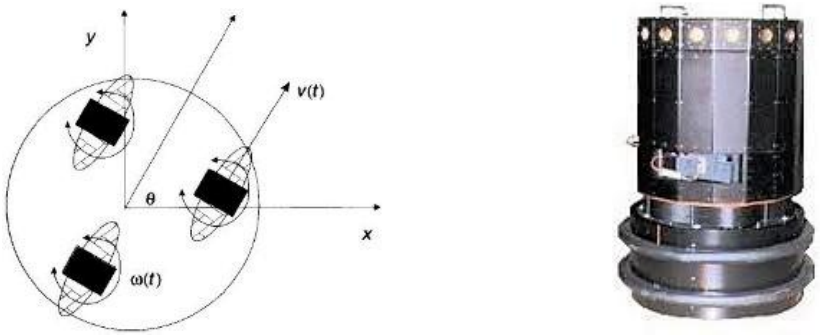


Imagen 3. Configuración síncrona

- **Configuración Ackerman:** Es la que nos resulta más familiar, ya que es la que suelen tener todos los automóviles. Esta configuración utiliza cuatro ruedas (dos en el eje trasero y otras dos en el delantero) de las cuales las delanteras son las utilizadas para direccionar el vehículo. La tracción puede implementarse tanto en las cuatro ruedas como en las delanteras o las traseras, independientemente. Su principal ventaja es que su diseño y realización son sencillos, además que permiten desarrollar velocidades elevadas. Por contra, su principal limitación es su reducida maniobrabilidad en comparación con otros sistemas. En la ilustración que sigue (imagen 4) puede verse un ejemplo de este sistema.

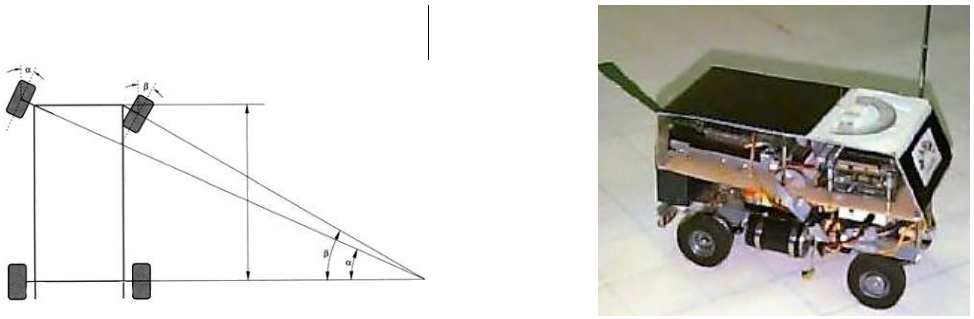


Imagen 4. Configuración Ackerman

- **Configuración omnidireccional:** su principal ventaja es su movilidad ya que admite movimientos muy complicados. No se garantiza el movimiento en línea recta y es necesario un control específico para conseguirlo. Además este tipo de sistema precisa de un tipo de ruedas especiales que le permitan rodar en cualquier dirección. Como ejemplo se muestra la imagen (5) siguiente.

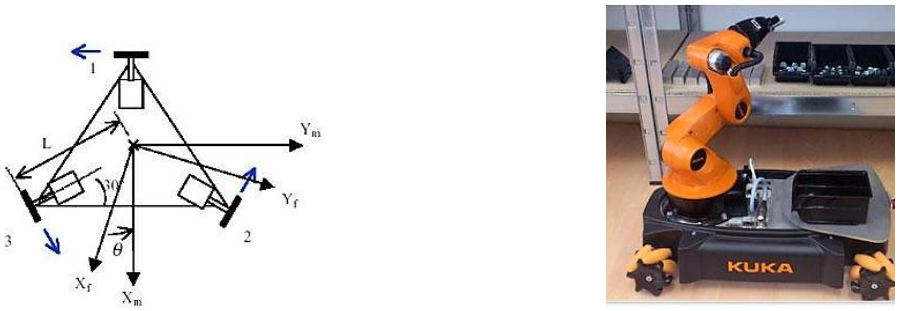


Imagen 5. Configuración omnidireccional

- **Locomoción por cintas de deslizamiento:** su principal ventaja es que es simple de controlar. Como inconvenientes pueden mencionarse el deslizamiento, que provoca resultados pobres en odometría, la falta de precisión de giro y el alto consumo de energía al realizar los giros.

Figura 6. Robot con locomoción por cintas de deslizamiento

2.3 Micro controlador y sistemas embebidos

La parte fundamental para conseguir el control de todos los sistemas de un robot es el micro controlador. Por esta razón se hará una breve introducción a los conceptos básicos sobre estos dispositivos, tanto en lo que concierne al dispositivo como tal (Hardware) y a la unión de estos con el sistema informático utilizado para su programación (sistema embebido). Además, se hará una relación de las principales familias de micro controladores existentes en el mercado.

Los parámetros más importantes del micro controlador son la velocidad de reloj, tamaño de palabra, tipos y capacidad de memoria, entradas analógicas, salidas analógicas PWM, I/O digitales, DAC, ADC, buses, UART y demás puertos de comunicaciones. Estos componentes se interconectan en la placa determinando los diferentes diseños que caracterizan a los numerosos modelos existentes en el mercado.

2.3.1 Familias de micro controladores

Al conjunto de los distintos patrones utilizados para interconectar los componentes de un micro-controlador se les denomina **familias**, las cuales comparten un conjunto de propiedades comunes en todos los dispositivos que la forman. Estas familias han ido evolucionando a medida que se extendía su uso. Cada familia está constituida por múltiples micro controladores que siguen unas especificaciones básicas, tales como el formato de las instrucciones o la arquitectura del procesador: una especificación y diseño de un núcleo, y el conjunto de funcionalidades que implementan, lo que se denomina *microcontroller unit* (MCU) o *core.* Las distintas especificaciones de cada familia pueden ser implementadas por uno o varios fabricantes que intentan mantener la compatibilidad entre los elementos que la forman y de este modo se evita que los usuarios tengan que cambiar todas las herramientas cada vez que cambian de procesador. En la siguiente tabla (2) se muestra los principales tipos de micro controladores, familia a que pertenecen y fabricante.

Fabricante	8 bits	16 bits	32 bits
Atmel	AVR(mega y tiny), 89Sxxxx familia similar 8051		SAM7 (ARM7TDMI), SAM3 (ARM Cortex-M3), SAM9 (ARM926), AVR32
Freescale	H8SX68HC05, 8HC08,68HC11, HCS08	68HC12, 68HCS12, 8HCSX12, 68HC16	683xx, PowerPC, ColdFire
Holtek	HT8		
Intel	MCS-48 (familia 8048)MCS51 (familia 8051)8xC251	MCS96, MXS296	
Microchip	Familia 10f2xx; Familia 12Cxx; Familia 12Fxx, 16Cxx, 16Fxx, 18Cxx y 18Fxx	PIC24F, PIC24H, dsPIC30FXX,dsPIC33F (con motor dsp integrado)	PIC32
National Semiconductor	COP8		
NXP Semiconductors	80C51	XA	Cortex-M3, Cortex-M0, ARM7, ARM9
Renesas	78K, H8	H8H8S, 78K0R, R8C, R32C/M32C/M16C	RX, V850, SuperH, SH-Mobile, H8SX
STMicroelectronics	ST 62, ST 7		STM32 (ARM7)
Texas Instruments	TMS370	MSP430	C2000, Cortex-M3 (ARM), TMS570 (ARM)
Zilog	Z8, Z86E02		

Tabla 2. Principales fabricantes de micro controladores y familias de productos.

2.3.2 Sistemas embebidos

Un sistema encastado, embebido o empotrado, es un sistema informático de uso específico que, por lo general, se basa en el control de un micro controlador en el que dicho sistema se encapsula. La denominación se debe a que generalmente suelen formar parte de un sistema

con funcionalidades más generales. Este sistema de computación es el resultado de la combinación de software y hardware, cuya finalidad es la de realizar una ó varias funciones dedicadas, frecuentemente en tiempo real. Las tareas realizadas por estos sistemas van desde las más simples, como por ejemplo la implementación de un sencillo sistema de alarma, hasta tareas mucho más complejas como puede ser el control de las distintas funciones de un osciloscopio digital. Esto es lo que los diferencia de los ordenadores personales que pueden desarrollar gran variedad de funciones.

La parte física, hardware del sistema encastado, suele estar constituida por una única tarjeta SBC (Single Board Computer) o placa de circuito impreso PCB (Printed Circuit Board). En la tarjeta se integran todos los elementos de hardware necesarios para efectuar la tarea específica para la que se diseña el sistema: unidad de cálculo o CPU (micro controlador o micro procesador), sistema de memoria, buses de datos, entradas y salidas (E/S) para conectar con sensores, actuadores y otros periféricos, y un sistema de regulación de la alimentación propio.

Actualmente es fácil encontrar una gran variedad de plataformas (basadas en micro controlador) para el desarrollo de sistemas embebidos. El uso de estas plataformas se ha popularizado, en gran medida, gracias a su bajo coste y facilidad de programación. Algunos ejemplos de estas plataformas son: Arduino, mbed, Raspberry Pi, BeagleBone, etc. En la imagen (7) pueden verse dos de los modelos más populares de estas plataformas de desarrollo.

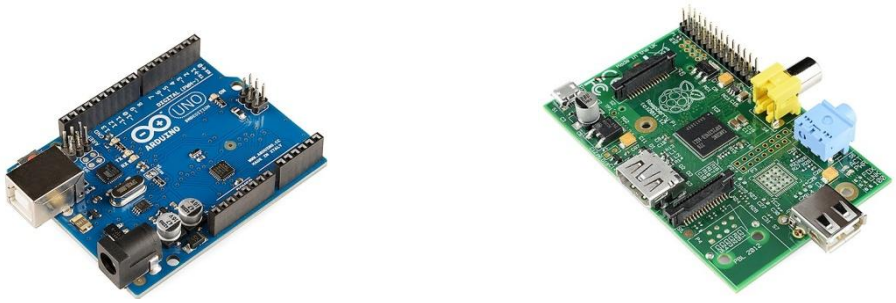


Imagen 7. Arduino Uno a la izquierda, y Raspberry Pi modelo A, a la derecha.

2.3.3. Entornos de programación

Suele ser habitual que los sistemas embebidos se puedan programar directamente en el lenguaje ensamblador del micro controlador incorporado sobre el mismo o, también, utilizando lenguajes como C ó C++ y compiladores específicos. Para llevar a cabo la programación puede que cada familia de procesadores tenga que usar un entorno de programación propio, que sólo sea compatible con las placas desarrolladas para esa familia en concreto, o por el contrario, utilizar un entorno de programación de propósito general que pueda ser utilizado por más de una familia de micro controladores.

2.4 Sensores

Sensores: son dispositivos que transforman magnitudes como la aceleración, inclinación, variaciones del flujo magnético, presión, sonido… en variaciones de tensión o intensidad eléctricas. La interpretación de estos parámetros se utiliza para controlar o modificar las distintas aplicaciones que se estén programando.

Entre los dispositivos más empleados para la detección y evasión de obstáculos se encuentran el sensor ultrasónico, el infrarrojo y el sensor laser.

2.4.1 Sensor ultrasónico

Los Ultrasonidos son ondas acústicas cuyo medio de propagación es el aire, cuya frecuencia no es audible por el oído humano (a partir de 20KHz). Los ultrasonidos se emplean en muchos sectores y aplicaciones, entre los que destaca su uso en la medicina como medio de generar imágenes mediante ecografías.

Los sensores de ultrasonidos se basan en la emisión de ultrasonidos mediante un emisor y un receptor. Un sensor ultrasónico emite impulsos ultrasónicos que rebotan en el objeto volviendo de nuevo al sensor. La velocidad de estos impulsos es conocida ya que estos se desplazan a la velocidad del sonido en el aire. El sensor cuenta el tiempo que ha pasado hasta que ha recibido el eco de la señal ultrasónica emitida. Conociendo el valor de ese tiempo y la velocidad de propagación del sonido en el aire, se puede calcular la distancia a un objeto:

$$Distancia = V_s \cdot \frac{t_v}{2}$$

$$343\frac{m}{s} \cdot 100\frac{cm}{m} \cdot \frac{1s}{10^6\ \mu s} = \frac{1cm}{29{,}2\ \mu s}$$

Sabiendo que se tardan 29,2 μs en recorrer cada centímetro y teniendo en cuenta que la onda recorre dos veces la distancia al objeto reflector (ida y vuelta):

$$velocidad = \frac{D(cm)}{t} \rightarrow D(cm) = \frac{1}{2} \cdot \frac{t(\mu s)}{29{,}2\ \mu s} cm$$

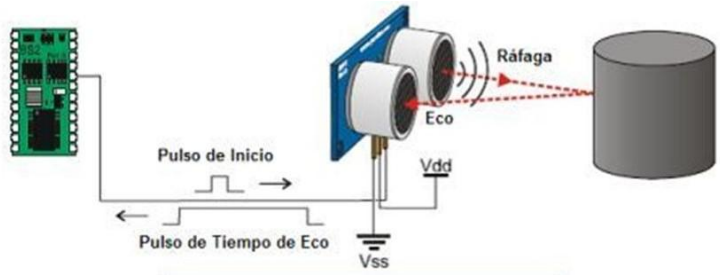


Figura 8. Esquema de funcionamiento de un sensor de ultrasonidos.

La principal ventaja de este tipo de dispositivos es su capacidad para funcionar correctamente en condiciones adversas, ya que la detección del obstáculo no depende ni de su color, ni de la luz solar, ni de factores ambientales.

Por otro lado, este tipo de sensores tienen ciertos inconvenientes que se deben tener en cuenta a la hora de su utilización:

- Estos sensores emiten las ondas con un haz cónico por lo que existe un cambio de sensibilidad en función del ángulo de de incidencia del haz, ya que en función de la distancia, el área que cubre la onda es mayor.
- Al alejarse del obstáculo el área aumenta y no se puede conocer exactamente la posición del objeto y si hay más de uno dentro de la superficie cubierta por el haz, solo se detectará el más cercano.
- Otra desventaja que se debe considerar son los errores en la lectura causados por la inclinación del obstáculo ya que puede ocasionar que el reflejo emitido no llegue hasta el receptor, por lo que este objeto seria invisible para el sensor ultrasónico.
- Por último, existe otro tipo de inconveniente a estudiar y es el hecho de que existen materiales que absorben el sonido, como por ejemplo las cortinas.

2.4.2 Sensor Infrarrojo

Las emisiones infrarrojas, con mayor longitud de onda que la luz visible, son imperceptibles para el ojo humano. Su longitud de onda se encuentra entre las ondas de radio y el espectro visible (imagen 9).

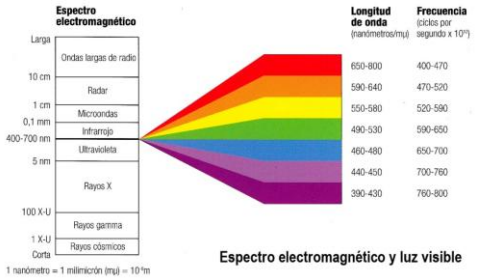


Imagen 9. Bandas del espectro electromagnético

Los sensores infrarrojos son unos componentes electrónicos constituidos, generalmente, de un LED emisor de infrarrojos y un fototransistor situado junto a aquel, actuando el LED como emisor y el fototransistor como receptor. El principio de funcionamiento de este tipo de sensores se basa en que la luz emitida por el LED, al incidir sobre una superficie blanca (mientras menos oscura más reflexión), se reflejará y llegará al fototransistor que entregará a su salida un valor analógico proporcional a la señal recibida. Si, en cambio, la superficie donde incide la emisión infrarroja es negra (mientras más oscura menos reflexión) será absorbida por ese material y el fototransistor no recibirá la luz ya que no habrá reflexión suficiente.

El sensor infrarrojo por lo tanto, como sensor de medición de distancia, se basa en un sistema de emisión/recepción de radiación lumínica en el espectro de los infrarrojos. Es un sensor muy utilizado para la medición de distancias pequeñas, en un rango entre 10 y 80 cm, y la detección de obstáculos debido, entre otras cosas, a su reducido precio. Debe tenerse en cuenta que el tipo de detección que realizan es direccional, es decir, sólo son capaces de detectar objetos que se encuentren frente al sensor.

Constructivamente, este tipo de detectores incorporan un filtro para reducir el ruido. La señal de salida (en el pin de señal) es un valor de voltaje analógico que es función de la posición en la que el rayo de luz incide sobre el obstáculo. Para medir la distancia la técnica más habitual es mediante la triangulación del haz de luz colimada que consiste en medir el ángulo con el que llega el reflejo, ya que dicho ángulo es diferente en función de la distancia al objeto. Uno de los inconvenientes de emplear este tipo de técnica de medición es que el ángulo de incidencia del eco detectado varía muy poco a grandes distancias por lo que es muy poco sensible a obstáculos lejanos. Otro método alternativo de estimación de la distancia a un objeto se basa en la medición de la cantidad de energía recibida tras rebotar la luz sobre aquel.

La principal ventaja de estos sensores es que muestran muy buenas cualidades en la detección de obstáculos compuestos por materiales que absorben el sonido, lo que los hace muy precisos a la hora de efectuar las medidas, además de la rapidez de respuesta si se comparan con los sensores de ultrasonidos.

Por otro lado, los inconvenientes más importantes de este tipo de sensores son:

- Son sensibles a la luz ambiente ya que la luz solar también se compone de luz infrarroja. Por este motivo son sensores que se utilizan habitualmente en interiores.
- La cantidad de luz infrarroja reflejada depende en gran medida de las características del objeto como su color, material y forma. Así, el receptor no es capaz de detectar

correctamente el eco cuando la superficie es oscura ya que la baja reflexión absorbe gran parte de la luz.

- Por último, como cualquier sensor óptico, este tipo de dispositivos fallan ante condiciones de iluminación pobres debidas a la presencia de factores ambientales como el humo o la niebla.

Imagen 10. Distintos tipos de sensores de infrarrojos detectores de obstáculos

2.4.3 Sensor láser

Un láser (del acrónimo inglés LASER, Light Amplification by Stimulated Emission of Radiation; amplificación de luz por emisión estimulada de radiación) es un dispositivo que utiliza un efecto de la mecánica cuántica, la emisión inducida o estimulada, para generar un haz de luz coherente (ondas luminosas con fase coherente y que por tanto conservan una relación de fase constante) tanto espacial (mantiene un tamaño pequeño al transmitirse) como temporal (concentra la emisión en un rango espectral muy estrecho). Este efecto supone una estimulación eléctrica o térmica de los átomos, iones y moléculas que conforman un material. Una de las características principales de un láser es su direccionalidad, es decir, es un rayo recto y concentrado.

Como ejemplos de dispositivos empleados para medir distancias, basados en tecnología láser, se pueden citar tres aplicaciones que se diferencian entre ellas por el rango de distancias en las que pueden trabajar, por los principios de funcionamiento empleados y por sus aplicaciones. Estos ejemplos son el telémetro láser (para distancias de hasta un kilómetro donde no importa la precisión), el distanciómetro láser (hasta 200 metros con una elevada precisión) y el sensor láser (donde la precisión milimétrica es fundamental).

En lo que respecta al método utilizado para medir la distancia, este dependerá de la precisión y la distancia máxima requerida por el sistema. Entre los principales métodos se encuentran la triangulación y el método basado en el principio del Time-Of-Flight (ToF). El método de triangulación es muy similar al explicado para los sensores infrarrojos, la luz reflejada por el objeto llega a la lente del receptor y según el ángulo con la que sea recibida determinará la distancia del obstáculo. Por otro lado, los sensores láser basados en el principio del ToF (figura 12) emiten un estrecho haz que viaja hacia el objeto cuya distancia se quiere conocer y se mide el tiempo transcurrido entre la emisión de la luz y la recepción de la señal rebotada. EL cálculo de la distancia, una vez medido el tiempo transcurrido, es directo.

Imagen 11.Tres modelos de sensores laser con aplicaciones e robótica.

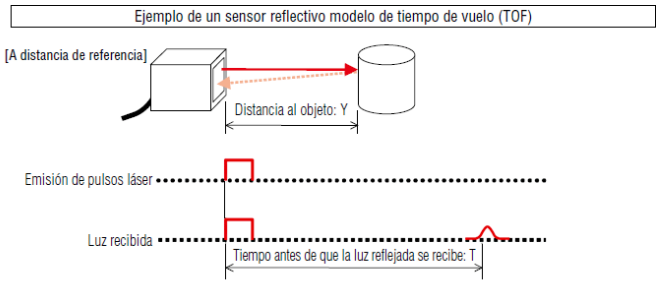


Imagen 12. Sistema de medición del tiempo (TOF).

La principal ventaja de estos sensores, gracias al elevado valor de la velocidad de la luz, es que la señal rebotada es devuelta en muy poco tiempo y esto es algo muy importante en aplicaciones en tiempo real, donde los tiempos de respuesta son fundamentales. Esto los hace muy precisos y direccionales, poco afectados por la mayoría de las perturbaciones.

Como inconvenientes pueden citarse:

- Son sensibles a las condiciones medioambientales del entorno donde se realiza la medida, nieve o lluvia por ejemplo, que puede provocar que el haz de luz rebote de manera incorrecta y afecte a la fiabilidad de las medidas.
- No pueden detectar obstáculos cuya superficie ofrezcan poco reflexión ya que buena parte de la radicación láser sería absorbida, haciendo que el eco retorne con poca intensidad al receptor.

2.5 Estudio de mercado

Existen muchos fabricantes que se dedican a la realización y comercialización de productos de muy diversa índole que utilizan la tracción con ruedas y la detección de obstáculos en sus realizaciones. Las aplicaciones a las que van dirigidos son innumerables, desde kits de montaje para iniciarse en el montaje y programación de pequeños robots hasta proyectos más serios de robots móviles dedicados a la video-vigilancia.

Una de las aplicaciones a las que se podría aplicar el presente proyecto es la de constituir la plataforma (tracción motora y la parte de evasión de obstáculos) de un robot de limpieza, por lo que las siguientes líneas se dedicarán a relacionar una serie de ejemplos relacionados con esta aplicación.

En la siguiente tabla (3) se representa una comparativa entre distintos modelos donde se exponen sus características básicas y su precio.

Modelo	Altura	Batería	Navegación	Filtro	Nivel de ruido	Precio
 **Taurus Mini Striker**	7 cm	75 min	No	No	50 dB	97,63 €
 **Conga Excellence 990**	7,4 cm	130 min	Sensores	HEPA	64 dB	234 €

 **iRobot Roomba 605**	9,2 cm	60 min	Sensores	AeroVac	60 dB	199 €
 **Ecovacs Robotics Deebot N79S**	7,8 cm	100 min	Sensores	HEPA	67 dB	173 €
 **ILife V5sPro**	10 cm	120 min	Sensores	HEPA	50 dB	169 €
 **Rowenta Smart Force Essential**	8 cm	150 min	Sensores	No	65 dB	179 €
 **iLife A6**	8 cm	140 min	Sensores	HEPA	68 dB	220 €
 **iRobot Roomba 680**	9,3 cm	60 min	Sensores	AeroVac	61 dB	323 €
 **iRobot Roomba 960**	9,1 cm	75 min	Mapeo con Wifi	AeroVac	70 dB	656 €

Tabla 3. Comparativa entre distintos modelos de robots de limpieza

3. Micro controladores, entorno de programación y otro software

En lo que respecta al micro controlador, elemento fundamental del proyecto, se decidió utilizar una tarjeta Arduino Mega 2560 para albergar el programa principal ya que dispone de un elevado número de pines, varios puertos y una alta capacidad de procesamiento. Por otra parte, se ha elegido una tarjeta de desarrollo NodeMCU para las comunicaciones inalámbricas, de la que se usará su módulo *wifi*, y se utilizará, además, para alguna aplicación secundaria más. Se barajaron otras opciones pero en todos los casos se consideró que no eran la mejor opción:

- Usar solo la placa de desarrollo nodeMCU, pero el número de pines resultaba insuficiente para la implementación de todos los sistemas.

- La Arduino Yun tampoco es adecuada por el mismo problema que la nodeMCU, además de su elevado precio.
- Utilizar un shield Wifi para Arduino, pero su precio suele ser demasiado elevado y se descartó por este motivo, principalmente.
- Un módulo de comunicaciones ESP que tiene un precio similar a una nodeMCU y que tiene muchas menos prestaciones que esta. No hay que olvidar que la nodeMCU, una vez instalada, puede ser utilizada en futuras ampliaciones o mejoras.

Seguidamente se hará una descripción de los elementos principales que serán utilizados en las diferentes fases de este proyecto. Esta descripción se centrará en la placa de micro controlador principal, la placa de desarrollo NodeMCU, la plataforma de desarrollo de Arduino (IDE) y el programa de desarrollo gráfico Draw.io.

Todos los demás elementos serán objeto de descripción a medida que vayan incorporándose al proyecto.

3.1 Micro controlador principal

La unidad principal de procesamiento será un micro-controlador Arduino Mega 2560 que se encargará de controlar todas las funciones del dispositivo.

3.1.1 Descripción general

La placa Arduino Mega 2560 (figura 13) es una placa electrónica basada en el chip Atmega2560. Cuenta con 54 pines digitales de entrada/salida (de los cuales 15 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (puertos serie de hardware), un oscilador de 16MHz, una conexión USB, un conector de alimentación, un conector ICSP, y un botón de reset. La placa Mega 2560 es compatible con la mayoría de los “shields” para el Arduino Uno y las placas anteriores Duemilanove o Diecimila.

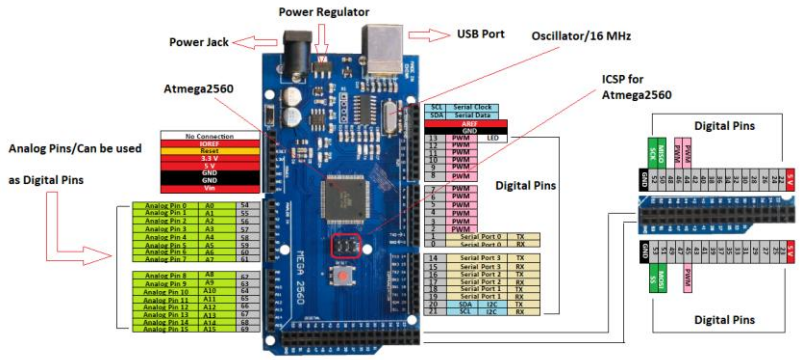


Imagen 13. Mapa de pines del Arduino Mega.

3.1.2 Especificaciones técnicas

Micro controlador	ATmega2560
Tensión de trabajo	5V
Tensión de entrada (recomendada)	7-12V
Tensión de entrada (límite)	6-20V
Pines Digitales I/O	54 (de los cuales 15 proporcionan salida PWM)
Pines de entradas Analógicas	16

DC Corriente por Pin I/O	20 mA
DC Corriente por Pin 3.3V	50 mA
Memoria Flash	256 KB (de los que 8 KB se usan por el bootloader)
SRAM	8 KB
EEPROM	4 KB
Velocidad del reloj	16 MHz
Largo	101.52 mm
Anchura	53.3 mm
Peso	37 g

Tabla 4. Especificaciones técnicas del Arduino Mega.

3.2 NodeMCU (V1.0/V3)

Para conseguir el control a distancia, y comunicación con el exterior, se usará el módulo de radio Wifi ESP8266 (que incorpora la placa NodeMCU) que será el encargado de transmitir los comandos a través de la UART.

3.2.1 Descripción general

NodeMCU es una placa de desarrollo totalmente abierta, a nivel de software y de hardware. Al igual que ocurre con Arduino, en NodeMCU (figura 14) todo está dispuesto para facilitar la programación de un micro controlador o MCU. Esta placa no es propiamente un micro controlador sino que se trata de una placa de desarrollo que lleva incorporado un chip que se suele llamar SoC (System on a Chip) que integra un micro controlador o MCU.

3.2.2 Características técnicas

Las características principales son las siguientes:

- Incorpora una MCU de 32-bit de bajo consumo.
- Módulo *wifi* de 2.4 GHz (módulo ESP-12).
- RAM de 50 KB.
- Una entrada analógica de 10 bit (ADC).
- 17 pines de entrada y salida GPIO (de propósito general).
- Conversor Serie-USB para poder programar y alimentar a través del USB.
- Fácil acceso a los pines.
- Pines de alimentación para sensores y componentes.
- LEDs para indicar estado.
- Botón de reset.
- Distribución de pines:

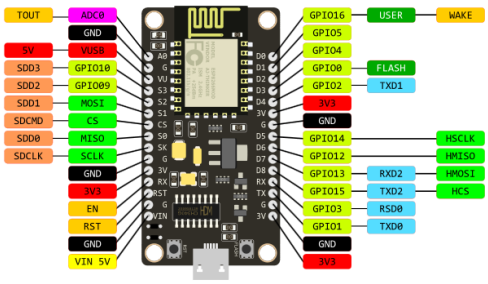


Imagen 14. Mapa de pines NodeMCU (V3).

3.3 Entorno de programación Arduino 1.8.1

Arduino IDE es una plataforma de programación de código abierto basado en facilitar el uso del software y hardware. El entorno se desarrolló con el lenguaje de programación Java y se basa en Processing, en el compilador avr-gcc y otros tipos de software de código abierto. Es destacable su facilidad de instalación en los distintos sistemas operativos, ya se trate de Windows, Mac OS o Linux.

La IDE Arduino se compone de un editor de texto para escribir los programas, una consola de texto, una barra de herramientas con los botones para las funciones comunes, un área de mensajes y otros menús con funciones adicionales. Desde la barra de herramientas se puede seleccionar el puerto y la placa con la que conectarse.

3.3.1 Escritura de Programas (Sketches)

Los programas escritos en el editor de texto de Arduino se denominan “sketches” y se guardan en archivos con extensión “.ino”. Ciertas características de la IDE habilitan para cortar, pegar, buscar y reemplazar el texto de los programas. El área de "feedback" proporciona información mientras se guarda el "sketch" o se exporta al hardware y muestra los errores que se hayan producido. La salida de texto y los mensajes de error, en su caso, serán mostrados por consola. La placa conectada a la IDE, y el puerto serie, se muestra en la esquina derecha inferior de la ventana. Al “Monitor serie” y al “Serial Plotter” se accede a través de la barra “Herramientas”, además de poder dar formato automático al código, acceder a los archivos del programa, seleccionar el tipo de placa, elección del puerto, etc.

3.3.2 Almacenamiento de programas

Los programas en Arduino se almacenan en una ubicación denominada “Proyecto” que ayuda a organizar el trabajo. Los programas se pueden abrir desde el menú "Archivo>Proyecto" o desde el botón "Abrir" de la barra de herramientas. La primera vez que se ejecute el software de Arduino se creará automáticamente un directorio para el proyecto. Desde el cuadro de diálogo “Preferencias” se puede consultar o modificar la situación de los programas.

3.3.3 Compilación

Se pueden gestionar programas compuestos por más de un archivo (cada uno de los cuales aparece en su propia pestaña), ya sean archivos normales Arduino (sin extensión), archivos de C (extensión .c), archivos de C ++ (.cpp), o archivos de cabecera (.h).

3.3.4 Subida de los programas a la placa

Antes de subir un programa, hay que seleccionar "Herramientas > Placa” y “Herramientas> Puerto Serie”. Una vez que ha seleccionado la placa y el puerto serie correctos, se tiene que pulsar el botón de carga en la barra de herramientas o seleccionar la opción "Cargar" en el menú "Archivo". La placa se restablecerá automáticamente y comenzarán la carga. En la mayoría de las placas se verá como se encienden el diodo Led RX y el TX mientras el "sketch" se carga. El entorno de Arduino mostrará un mensaje de carga completa o de error cuando finalice.
Durante la subida de un programa a la placa, la IDE utiliza un pequeño programa llamado "bootloader" cargado en el micro controlador de la placa. Este programa es el único necesario para subir el código a la placa. El "bootloader" se activa durante unos segundos mientras la placa se “resetea” y una vez finalizada esta acción el "sketch", que se acaba de cargar en el micro controlador, se inicia y comienza a funcionar.

3.3.5 Librerías

Las librerías proporcionan funcionalidades adicionales para su uso en los programas, ya sean de trabajo con el hardware o manipulación de datos. Para utilizar una librería en un programa hay que seleccionarlo en el menú “Programa > Incluir Librería ". Esto insertará una o más sentencias "#include" en la parte superior del programa y compilará la librería junto con el "sketch". Debido a que las librerías se cargan en la placa con el programa se aumenta la cantidad de espacio que el programa ocupa. Si un programa ya no necesita una librería sólo hay que borrar la instrucción "#include" en la parte superior del código.

3.3.7 Lenguaje de programación Arduino

Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel "Processing", aunque es posible utilizar otros lenguajes de programación debido a que Arduino usa la transmisión serial de datos, soportada por un gran número de lenguajes de programación. El lenguaje de programación "Processing" es un lenguaje de programación y entorno de desarrollo integrado, de código abierto basado en Java, que es fácil de utilizar y sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital.

Imagen 15. IDE Arduino.

3.4 Draw.io

Todos los esquemas y diagramas realizados en este proyectos se han realizado utilizando Draw.io.

Se trata de una aplicación gratuita para elaborar diagramas online. Con esta aplicación se pueden realizar diagramas sin necesidad de instalar el software en el PC. Dispone de una interfaz muy sencilla de utilizar, además de ser tan completa que no tiene mucho que envidiar a cualquier software de pago para escritorio.

Tiene muchas opciones para elaborar completos diagramas y dispone de una gran variedad de formas y diseños predeterminados que luego se pueden moldear al gusto de cada usuario. Permite agregar rápidamente imágenes externas utilizando el buscador de Google y tiene, además, múltiples opciones de texto que se pueden configurar como se desee.

No es necesario ningún tipo de registro para utilizar esta aplicación y los trabajos realizados pueden ser guardados en formato .XML para poder modificarlos posteriormente. Pueden imprimirse los diseños realizados o exportarlos en distintos formatos tales como .PNG, .GIF, .JPG, .PDF y .SVG, o si se prefiere se puede insertar el diagrama realizado en cualquier sitio web utilizando un código que genera la aplicación.

4. Desarrollo Hardware y software del prototipo

4.1 Esquema de bloques del dispositivo

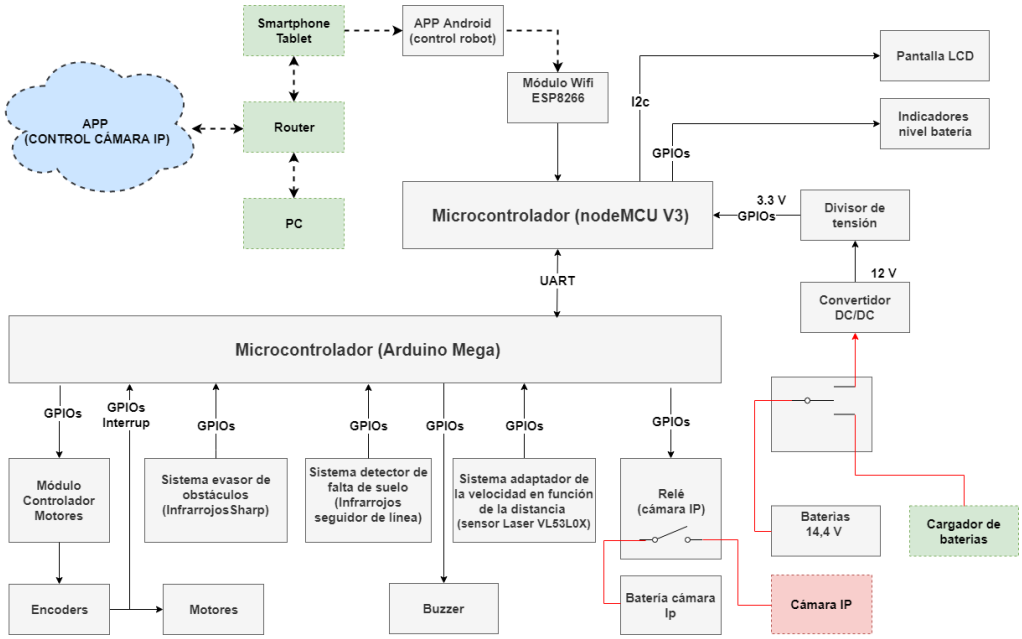


Imagen 16. Esquema de bloques del sistema.

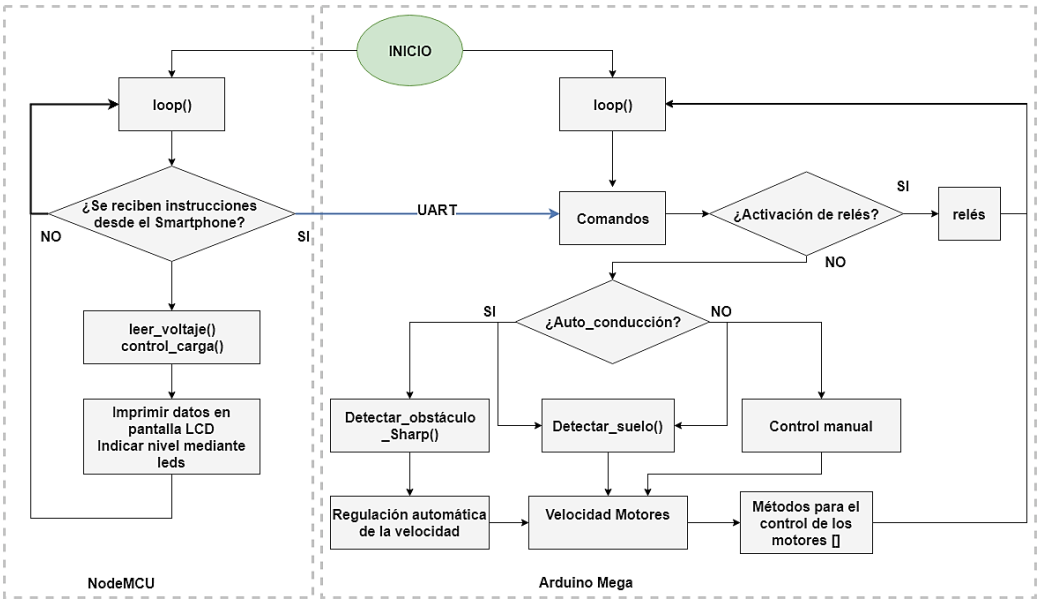


Imagen 17. Diagrama de flujos general del sistema.

Descripción de las distintas partes:

El objeto de este proyecto se ha centrado en crear una plataforma robótica que debe contar con una seríe de elementos que le permitan moverse, detectar obstáculos y que se pueda controlar mediante el uso de un mando a distancia vía radio *wifi*.

En el esquema (imagen 17) se han incluido todos los bloques constituyentes que pueden ser enumerados de la siguiente manera:

- Bloque de procesamiento principal, constituido por la placa Arduino Mega.
- Módulo de control del movimiento, formado por los puentes H controladores de potencia, 3 motores y las correspondientes ruedas y el software necerio para su control.
- Un sistema evasor de obstáculos (sotware), construido mediante 4 sensores infrarrojos tipo Sharp.
- Un sistema detector de falta de piso o suelo, implementado con 4 sensores seguidores de linea y el programa correspondiente para su control.
- Un sistema adaptador de la velocidad en función de la distancia, elaborado con un sensor láser tipo VL53L0X y el programa de control.
- Para el control del robot se ha creado un mando a distancia (APP) mediante la plataforma de desarrollo APP Inventor, para instalar en cualquier dispositivo movil que funciones con el sistema operativo Android.
- Para poder controlar el robot se ha creado un software instalado en la placa nodeMCU con la que se pueden recibir los datos enviados mediante el Smartphone o tablet utilizadas como mando a distancia.
- Se han conectado las dos placas, Arduino y nodeMCU mediante puerto serie para así poder recibir en la Arduino Mega los comandos enviados desde el dispositivo movil.
- Se ha creado un software específico para tratar los comandos u ordenes recibidos en la placa Arduino Mega y de este modo poder controlar el robot.
- Se ha implementado un sistema de control de la carga de las baterias mediante un software instalado en la placa nodeMCU. Para poder medir el voltaje de las baterías en la placa nodeMCU se ha utilizado su única entrada analógica, adaptando el voltaje entregado por la fuente de alimentación a los 3.3 V máximo que admite esta placa, mediante un divisor de tensión. El sistema de alimentación lo confomarán dos baterías de 7,2 voltios conectadas en seríe y un convertidor DC/DC que limitará la tensión de alimentación del robot a 12 voltios.Otro elementos constituyente de este sistema es una pantalla LCD donde se visualizará el nivel de carga en tantos por ciento, el voltaje de la fuente de alimentación y un reloj indicador del tiempo transcurrido desde que se conectó el robot.
- El robot dispone de una placa de relés de 4 elementos, para controlar el encendido de una camara IP y otras funciones como la habilitación de la carga de las barerías.
- Otro elemento incorporado lo constituye un avisador acústico, buzzer, controlado por un sencillo programa.
- Por último, se realizó un sotware, como objetivo secundario, para poder medir la velocidad de desplazamiento de dispositivo y su ángulo de giro.

A continuación, en la imagen (18), se muestra un diagrama general de todos los componentes del robot.

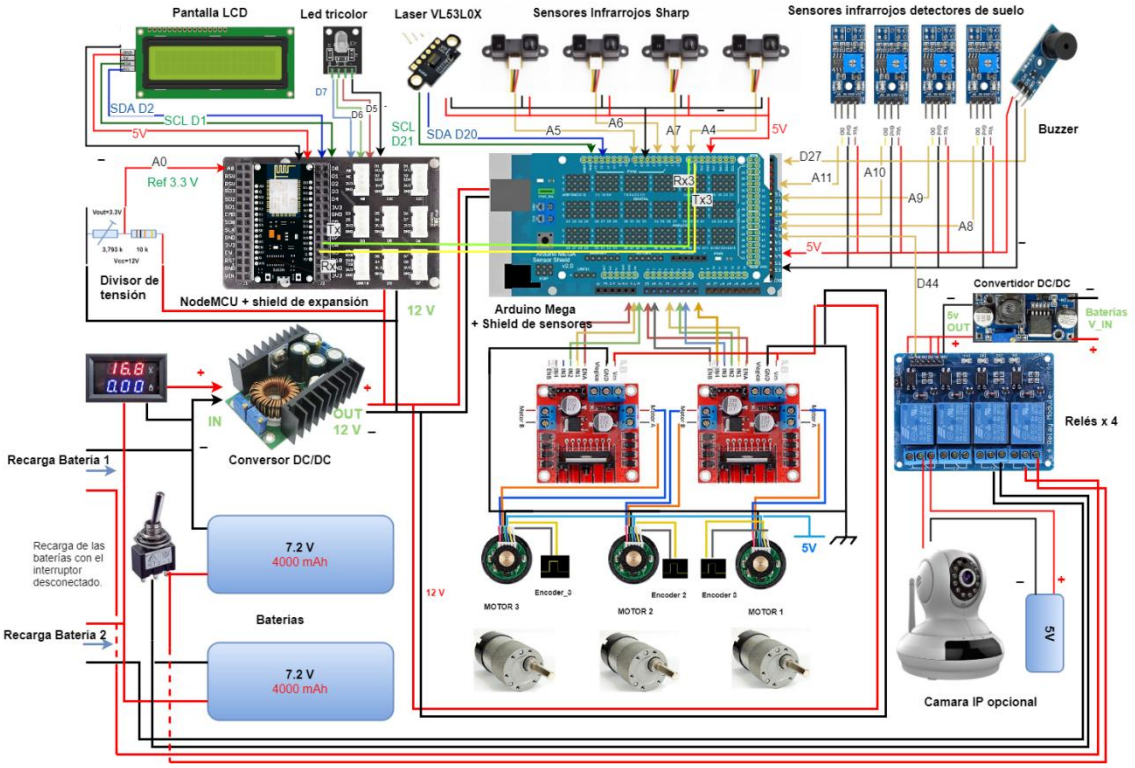


Imagen 18 . Diagrama de conexionado general de todo el sistema.

Seguidamente se han realizado dos tablas (5 y 6) con los pines utilizados en ambas placas y su aplicación.

Pines utilizados del Arduino Mega					
PIN	**IN/ OUT**	**TIPO**	**APLICACIÓN**	**CONECTADO A:**	**DEF. EN ARDUINO**
Control de desplazamiento del robot:					
3	OUT	PWM	Dirección motor 3	IN1, puente H 298N (2)	IN5
4	OUT	PWM	Dirección motor 3	IN2, puente H 298 N (2)	IN6
5	OUT	PWM	Velocidad motor 1	ENA, puente H 298 N(1)	ENA
6	OUT	PWM	Velocidad motor 2	ENB, puente H 298 N (1)	ENB
7	OUT	PWM	Velocidad motor 3	ENA, puente H 298 N (2)	ENC
8	OUT	PWM	Dirección motor 1	IN1, puente H 298 N (1)	1N1
9	OUT	PWM	Dirección motor 1	IN2, puente H 298 N (1)	1N2
10	OUT	PWM	Dirección motor 2	IN3, puente H 298 N (1)	1N3
12	OUT	PWM	Dirección motor 2	IN4, puente H 298 N (1)	1N4
Sistema detector de obstáculos:					
A4	IN	Analog	Lectura voltaje del sensor	Salida de señal del Sharp Izquierda	4
A5	IN	Analog	Lectura voltaje del sensor	Salida de señal del Sharp Derecha	5
A6	IN	Analog	Lectura voltaje del sensor	Salida de señal del Sharp Cent/Derecha	6
A7	IN	Analog	Lectura voltaje del sensor	Salida de señal del Sharp Cent/Izquierda	7
Sistema detector de falta de piso:					
A8	IN	Analog	Lectura estado sensor	Salida de señal del seguidor de linea 1	DsSensor_1
A9	IN	Analog	Lectura estado sensor	Salida de señal del seguidor de linea 2	DsSensor_2
A10	IN	Analog	Lectura estado sensor	Salida de señal del seguidor de linea 3	DsSensor_3
A11	IN	Analog	Lectura estado sensor	Salida de señal del seguidor de linea 4	DsSensor_4
Sistema de adaptación de la velocidad en función de la distancia al obstáculo:					
D20	Serial	SDA	BUS I2C	Pin SDA del sensor láser VL53L0X	

D21	Serial	SCL	BUS I2C	Pin SCL del sensor láser VL53L0X	
Conexión mediante puerto serie de la placa Arduino Mega y nodeMCU:					
D14	OUT	TX3	Conexión con nodeMCU (transmisión)	Pin RX de la placa nodeMCU	
D15	IN	RX3	Conexión con nodeMCU (recepción)	Pin TX de la placa nodeMCU	
Otros :					
D26	OUT	Digital	Activación buzzer	Entrada de señal del buzzer	buzzer 26
D42	OUT	Digital	Activación relé_1	Entrada tarjeta relés (sin aplicar)	rele1_ON 42
D43	OUT	Digital	Activ. Aliment. cámara IP	Entrada tarjeta relés 1	Camara_ON 43

Tabla 5. Pines utilizados del Arduino Mega.

Aplicación de los pines de la nodeMCU					
PIN	**IN/ OUT**	**TIPO**	**APLICACIÓN**	**CONECTADO A:**	**DEF. EN nodeMCU**
Control del nivel de voltaje de la fuente de alimentación:					
A0	IN	Analog	Lectura voltaje fuente de alimentación	Divisor de tensión de 12 a 3,3 Voltios, a la salida del convertidor DC/DC	pinV=A0
D1	Serial	SCL	BUS I2C	Pin SCL de la pantalla LCD	
D2	Serial	SDA	BUS I2C	Pin SDA de la pantalla LCD	
D5	OUT	Digital	Activación LED rojo	Pin red del led tricolor	ledR=14
D6	OUT	Digital	Activación LED azul	Pin Blue del led tricolor	ledA=12
D7	OUT	Digital	Activación LED verde	Pin Green del led tricolor	ledV=13
Conexión mediante puerto serie de la placa nodeMCU Y Arduino Mega:					
TX	OUT	TX	Conexión con Arduino Mega (transmisión datos)	Pin RX3 de la placa Arduino Mega	
RX	IN	RX	Conexión con Arduino Mega (recepción de datos)	Pin TX3 de la placa Arduino Mega	

Tabla 6. Aplicación de los pines de la nodeMCU.

Producto obtenido (prototipo):

A nivel de Hardware, el prototipo implementado tiene el aspecto de la Imagen (19).

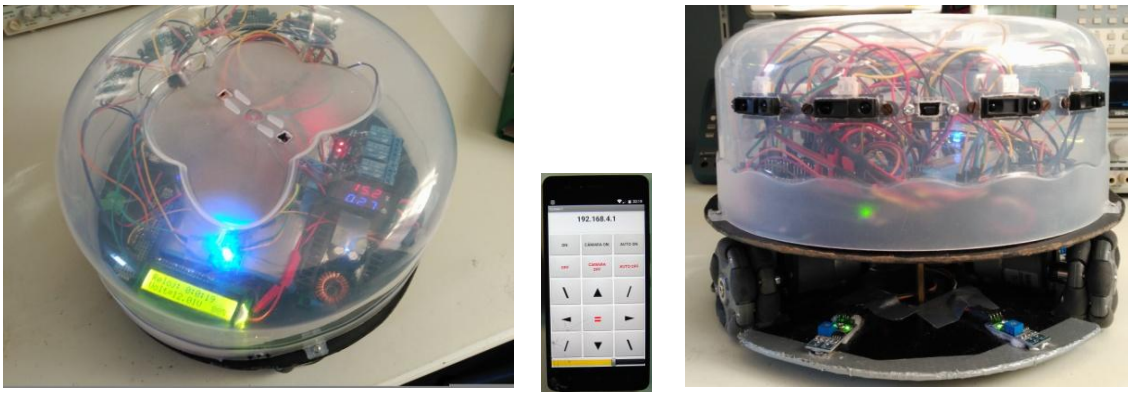


Imagen 19. Aspecto final del prototipo y de la APP de control para Android.

En los siguientes apartados se explicará el procedimiento seguido para la realización completa del prototipo, estructurando cada apartado en requerimientos, elaboración del programa de control, montaje práctico y conclusiones sobre los resultados obtenidos.

4.2 Tracción del robot (Módulo 1)

Como ya se indicó en el apartado correspondiente del estado del arte, existen varias soluciones cinemáticas para lograr que un robot tenga movimiento, para este trabajo se consideran sólo dos de estos sistemas, el diferencial y el omnidireccional:

4.2.1 Opciones

- **Desplazamiento diferencial:** Un desplazamiento diferencial (imagen 20) considera un arreglo par de ruedas. El principio de funcionamiento es simple: para que el robot se desplace hacia adelante, o hacia atrás, conservando su orientación las ruedas deben girar a la misma velocidad y en la misma dirección. Para que el robot cambie su orientación debe existir una diferencia de velocidades en las ruedas: mientras más grande sea el diferencial de velocidades más grande será el cambio en la orientación del robot.

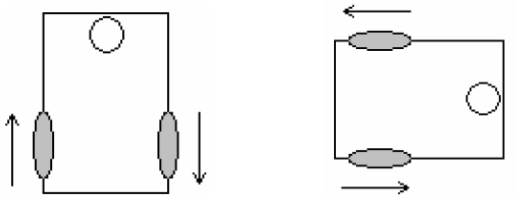


Imagen 20. Movimientos Desplazamiento diferencial.

- **Desplazamiento omnidireccional:** El desplazamiento omnidireccional es de gran interés porque brinda una completa maniobrabilidad. Los robots omnidireccionales pueden moverse en cualquier dirección y en cualquier momento sin requerir una orientación específica para lograrlo. Este tipo de desplazamiento requiere de ruedas que se puedan mover en más de una dirección. La figura (21) muestra el diseño de una rueda omnidireccional y la otra, imagen (22), ilustra uno de los movimientos que puede realizar.

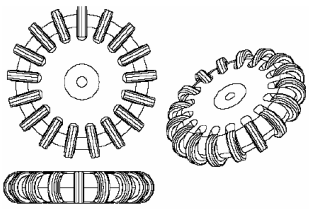


Imagen 21. Ruedas diferenciales

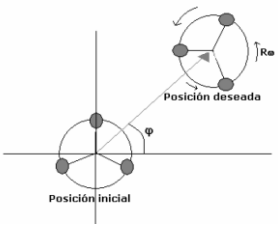


Imagen 22. Tipo de desplazamiento omnidireccional

Se requiere de más de dos ruedas omnidireccionales para mover a un robot. Cada rueda proporciona una fuerza en una dirección normal al eje del motor y paralela al piso. La suma de fuerzas provén la traslación y rotación del robot.

Existe una relación inversa entre la maniobrabilidad y el control, ya que los diseños omnidireccionales requieren un procesamiento adicional para convertir las velocidades de rotación y traslación del robot en velocidades individuales para cada rueda. Controlar un robot omnidireccional para que se mueva en una dirección deseada es más complicado que los métodos diferenciales. Para ello es necesario establecer un modelo cinemático omnidireccional, lo cual queda fuera del ámbito de este proyecto. La tarea a realizar se limitará a definir una serie de movimientos concretos utilizando las tres ruedas motrices.

4.2.2 Configuración elegida

Para el chasis se barajaron dos posibles opciones:

- Un primer modelo de plástico transparente, con cuatro motores. Este tipo de chasis, muy económico, suponía dos problemas, por una parte el espacio para albergar componentes resultaba insuficiente y por otro lado, la razón más importante para su descarte, la baja calidad de los motores y el alto consumo de energía de los mismos.

Imagen 23. Dos modelos de chasis disponibles para realizar el proyecto.

- Como segunda opción, un chasis metálico triangular con tres ruedas omnidireccionales (imagen 23 derecha). En este caso la calidad de los motores y sus engranajes reductores, los encoders que incorporan y el consumo (mucho más comedido que en el caso anterior) hicieron decidirnos por esta segunda opción. Posteriormente se solventaría el problema de espacio incorporando dos planchas de madera redondas sobre la estructura triangular.

La implementación final será un compromiso entre las capacidades de movilidad que ofrece este tipo de chasis (mayor maniobrabilidad gracias al tipo de ruedas que monta) y la definición de una serie de movimientos como si de un sistema diferencial se tratase.

4.3 Etapa Controladora de los motores

Su finalidad es el control del movimiento del robot. Las tareas realizadas se han basado en el cumplimiento de los siguientes requisitos:

- En primer lugar se debe definir la parte delantera del robot (una de las tres caras de un chasis con forma de triangulo equilátero) ya que los movimientos de los motores estarán condicionados por esa elección.
- El robot deberá realizar ocho movimientos básicos: adelante, atrás, girar a la derecha, girar a la izquierda, adelante/Izquierda, adelante/derecha, atrás/izquierda, atrás/derecha, además del estado de parada.
- Los movimientos de giro utilizaran los tres motores conjuntamente y los demás dos motores, dependiendo su activación y sentido de giro de la ubicación en el chasis.
- El software creado deberá ser capaz de establecer la velocidad de giro de las ruedas y el sentido del mismo, de modo que se consigan los distintos movimientos antes mencionados.
- Aunque parezca obvio, señalar que las señales de control de los motores no se aplicarán directamente a los mismos, sino que se hará a través de unos módulos de potencia específicos para tales cometidos.

En resumen, las tareas realizadas consistirán en crear los métodos para cada movimiento y uno específico para poder establecer distintas velocidades de desplazamiento.

4.3.1 Movimientos definidos

En la tabla (7) de abajo se enumeran las posiciones de los motores en el chasis y los movimientos correspondientes.

MOVIMIENTO		MOTORES ACTIVOS
goAhead	(Adelante)	1, 2
goBack	(Marcha atrás)	1, 2
stopRobot	(Parar motores)	1, 2, 3
turnLeft	(Girar izquierda)	1, 2, 3
turnRight	(Girar derecha)	1, 2, 3
goAheadLeft	(Ir hacia adelante y a la izquierda)	2, 3
goAheadRight	(Ir hacia adelante y a la derecha)	1, 3
goBackLeft	(Ir hacia atrás y a la izquierda)	2, 3
goBackRight	(Ir hacia atrás y a la derecha)	1, 3

Tabla 7. Definición de movimientos y motores implicados.

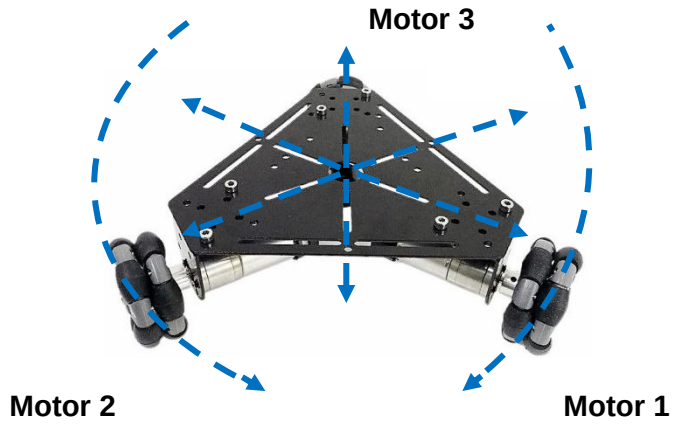


Imagen 24. Chasis con ruedas omnidireccionales.

4.4 Código de control

En la figura (24) se muestra los distintos métodos utilizados para cada uno de los movimientos. En este diagrama puede verse como las instrucciones se han representado mediante una serie de triángulos, donde cada uno de los cuales llama a la función correspondiente al mismo tiempo que a “velocMotores()”. Si no se recibe ninguna instrucción de movimiento se llamará a la función “stopRobot()” que pondrá todas las salidas a LOW (estado bajo). Para facilitar su comprensión se han representado todos los estados de los pines digitales, configuradas como salidas, en cada uno de los métodos creados. Las salidas digitales se conectaran a las correspondientes entradas del puente H utilizado para controlar los motores.

Dado que se están utilizando tres motores se han creado tres pares de señales de control del sentido de giro de los motores y tres señales de control del PWM, cuya utilidad, tipo de señal y valores que pueden tener se han recogido en la tabla (8) siguiente:

Señal (a los pines del puente H)	Motor	Aplicación	Valores
IN1	1	Sentido giro	Digital (HIGH/LOW)
IN2	1	Sentido giro	Digital (HIGH/LOW)
IN3	2	Sentido giro	Digital (HIGH/LOW)
IN4	2	Sentido giro	Digital (HIGH/LOW)

IN5	3	Sentido giro	Digital (HIGH/LOW)
IN6	3	Sentido giro	Digital (HIGH/LOW)
ENA	1	PWM	Analógica (0-255)
ENB	2	PWM	Analógica (0-255)
ENC	3	PWM	Analógica (0-255)

Tabla 8. Descripción de cada tipo de señal y su aplicación en el programa de control de los motores.

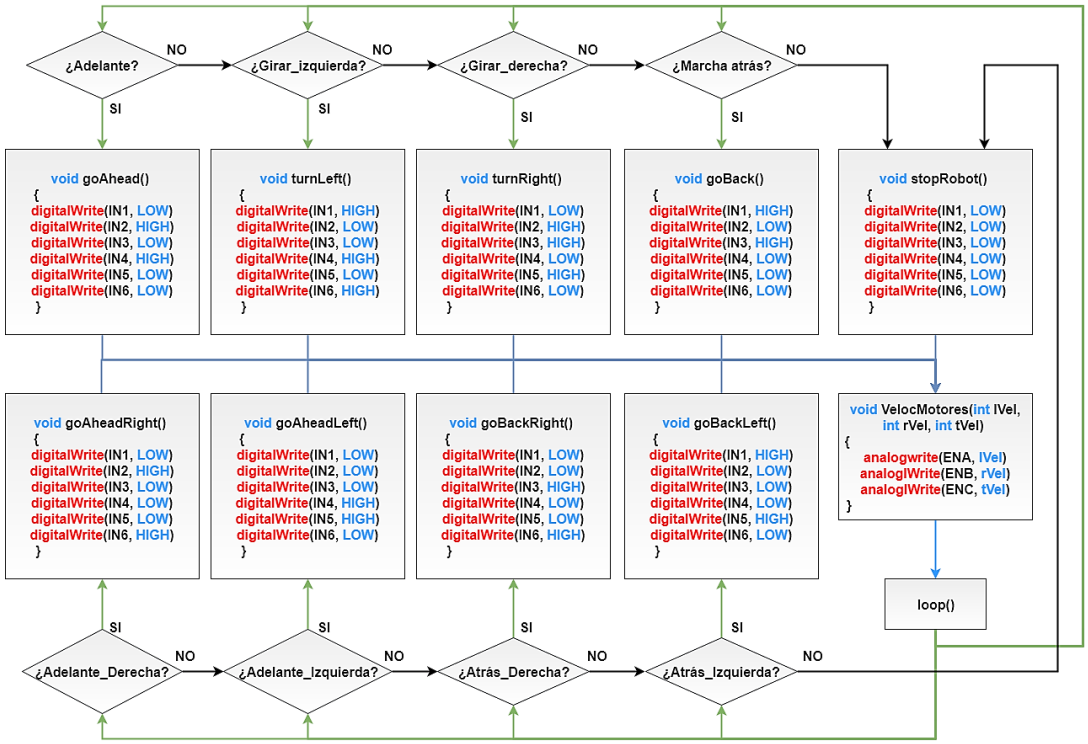


Imagen 25. Definición de los métodos de control de los motores.

4.5 Elementos de Hardware utilizados para su implementación

4.5.1 Chasis y motores con 3 ruedas omnidireccionales

El tipo de chasis puede verse en la figura (24) del apartado 4.3.1. Este chasis ha sido complementado con dos planchas circulares de madera, de 27 cm de diámetro y 3 mm de grosor, a fin de conseguir mayor superficie para la instalación de los distintos elementos y al mismo tiempo lograr mayor facilidad en el desplazamiento del dispositivo.

Para el presente proyecto se van a utilizar 3 motores con encoders. Las características, según el fabricante, se muestran en la ilustración (26) de abajo.

减速电机仕样性能表 Specifications

型号：GM37-520 霍尔编码减速直流电机 低功耗 大扭力 启动响应快

警告： 12V 转速36转（减速比1：270以上）均不允许堵转及超过35Kg.cm 使用，否则很容易打碎齿轮，造成电机损坏损毁！

使用电压:DC12.0V时候 标准功率7W 测试规格参数表

减速比（变比）	1：6.3	1：10	1：18.8	1：30	1：50	1：90	1：150	1：270	1：450	1：650	1：810
空载电流（mA）	≤160	≤160	≤160	≤160	≤160	≤160	≤160	≤160	≤160	≤160	≤160
空载转速(rpm)	1530	960	510	320	190	107	65	35	21	15	12
额定转矩(Kg.cm)	0.4	0.7	1.2	2.0	3.0	5.0	9.0	17.0	28.0	41.0	54.0
额定转速（rpm）	1150	720	385	240	140	80	45	26	16	11	9
额定电流（A）	≤0.7	≤0.7	≤0.7	≤0.7	≤0.7	≤0.7	≤0.7	≤0.7	≤0.7	≤0.7	≤0.7
最大转矩（kg.cm	1.3	2.0	3.8	6.0	10.0	18.0	30.0	不可超过35Kg			
停转电流（A）	3.5	3.5	3.5	3.5	3.5	3.5	3.5	不可堵转使用			
霍尔分辨率 PPR	69.3	110.0	206.0	330.0	605.0	990.0	1650.0	2970.0	4950.0	7150.0	8910.0
减速器长度（L）	19.0	19.0	19.0	22.0	22.0	24.0	26.5	26.5	29.0	29.0	29.0

Imagen 26. Datos técnicos de los motores.

El engranaje del motor utilizado tiene una relación de reducción de 1:90, lo que hace que gire a una velocidad (se supone sin carga) de 107 rpm. Los pulsos generados por el encoder, en cada vuelta del eje después de la reductora, son 990 PPR (pulsos por revolución). Esto último quiere decir que el encoder genera 11 pulsos por vuelta de eje antes de la reductora. Los demás datos corresponden a consumos y torque del motor.

Los encoders son componentes que se añaden a un motor de corriente continua para convertir el movimiento mecánico en pulsos digitales que pueden ser interpretados por un sistema electrónico de control. El encoder que montan estos motores hay que alimentarlo con una tensión de 5 voltios. El dispositivo genera dos señales de salida (A y B) desfasadas 90 grados una de la otra (imagen 27). Las salidas se conectan a los pines con interrupciones de la placa arduino Mega que dispone de pines para interrupciones referenciados con los números 2, 3, 21, 20, 19 y 21.

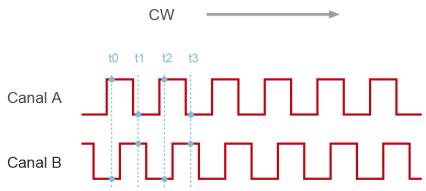


Imagen 27. Desfase entre la dos señales generadas por los canales A y B del encoder.

Estos dispositivos, además de ayudar a regular el giro de los motores, también sirven para determinar la distancia y orientación que describe una rueda montada en un motor con este dispositivo. Una aplicación muy importante en la robótica móvil, que se obtiene gracias al uso de los encoders, es la odometria que trata la estimación la posición de los vehículos con ruedas. Sobre la medición de la velocidad y el sentido de giro se ha realizado un sencillo programa incluido en el capítulo 8 de este trabajo, donde se tratan los objetivos secundarios del proyecto.

4.5.2 Módulos controladores de motores L298 N

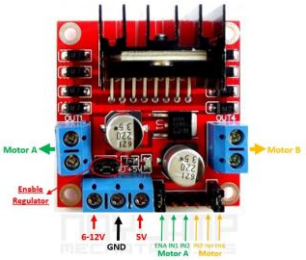


Imagen 28. Controlador de motores *L298N*

Se utilizaran dos unidades para controlar los tres motores. Este módulo (figura 28) se basa en el Chip 298N que permite controlar dos motores de corriente continua o un motor paso a paso bipolar de hasta 2 amperios. El módulo cuenta con todos los componentes necesarios para funcionar sin necesidad de elementos adicionales. Dispone de jumpers de selección para habilitar cada una de las salidas del módulo (A y B). La salida del motor A se corresponde con las entradas IN1y IN2 y la salida B con IN3 y IN4. Los pines de

habilitación son ENA y ENB, respectivamente. Para controlar la velocidad del motor se tiene que hacer uso de PWM que se aplicarán a los pines de activación de cada salida (ENA y ENB) con lo que los “jumpers” de activación ya no se utilizan.

4.6 Montaje práctico, conexionado de los distintos elementos

El conexionado de los componentes se realiza tal y como se muestra en la ilustración adjunta, (imagen 29) por lo que no se considera necesario entrar en más detalles constructivos. Sólo indicar que ha sido necesario utilizar dos puentes H para poder controlar el tercer motor, del que sólo se ha utilizado una de las salidas.

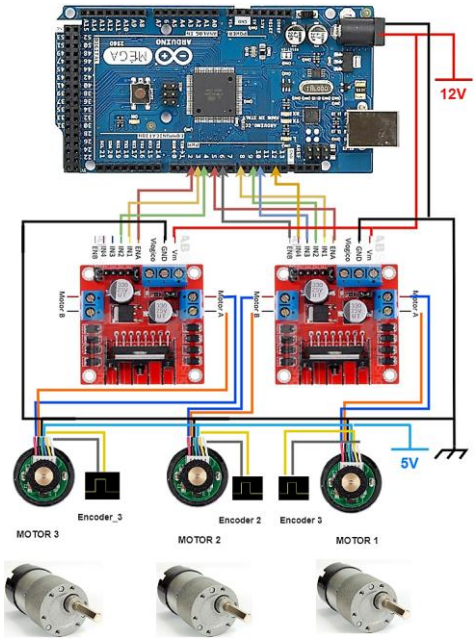


Imagen 29. Conexionado de los motores a la placa Arduino Mega.

4.7 Análisis de resultados y conclusiones

El test destinado a comprobar el correcto funcionamiento de los métodos implementados para el control de los motores ha consistido en la implementación de un código pruebas. El procedimiento seguido para la correcta realización de las pruebas, se ha resumido en los siguientes puntos:

1. El robot comienza a realizar todos los movimientos definidos durante un tiempo igual para cada uno de estos, de modo que, en teoría, una vez finalizada toda la serie, el robot debería de acabar en el mismo sitio donde empezó. Hay que aclarar que los movimientos deben ser complementarios entre ellos, es decir, si primero se va hacia adelante, después se marcha hacia atrás durante el mismo tiempo, y lo mismo para todos los demás.
2. Una vez recorrida la secuencia de movimientos, se hace parar el robot durante 3 segundos y comienza de nuevo el bucle.
3. Se repite el bucle varias veces (20 veces).
4. Una vez finalizados todos los bucles, se comprueba la correcta ejecución de los movimientos y se mide la desviación en distancia, sobre la posición de partida inicial, donde se ha parado el robot.

Movimiento	(t) en ms
goAhead(t);	1000
goBack(t);	1000
turnLeft(t);	2000
turnRight(t);	2000
goAheadRight(t);	1000
goAheadLeft(t);	1000
goBackRight(t);	1000
goBackLeft(t);	1000
goStop(t);	3000

Tabla 9. Secuencia de movimientos del test.

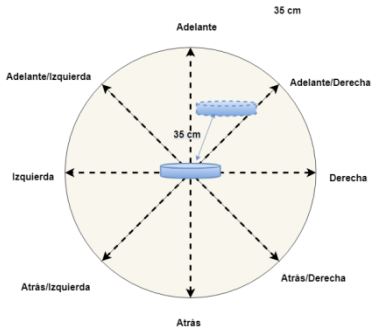


Imagen 30. Desviación sobre la posición inicial del robot.

En la figura (30) se han representado la posición inicial de salida del robot y la de finalización (a líneas discontinua) tras repetirse veinte veces seguidas la serie de movimientos del test (tabla 9), donde se registró una desviación sobre la posición inicial de 35 cm (hay que indicar que la prueba se realizó sobre una mesa de oficina). Esta discrepancia se debe tanto al tipo especial de construcción de las ruedas omnidireccionales (diseñadas para deslizarse en cualquier sentido) como a la diferencia de revoluciones que pueda existir entre los tres motores, además de otros factores como las características del piso donde se han realizado las pruebas (no es lo mismo realizar la prueba en un suelo de parquet que en un suelo constituido por losetas con llagas). En todo caso, para alcanzar los objetivos principales de este proyecto no se requiere de una gran precisión por lo que no se incidirá más sobre este tema, dando por aceptables los resultados alcanzados.

5. Sistema evasor de obstáculos (Módulo 2)

5.1 Sistema de detección por infrarrojos

A la hora de escoger un sistema de detección de obstáculos se analizó, en principio, la posibilidad de utilizar los ultrasonidos como sistema de detección, pero una vez examidadas las posibilidades de los infrarrojos y en concreto las de un determinado tipo de dispositivo, se decidió descartar aquella posibilidad y centrar el trabajo del diseño en un tipo específico de sensor de infrarrojos conocido como SHARP.

5.1.2 Estudio de los requerimientos del sistema

Se parte de la premisa de que el sistema debe poder detectar un objeto a una distancia aproximada de unos 20 cm e incluso menor. Al mismo tiempo el área a barrer será lo más amplia posible, dentro de unos límites útiles. En cualquier caso se ha de procurar usar pocos sensores con el fín de simplicar el sistema.

Los requesitos a cubrir por el sistema serán los siguientes:

- Poder detectar obstáculos habituales en cualquier casa, dispuestos a rás de suelo y en distintas posiciones en relación a la trayectoria del robot.
- La distancia de detección debe estar comprendida sobre los 20 cm.
- El área de detección debe ser lo más amplia posible, dentro de unos límites útiles.
- La disposición de los sensores debe estar localizado en la parte delantera del robot para poder establecer facilmente el desplazamiento que debe realizar en caso de detectar un obstáculo.
- Mínimo número de sensores para conseguir los objetivos.
- Disposición fija en el chásis del robot.
- Elección del componente adecuado entre el material disponible.

5.1.3 Hardware utilizado (elección del sensor)

Para este proyecto se ha elegido el sensor de infrarrojos Sharp GP2YA21. Este sensor tiene unas caracteristicas de detección bastante buenas además de resultar muy económico. Se trata de un tipo de sensor óptico compuesto de un emisor infrarrojo y un receptor, capaz de medir la distancia entre él y un objeto usando triangulación, midiendo uno de los ángulos que forman el triángulo emisor-objeto-receptor. El método de triangulación se basa en que el receptor PSE (Position Sensitive Detector) es capaz de detectar el punto de incidencia de la luz rebotada del objeto que depende del ángulo y de la distancia del mismo (imagen 31).

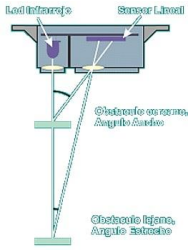


Imagen 31. Modo de detección de obstáculosn utilizado por el sensor Sharp.

La elección de este modelo, en concreto, se ha basado en que tiene un rango de detección que vá desde los 10 hasta los 80 cm, lo cual lo hace idonea para los requerimientos del sistema que se quiere implementar. En la siguiente figura (32) se presenta una imagen del sensor, rango de detección y una tabla con las características.

SHARP	MODELO	RANGO
	GP2Y0A21	10 a 80cm

■Electro-optical Characteristics (Ta=25℃,Vcc=5V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Average supply current	I_{CC}	L=80cm (Note 1)	—	30	40	mA
Distance measuring	ΔL	(Note 1)	10	—	80	cm
Output voltage	V_O	L=80cm (Note 1)	0.25	0.4	0.55	V
Output voltage differential	ΔV_O	Output voltage differece between L=10cm and L=80cm (Note 1)	1.65	1.9	2.15	V

* L : Distance to reflective object

Note 1 : Using reflective object : White paper (Made by Kodak Co., Ltd. gray cards R-27・white face, reflectance; 90%)

Imagen 32. Sensor Sharp, modelo y rango de detección.

5.1.4 Número de sensores y su distribución en el chasis

El número de sensores a utilizar y la separación entre los mismos se han establecido en base a la forma del robot y a sus dimensiones físicas. Teniendo en cuenta que la forma final del robot es circular y que el diametro del mismo es de unos 27 cm, desde el principio se hizo evidente que con sólo dos sensores no sería suficiente ya que su ángulo de medición es estrecho, sobre unos 40º. Con este dato la longitud del arco que conforma el área, a una distancia de 20 cm por cada uno de los sensores, es de algo más de 15 cm. La determinación del número de

dispositivos a utilizar se ha llevado a cabo mediante el método de prueba-error ,utilizando en principio sólo dos sensores en base a la distancia de detección de 20 cm. Después se aumentó el número a tres y por último se utilizaron cuatro sensores. Los resultados obtenidos se han expuesto en la tabla (10) siguiente:

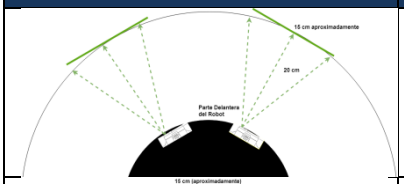
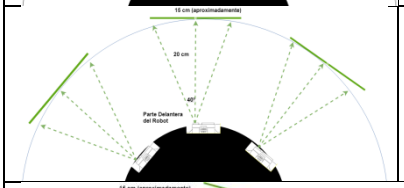
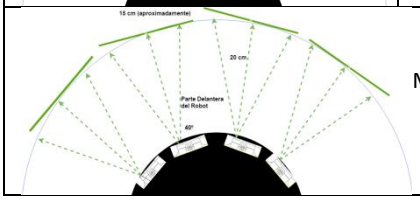
Número de sensores y su disposición en el chasis	Prob. de detección objetos centrales	Prob. de detección objetos centro-izq	Prob. de detección objetos centro-der	Prob. de deteccion objetos lateral izquierdo	Prob. de detección objetos lateral derecho
	Baja	muy alta	Muy alta	Baja	Baja
	Muy Alta	Alta	Alta	Muy Alta	Muy Alta
	Muy Alta	Muy Alta	Muy Alta	Muy Alta	Muy Alta

Tabla 10. Resultados en función del número de sensores utilizados.

La distribución se ha realizado como se observa en la figura (33) de abajo. Se han colocado los cuatro sensores en el arco definido por un ángulo de 90 grados y un radio de 12 cm. De este modo, a una distancia de 20 cm del perimétro del robot, la longitud del arco que encierra la proyección de este ángulo es de unos 52 cm:

$$radio\ de\ la\ circunferencia\ del\ robot = 13{,}5\ cm$$

$$radio\ total\ del\ área\ a\ cubrir = 20 + 13{,}5\ cm = 33{,}5\ cm$$

$$longitud\ del\ arco\ correspondiente\ a\ 90° = \frac{33{,}5 \cdot 2\pi}{4} \approx 52{,}62\ cm$$

La configuración final ha consistido en disponer 4 sensores dirigidos hacia la parte establecida como delantera, evitando la interferencia mútua entre los mismos, de modo que cubran tanto los laterales como el frontal del robot. Se procedió, además, a alejar los sensores un par de centrimetros de la parte delantera del robot para poder conseguir una disminución de la distancia de detección. Con esa disposición de los sensores se ha comprobado que el área de detección real que cubren es de unos 100º que, a 20 cm de distancia, equivale a unos 60 cm de arco. En conclusión, con la disposición elegida se ha logrado cubrir un angulo de medición de algo más de 100º alrededor del centro de la parte delantera del robot.

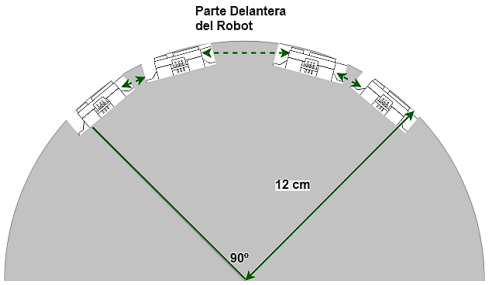


Imagen 33. Distribución de los sensores Sharp.

5.1.5 Elaboración del programa de control

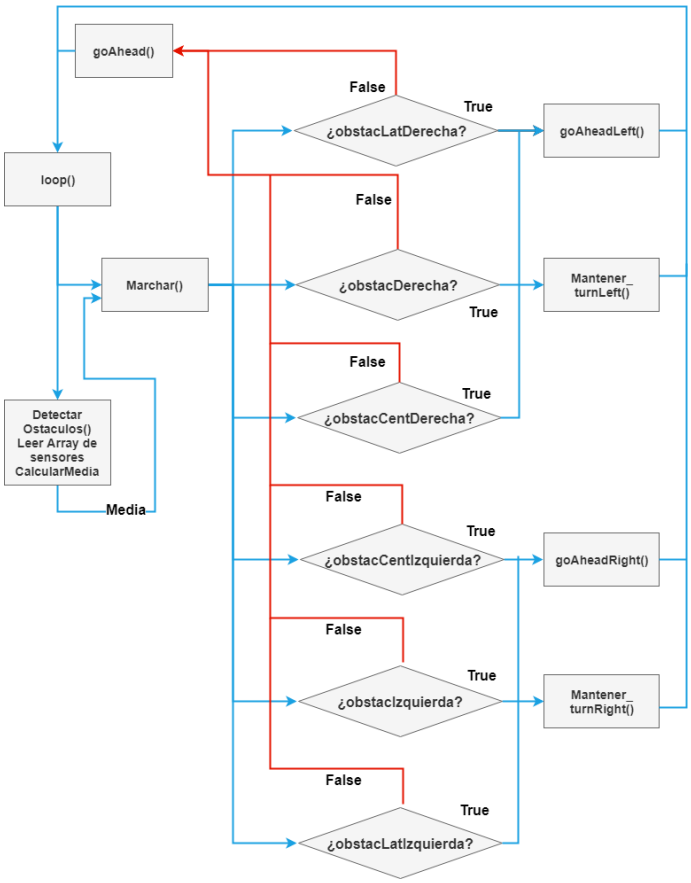


Imagen 34. Diagrama de flujo del sistema de detección mediante sensores Sharp.

En la figura (34), se representa el diagrama de flujo del sistema compuesto por los cuatro sensores Sharp, que servirá como base para explicar el funcionamiento del programa de control.

La detección de obstáculos se va a realizar utilizando una función principal que se ha llamado “detectarObstaculo()” que se encarga de guardar los datos en las variables asignadas a cada uno de los sensores (lectura, lectura2, lectura3, lectura4), datos medidos por otra función denominada “leersensor()”. Para establecer si se ha detectado un obstáculo se han creado distintas variables booleanas como “obstacIzquierda”, “ obstacDerecha”, obstacCentroIzq”… inicializadas como “false”. El método, en función del valor de los datos entregados por los sensores, establecerá si hay detección.

Un sensor habrá detectado un obstáculo si el valor medido por el mismo es mayor que 260. Este valor guardado en las variables “lectura..” no es un valor de distancia, sino una transformación a niveles lógicos (o escalones binarios) mediante un conversor de 10 bits, del voltaje entregado por la salida del sensor al pin análogico. La conversión se realiza con una sencilla fórmula teniendo en cuenta que la referencia serán los 5 voltios con los que se alimenta el sensor Sharp:

$$Valor\ digital = \frac{V_o}{5} \cdot 1023$$

Donde V_o es el voltaje de la salida del sensor.

La elección de esta magnitud de 260 se debe a que es equivalente, aproximadamente, a una distancia de algo más de 20 cm. A mayor cercania con el objeto detectado, mayor valor de tensión. Se ha obviado la transformación de esta magnitud a distancia porque de este modo se ahorra código. En la figura (35) puede verse la relación existente entre distancia y tensión registrada en la salida del sensor, representado en forma de gráfica.

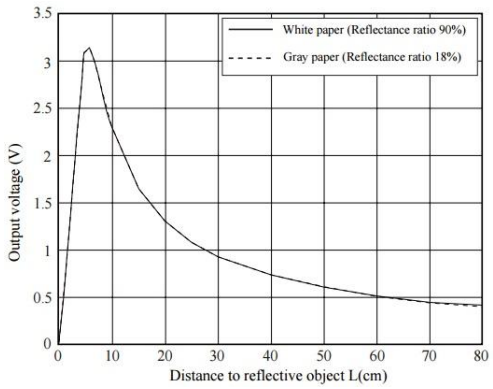


Imagen 35. Relación entre distancia y voltaje en el sensor SHARP.

Por otra parte, para disminuir el ruido de las mediciones se ha creado una función (calcular media) que realiza la media de 5 mediciones consecutivas de la señal de cada uno de los cuatro sensores entregadas por el método “leersensor()” y las guarda en una variable que será el valor utilizado en la función “Marchar()”.

Para saber el movimiento que el robot tiene que realizar, una vez detectado un obstaculo, se ha creado la función “void Marchar()”. En esta función se decide el movimiento que tiene que desarrollar el robot dependiendo del tipo de obstaculo detectado. En el caso de que no se haya encontrado ningún impedimento, porque todos los casos de obstáculos sean “false”, el dispositivo avanzará (“goAhead”). Por el contrario, si se ha detectado algún tipo de obstáculo este será “true” y el robot realizará el movimiento establecido para cada caso en concreto.

Hay situaciones donde puede suceder que todos, o casi todos, los sensores detectan obstáculos y esto provoque que el robot entre en una especie de bucle donde no puede establecer hacia que dirección debe desplazarse (gira hacia un lado y seguidamente hacia el otro sin ser capaz de salir de este bucle). Para evitar esta situación se han creado dos funciones (“Mantener_turnRight()” y “Mantener_turnLeft()”), que integran las funciones “turnLeft()” y “turnRight()”, que permiten salir del bucle antes mencionado. En este método se ejecutan el giro a la izquierda y el giro a la derecha, dependiendo de cual de las dos funciones los llamen, y después se llama al método “detectarObstaculo()” para establecer si las dos variables booleanas, “obstaculoDerecha” y “obstaculoIzquierda“, siguen siendo “true” o “false”.

5.1.6 Implementación práctica, montaje y conexionado

Como ya se explicó en el punto 5.1.4 la distribución en el chasis tendrá la forma que se muestra en la figura de la derecha.

Por lo que respecta al conexionado de los elementos, se ha realizado tal y como puede verse en la imagen (36), donde puede observarse que los cables de alimentación del sensor (rojo y negro) se conectan a un voltaje de 5 voltios y el otro cable (amarillo) es el encargado de enviar la señal análogica al pin asignado como entrada en el micro controlador. Los pines analógicos asigados como entradas, para cada uno de los 4 sensores, son el A4, A5, A6 y A7, respectivamente.

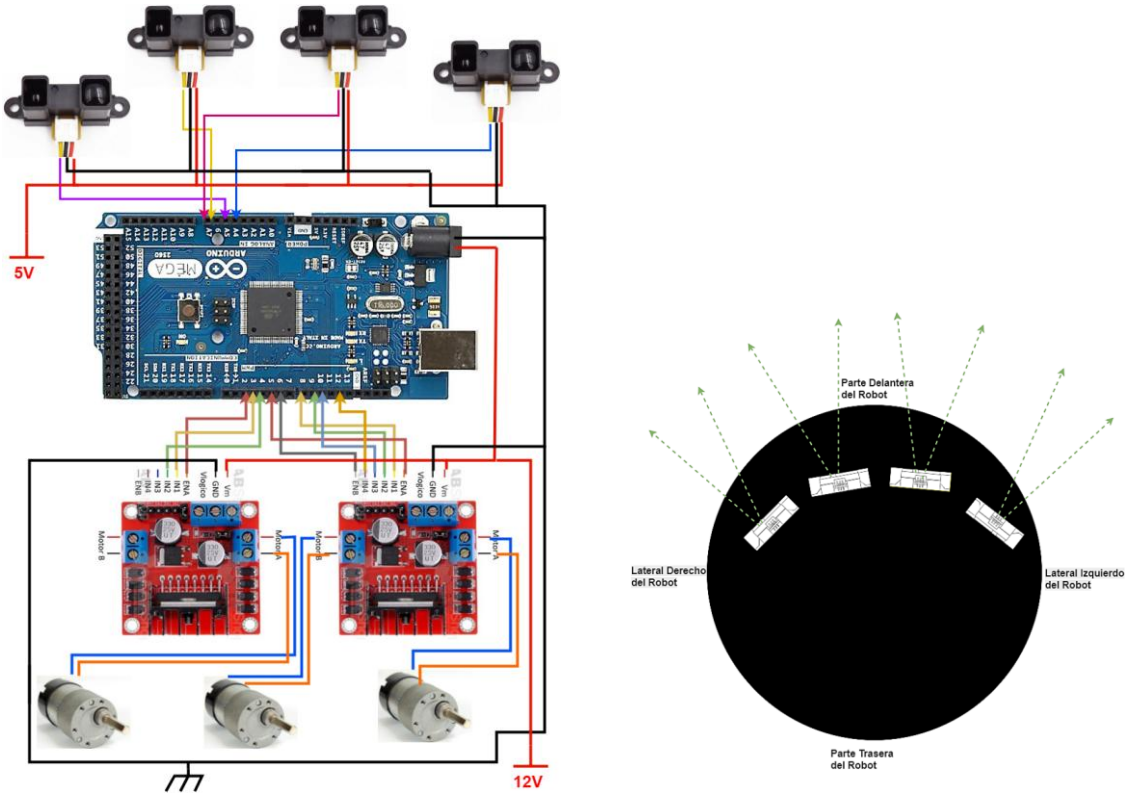


Imagen 36.Conexionado de los sensores Sharp a la placa Arduino y ubicación de los sensores en el chasis.

5.1.7 Análisis de resultados y conclusiones

Los resultados obtenidos han constatado que este sistema evasor de obstáculos, mediante sensores de infrarrojos, cumple con los requerimientos exigidos.

Para comprobar el correcto funcionamiento del sensor SHARP se ha realizado, en primer lugar, un programa de prueba donde se ha testado la respuesta del sensor imprimiendo por pantalla el valor analógico medido, así como la distancia correspondiente a esos valores. Se ha comprobado la validez de los datos colocando una regla sobre la horizontal y se ha ido variando la distancia de un objeto a lo largo de la longitud de la misma. De este modo se ha realizado una comparación entre los datos del datasheet del sensor, la lectura ofrecida por el plotter de Arduino, la medición mediante un polimetro de la tensión analógica a la salida del sensor y la conversión del voltaje a valores digitales. En la tabla (11) puede observarse la equivalencia entre la distancia y el valor leido por el sensor para un objeto de 24x16 cm, de color oscuro.

Distancia (cm)	80	70	60	50	40	30	20	10
Voltaje a la salida del Sharp (Voltios)	0,46	0,54	0,58	0,63	0,78	0,96	1,37	2,50
Valor Digital Representado en el plotter de Arduino	96	112	117	129	160	198	284	522

Tabla 11. Equivalencia entre distancia y valor a la salida del sensor Sharp.

Seguidamente se ha evaluado la respuesta del sensor ante objetos de distinto tamaño, forma, color y textura, a una distancia fija de unos 20 cm que es el límite establecido para detectar obstáculos. Para simplificar se han representado las gráficas de cuatro objetos diferentes entres sí, en lo que respecta al color, tamaño y forma.

La figura (37) corresponde a un objeto 24 x 16 cm, de color claro y superficie lisa (un libro) que se ha colocado a una distancia de 22 cm del sensor. En este caso se observa que la reflexión obtenida oscila sobre los 260 niveles, siendo la diferencia entre los máximos y mínimos (picos) poco significativa. Sobre este aspecto, podría “aplanarse” la respuesta aumentando el número de muestras utilizadas para hacer la media, aunque se considera bastante aceptable el resultado obtenido. En la siguiente imagen (38), el objeto tiene una superficie de 24 cm de altura y 4 cm de anchura (el libro puesto de perfil), a la misma distancia, siendo la respuesta similar que en el caso anterior.

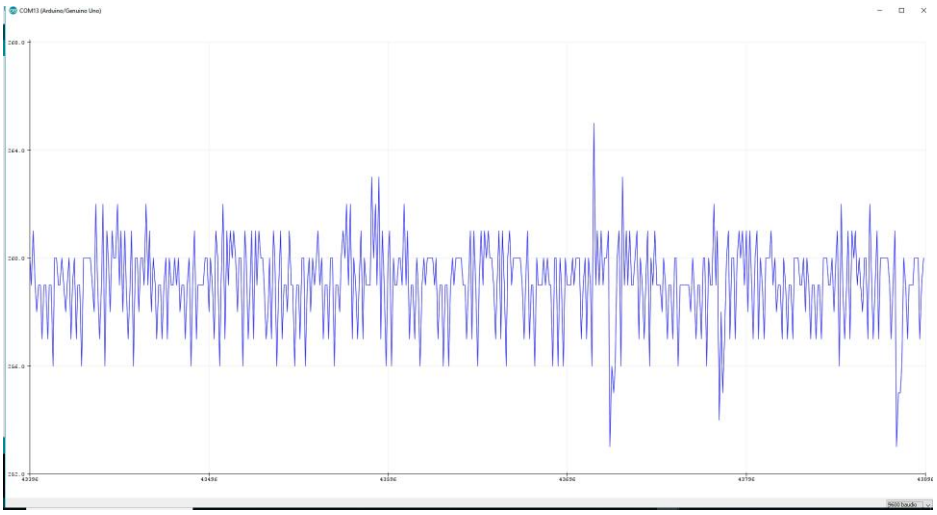


Imagen 37. Lectura obtenida de un objeto de 24 cm de altura y 16 cm de anchura a una distancia de 22 cm.

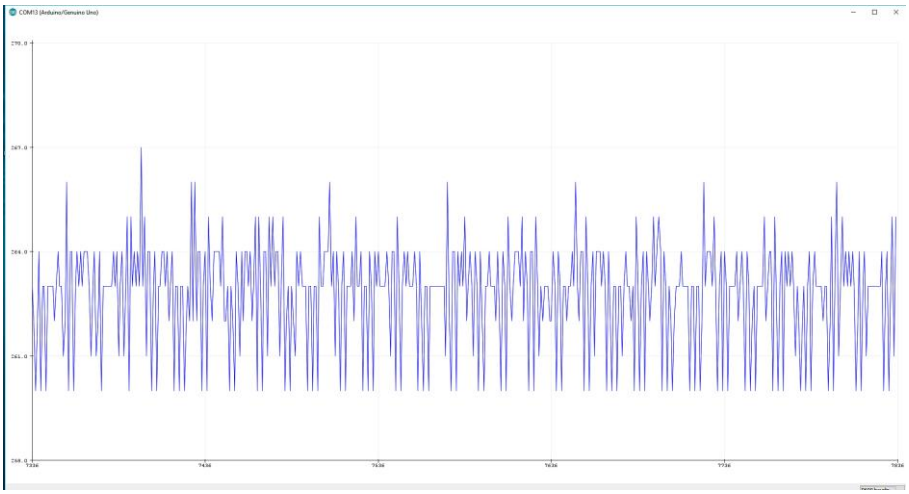


Imagen 38. Lectura obtenida de un objeto de 24 cm de altura y 4 cm de anchura a una distancia de 22 cm.

La imagen (39) corresponde al nivel alcanzado cuando el primer objeto (24x16 cm) se encuentra a 75 cm de distancia. El nivel medido se ha reducido hasta 104, aproximandamente, y la diferencia entre los máximos y mínimos (picos) también han aumentado con respecto a los dos casos anteriores.

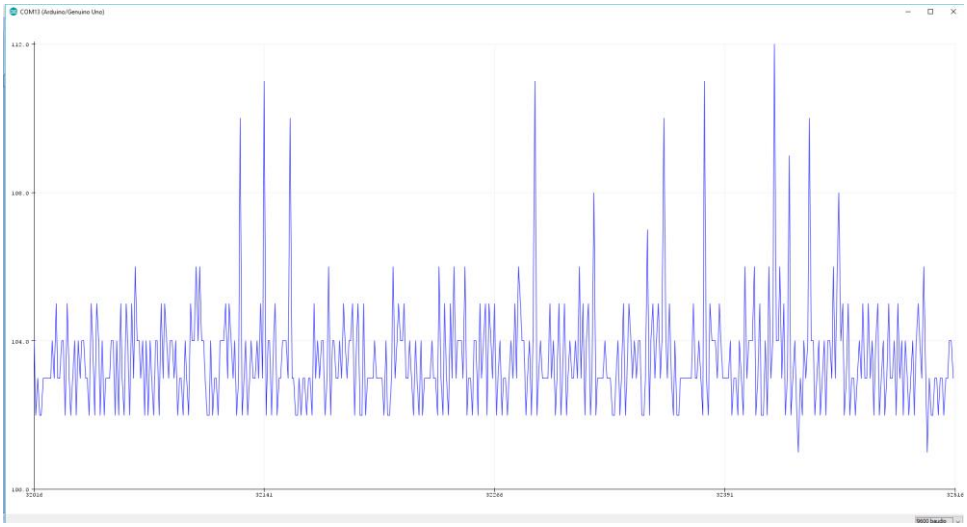


Imagen 39. Lectura obtenida de un objeto de 24 cm de altura y 16 cm de anchura a una distancia de 75 cm.

Para la siguiente prueba, un caso mucho más extremo, se ha elegido un objeto con unas medidas donde la superficie enfrentada al sensor tiene menos de 1 mm de anchura y 30 cm de altura, concretamente se trata de una regla metálica puesta de canto a 20 cm de distancia del sensor. Como puede verse el sensor sigue detectando el objeto aunque con bastante menos fiabilidad que en los casos anteriores, con máximos y mínimos mucho mas evidentes, como era de esperar (imagen 40).

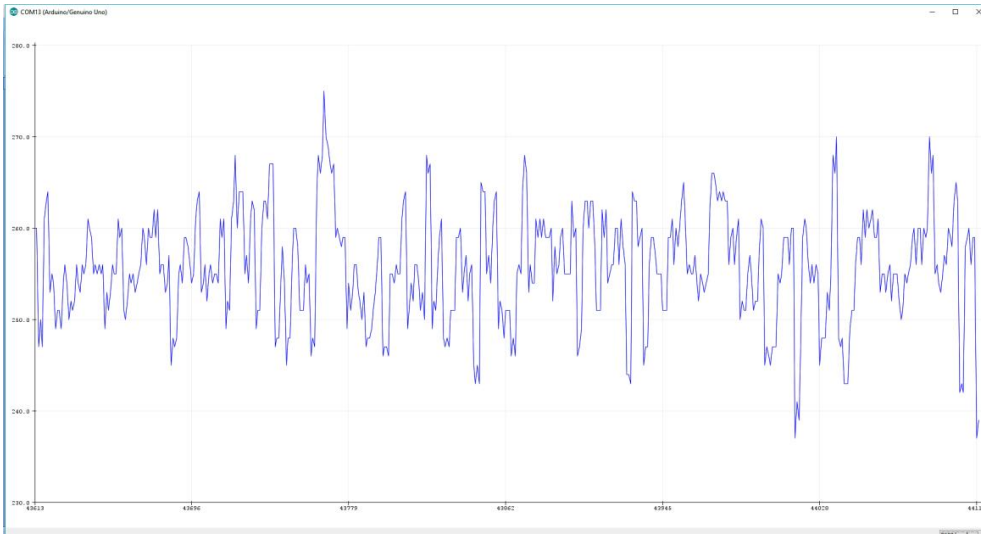


Imagen 40. Niveles obtenidos con una regla de 30 cm de altura y menos de 1 mm de anchura a una distancia de 20 cm, puesta de perfil.

En la última gráfica (figura 41) el objeto enfrentado al sensor, a 20 cm de distancia, es un cable coaxial negro de aproximadamente 0,5 de diametro. En este caso el sensor arroja valores mucho más bajos y con niveles máximos y mínimos, sobre el punto de oscilación, mucho mayores que en los tres casos anteriores.

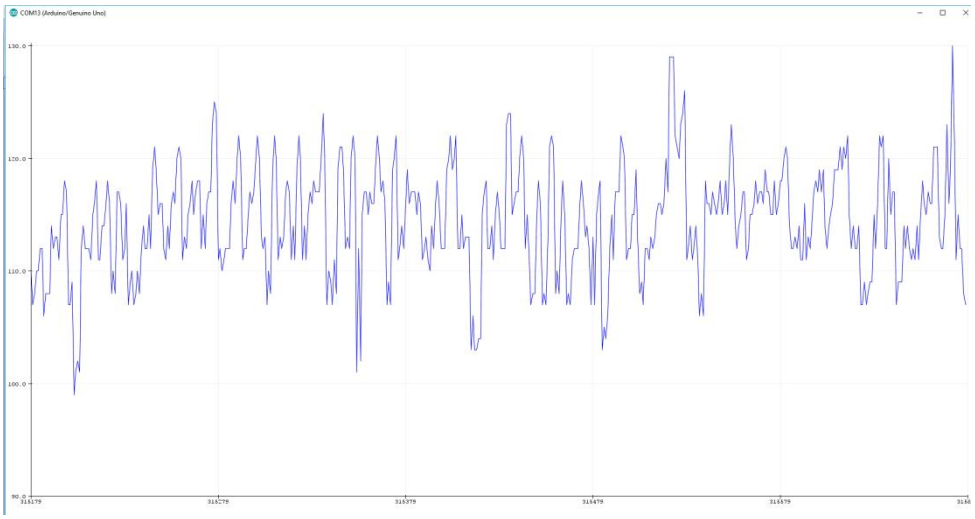


Imagen 41. Niveles obtenidos a 20 cm de distancia con un cable coaxial negro de 0,5 cm de diametro como obstáculo.

En conclusión, se puede afirmar que la diferencia de tamaño de los objetos no afecta sustancialmente en las medidas obtenidos, salvo en casos muy extremos como el de la regla puesta de perfil. Por el contrario, sí que es más determinante en los resultados obtenidos el tipo de rugosidad de la superficie del obstáculo y el color del mismo. El error será mucho mayor para objetos poco reflectantes y colores oscuros que para objetos de color claro y de superficie totalmente lisa.

5.1.8 Análisis del sistema en su conjunto

Era necesario evaluar la respuesta del sistema, en su conjunto, ante distintos obstáculos y comprobar así que los movimientos realizados por el robot cuando detecta distintos objetos, en distintas posiciones, son los correctos. En las siguientes figuras se ha intentado reflejar la reacción del sistema ante distintas condiciones y distribución de los obstáculos.

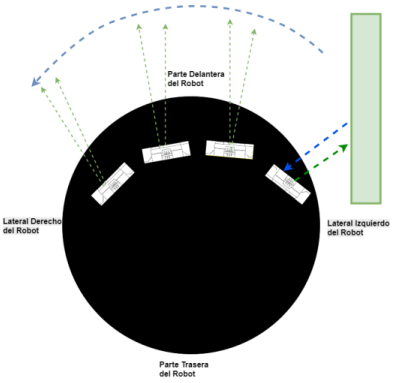


Imagen 42. Sensor Izquierdo detecta obstáculo.

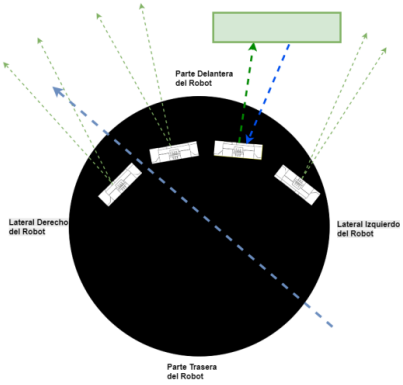


Imagen 43.Sensor centro-izquierda detecta obstáculo.

En la figura (42) el robot tiene el obstáculo en el lado izquierdo (mirándolo desde el frente), el movimiento efectuado es girar hacia su derecha. En la figura (43) el objeto es detectado por el sensor centro-izquierda, por lo que el movimiento de evasión realizado desplaza el robot hacia adelante y hacia la derecha.

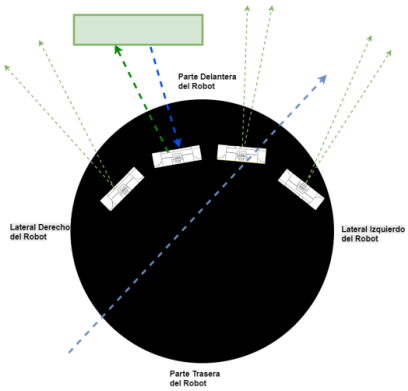


Imagen 44. Sensor centro-derecha detecta obstáculo.

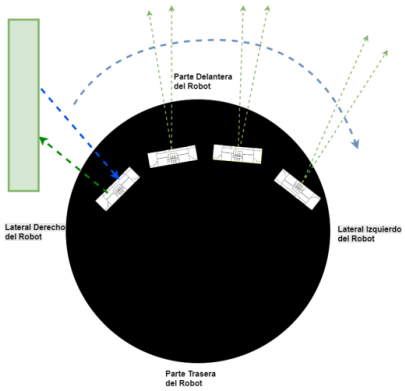


Imagen 45.Sensor derecho detecta obstáculo.

En la figura (44) el objeto es detectado por el sensor centro-derecha (mirándolo desde el frente) por lo que el robot se moverá hacia adelante y hacia la izquierda. En la figura (45) de la derecha, el objeto es detectado por el sensor de la derecha por lo el que robot girará hacia su izquierda.

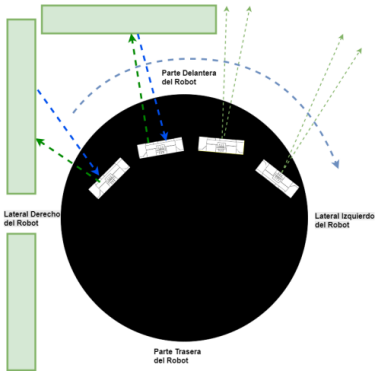


Imagen 46. Sensor der. Y centro-der. detectan.

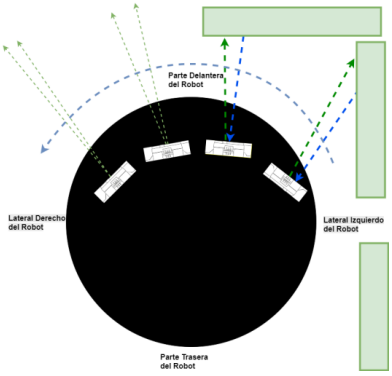


Imagen 47. Sensor izq. y centro-izq. detectan.

En la imagen (46) los sensores implicados en la detección de obstáculos son el sensor de la derecha y el sensor centro-derecha del robot, que hacen que el robot gire hacia su izquierda. En cambio, en la figura (47), los sensores implicados son el izquierdo y centro-izquierda, por lo que el robot girará hacia su derecha.

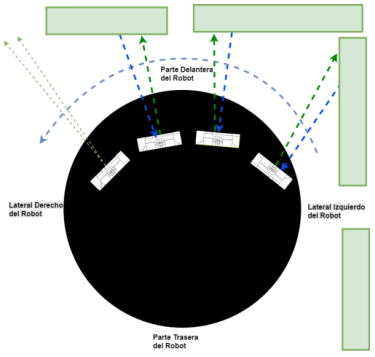


Imagen 48.Todos menos sensor derecha detectan.

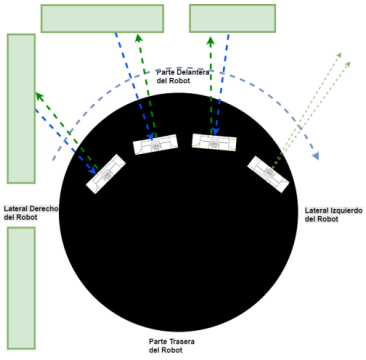


Imagen 49. Todos menos sensor izquierda detectan.

En la Imagen (48) hay tres sensores implicados (izquierda, centro-izquierda y centro-derecha) y el robot girará hacia su derecha. En la siguiente imagen (49), los sensores implicados (derecha, centro-derecha y centro-izquierda) condicionarán el giro hacia la izquierda.

El último caso a analizar (imagen 50), es aquel en que todos los sensores detectan obstáculos. En este caso se ha observado que las funciones implementadas a tal efecto (explicadas en el apartado 5.1.4), permiten al robot salir del bucle. En la figura se muestra mediante flechas de distinto color (rojo y azul) el giro que describirá el robot cuando decida cuál es la mejor opción.

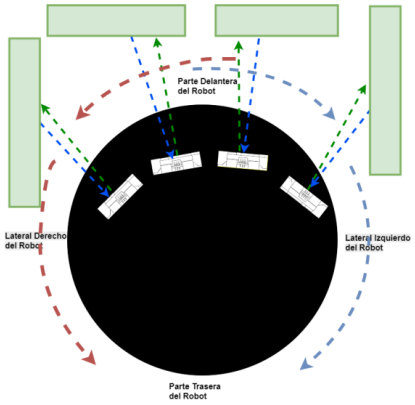


Imagen 50. Los cuatro sensores detectan obstáculos.

El robot consigue salir del bucle porque se hacen funcionar simultáneamente los métodos “Mantener_turnRight()” y “Manterner_turnLeft( )” cuando se detectan objetos a la derecha y a la izquierda, al mismo tiempo. El primer método hace girar el robot hacia la derecha y el segundo hacia la izquierda, mientras se sigue detectando obstáculos en ambos casos, lo que permite salir del bucle hacia la derecha o hacia la izquierda dependiendo de qué método se encuentre funcionando en ese preciso instante.

void Mantener_turnRight()	**void Mantener_turnLeft()**

{
while ((obstacDerecha == true) || (obstacIzquierda == true))
{
turnRight();
detectarObstaculo();
}
}
	{
	while ((obstacDerecha == true) || (obstacIzquierda == true))
	{
	turnLeft();
	detectarObstaculo();
	}
	}

Hay que tener en cuenta que los resultados obtenidos dependen, en gran medida, de la ubicación de los sensores, de la distancia entre los mismos y del límite de detección establecido, además del software utilizado para su control. El límite (20 cm) dificultará que el dispositivo pueda entrar en ciertos sitios donde la anchura total sea inferior al diámetro del chasis más la suma del doble del límite de detección establecido (27+20=47 cm), como puede observarse en la imagen (51). En las pruebas realizadas se rebajó este límite, retrasando la posición de los sensores con respecto al borde del chasis y de este modo se ganaron unos 10 cm. De este modo, se ha podido constatar que el robot pasaba sin dificultad en lugares con una anchura de paso igual o superior a 40 cm, aunque lo puede hacer a través de pasos más angostos, dependiendo del ángulo en que se encuentren los objetos.

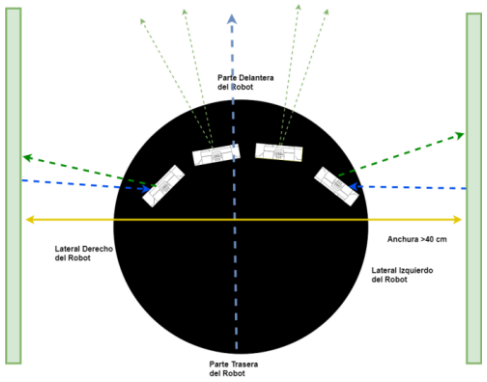


Imagen 51. Anchura mínima de paso, según la ubicación de los sensores con respecto al borde del chasis.

5.2 Sistema de detección por infrarrojos de la falta de piso bajo el chasis

Para conseguir que el robot evite caer por unas escaleras o cualquier otra circunstancia donde no exista suelo bajo el chasis, se ha diseñado un sistema de detección basado en sensores infrarrojos.

5.2.1 Estudio de los requerimientos del sistema

Los requesitos que debe cumplir el sistema son:

- Detectar la falta de suelo.
- Repuesta rápida y prioritaria.
- El sistema debe funcionar tanto si el robot se encuentra en modo de control manual como en modo evasor de obstáculos.

Para cumplir el requisito de prioridad se utilizarán “delays” en el código a realizar y de este modo las acciones ejecutadas tendrán preferencia durante el tiempo de duración de las mismas.

El tipo de sensores a utilizar será el mismo que se emplea en los dispositivos seguidores de línea (imagen 51). Su funcionamiento se basa en que la falta de reflexión del haz de infrarrojos provoca que la salida digital del sensor pase de estado bajo a alto,haciendo que el robot ejecute los movimientos pertinentes con tal de alejarlo de esa situación.

En la imagen (52), abajo a la izquierda, se ha representado la situación de detección de suelo. La flecha verde representa la emisión del dispositivo y la flecha azul la señal rebotada que es detectada por el sensor. En la figura de abajo, en cambio, el sensor no recoge la señal rebotada y por lo tanto pasará a estado alto provocando el retroceso del robot durante un tiempo y posteriormente el giro a derecha o izquierda, dependiendo de cual de los dos sensores delanteros haya sido el que ha cambiado de estado.

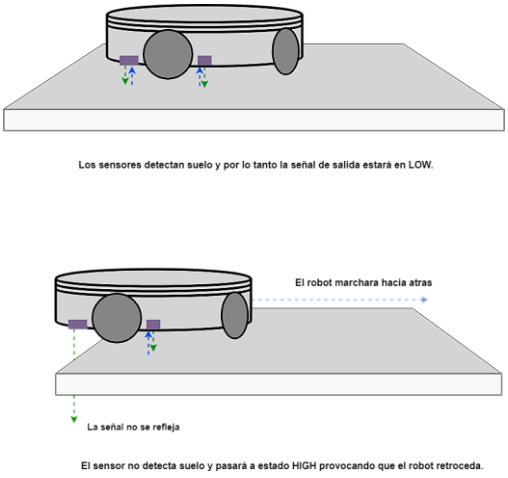


Imagen 52. Detección de falta de suelo.

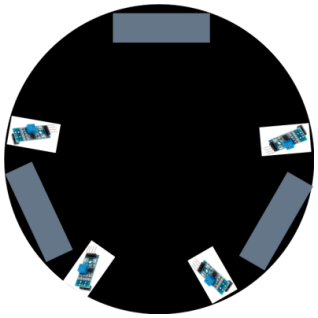


Imagen 53. Distribución de los sensores.

Se ha decidido instalar 4 sensores en la parte baja del chasis (plancha circular de madera) en una primera implementación y así poder comprobar el buén funcionamiento del sistema cuya distribución puede verse en la figura (53). La amplicación del número de estos dispositivos se dejará para futuras mejoras del proyecto.

5.2.2 Elaboración del programa de control

El método desarrollado tiene el nombre de “void Detectar_suelo()” que funciona en bucle permanente (loop). Esta función llama a otra encargada de leer el estado de los 4 sensores “Leer_Sensores()” donde se ha creado un array de 4 elementos (“sensor[0], sensor[1], sensor[2], sensor[3]”) que guardan el estado en el que se encuentran los mismos (“Dsensor_...”). Cada sensor estará en estado alto (HIGH) cuando se detecte la falta de suelo o estado bajo (LOW) en caso contrario.

Dentro del método “Detectar_suelo()” se utiliza el condicional “if”, evaluando distintos casos, para decidir que movimientos debe realizar el motor para escapar de la situación de falta de firme. Se han utilizado “delays” para mantener el movimiento de los motores durante el tiempo suficiente, interrumpiendo cualquier otro proceso que pudiera estar ejecutandose. También se ha implementado un caso “if” específico para cuando todos los sensores estan en estado alto, lo que implicaria que el robot, por ejemplo, ha sido levantado del suelo, que hace que los motores se detengan llamando a “stopRobot()”. Tras finalizar cualquiera de los movimientos establecidos, durante el tiempo de los delays, el robot pasará al estado stopRobot(), parandose los motores. En el diagrama de flujos de la imagen (54) se representa el funcionamiento del programa. En todo caso el diagrama corresponde a una simulación y las acciones a realizar podrán ser modificadas cuando se considere oportuno.

En el esquema del diagrama de flujos (imagen 54), para simplificar su comprensión, se ha creado una linea de flujo específica para cada caso, los cuales se han identificado con un color diferente con el fin de poder identificar facilmente las acciones que se realizan.

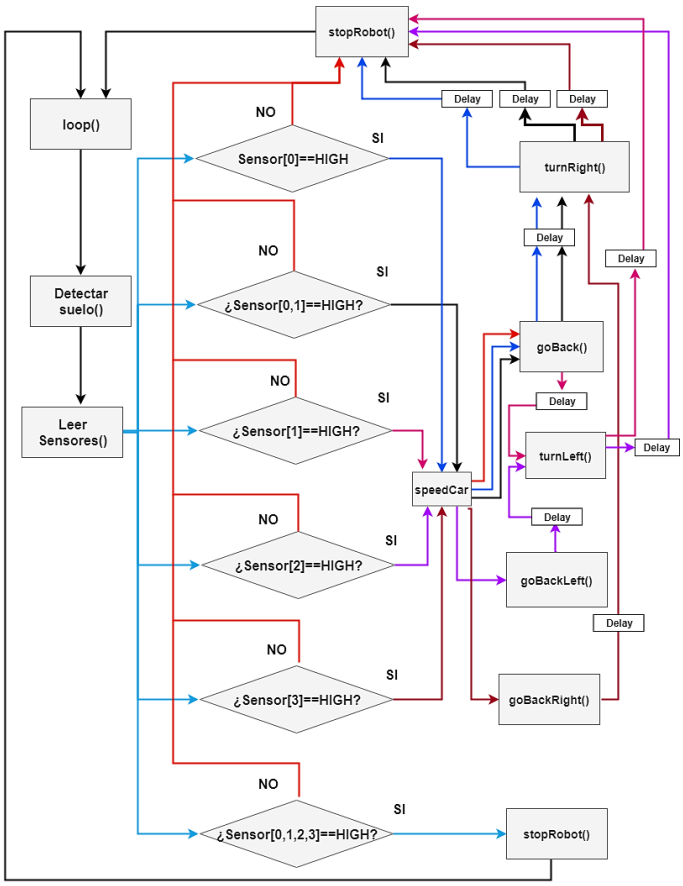


Imagen 54. Diagrama de flujos del programa para el sistema detector de falta de suelo

5.2.3 Montaje y conexionado

Imagen 55. Sensor utilizado.

Como ya se ha comentado, la colocación de los sensores en el sitio adecuado es fundamental para que cumplan su función eficazmente. A tal fin se han colocado dos sensores en la parte delantera, cerca de las ruedas, y otros dos justo detrás de estas. Con esta configuración, que no resulta suficiente, sí que se ha podido comprobar la eficacia del sistema en cuanto a la velocidad de respuesta de los sensores.

El conexionado no ofrece ninguna dificultad, ya que solo ha de conectarse la alimentación a las conexiones correspondientes y la salida digital del sensor conectarla al pin elegido como entrada en el micro controlador. La salida entregada por el sensor es digital (HIGH, LOW) con lo que en las entradas asignadas en el micro controlador (A8, A9, A10, A11) para cada uno de los sensores, respectivamente, se tendrán los niveles de tensión a 0 voltios ó 5 voltios, según sea el caso. La alimentación del dispositivo es de 5 voltios aplicada al cable rojo, siendo el negro la masa, como puede verse en la imagen (56).

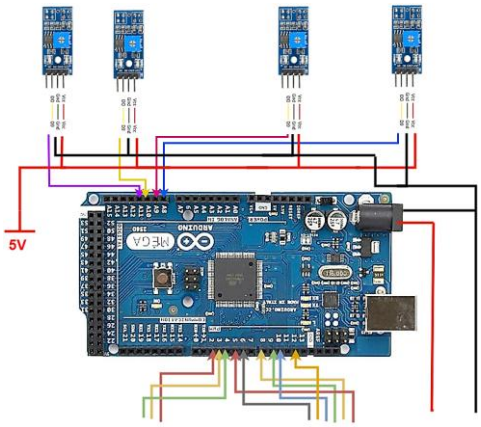


Imagen 56. Conexionado de los sensores seguidores de línea.

5.2.4 Análisis de resultados y conclusiones

En lo que respecta al funcionamiento se ha constatado que funciona con suficiente precisión y en todas las pruebas realizadas con el robot en ningún momento ha fallado la detección. Para testar el funcionamiento del sistema se ha realizado un número de pruebras elevada, donde se han puesto a prueba los sensores acercando el robot al borde de la mesa de modo repetitivo, sometiendo a cada uno de los 4 sensores a un mínimo de 100 pruebas. En la tabla (12) se indican los resultados obtenidos, mostrando el número de intentos, los sensores implicados y la respuesta obtenida.

	Sensor[0]	Sensor[1]	Sensor[2]	Sensor[3]
nº pruebas	100	100	100	100
Total fallos	0	0	0	0

Tabla 12. Efectividad del sistema detector de falta de suelo.

En cualquier caso no se descarta la posibilidad de modificar la ubicación de los sensores en el chásis y también aumentar un poco el diametro de la plataforma circular para poder elevar el número de dispositivos y mejorar su ubicación.

5.3 Sistema para la variación progresiva de la velocidad en función de la distancia de un obstáculo

Como complemento al sistema de detección de obstáculos se ha realizado un sistema de variación de velocidad con un sensor láser. De este modo, si el robot se encuentra en modo evasor de obstáculos, cuando los objetos se encuentren a muy poca distancia la velocidad se reducirá, aumentando la precisión de los movimientos.

5.3.1 Requerimientos

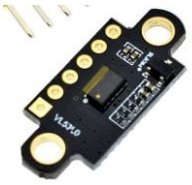
Los requerimientos exigidos a este sistema son:

- El sistema debe ser capaz de detectar la distancia a que se encuentra el robot de un obstáculo. En lo que respecta a la precisión de la medida, el sensor tiene que poder medir distancias desde unos pocos centimetros (4 cm, como mínimo) hasta algo más de 1 metro y hacerlo del modo más preciso posible.
- Debe poder adaptar la velocidad del robot en función de la distancia.
- Sólo funcionará cuando el robot se encuentre en el estado de auto-conducción o evasión de obstáculos.
- Por otra parte, este sistema también se podrá utililizar para complementar al sistema de detección de obstáculos mediante infrarrojos descrito en el apartado 5.2.

5.3.2 Sensor láser VL53L0X

La elección de este dispositivo se ha basado en que es capaz de medir distancias con una precisión de milimetros, lo cual lo hace ideal para este proposito.

Imagen 57. Sensor láser VL53L0X



Este sensor dispone de un emisor láser que cada cierto tiempo emite un haz de luz (pulso) que es reflejada al encontrar un objeto. El sensor se encarga de medir el tiempo entre la emisión y la emisión de la luz mediante la electrónica interna que incorpora. El haz de luz emitido es del tipo VCSEL (Vertical Cavity Surfac-Emiting Laser) que es totalmente invisible al ojo humano. Este sensor esta equipado con óptica con filtros anti-infrarrojo. Su angulo de medición (FOV) es relativamente estrecho: 25 grados, lo que supone que el area de medicón a 1 metro es de 0,44 m. Esto último puede ser una ventaja si lo que se quiere medir es la distancia justo en frente del sensor, por este motivo no es el mejor sistema de detección de obstáculos aunque puede ser muy útil para medir distancias con mayor precisión que el sensor Sharp.

Se puede hacer funcionar bajo dos modos básicos en función de la distancia: por una parte, el modo estándar que comprende entre entre 50 a 120 cm de alcance y un modo ampliado hasta los 200 cm. La presición dependerá del entorno, objetivo y modo de funcionamiento.

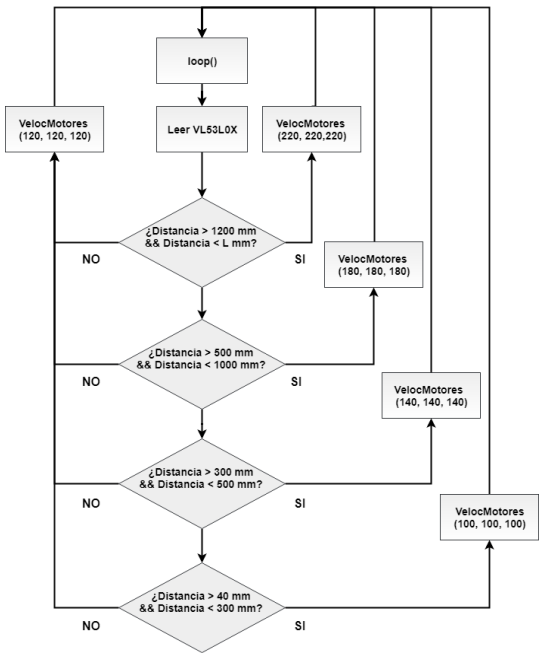


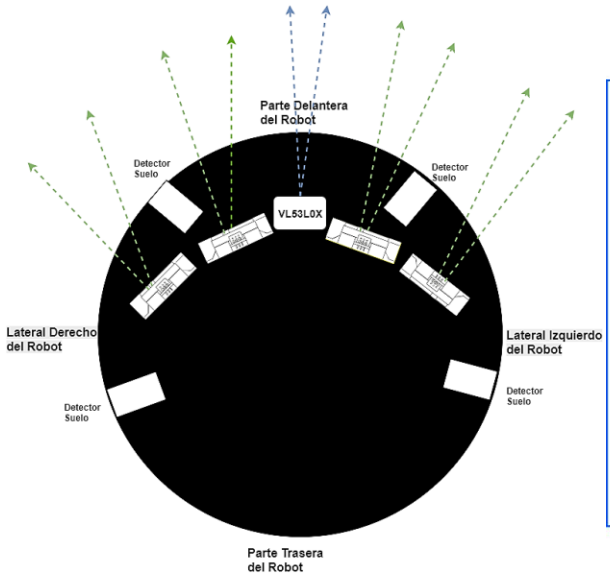
Imagen 58. Diagrama adaptacion velocidad por laser (derecha)

5.3.3 Elaboración del programa de control

Una vez vistos los requesitos exigidos al sistema, se ha procedido a la realización del código de control. El programa realizado ha sido una adaptación directa de uno de los ejemplos incluidos en la librería “Adafruit_VL530X.h”. Las pruebas de medición de distancia se hicieron utilizando uno de esos ejemplos y posteriormente se incluyó dentro de la función “void detectarObstaculo()” donde se ha limitado su función a medir distancias y adaptar la velocidad del robot a esas mediciones

Se ha realizado un sistema con una estructura compuesta por una serie de “if” condicionales que llaman a la función “VelocMotores()” que establece la velocidad de los motores en función de la distancia a la que se encuentra el obstáculo, tal y como puede verse en el diagrama de flujos de la figura (58).

5.3.4 Montaje y conexionado



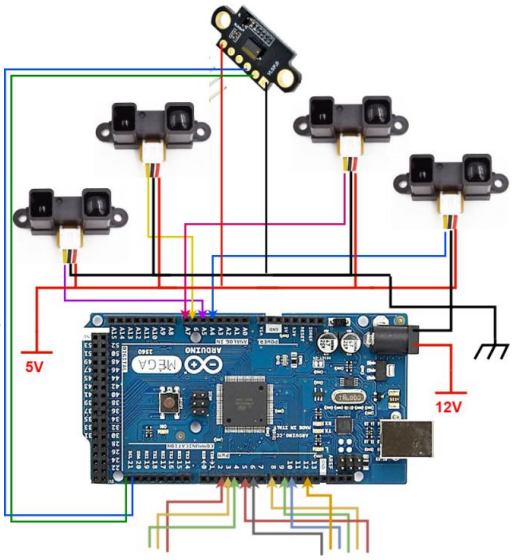


Imagen 59. Situación del sensor VL53L0X en el chasis.

Imagen 60. Conexionado del sensor VL53L0X.

El sensor se ha situado justo en el centro de la parte delantera, entre los sensores de infrarrojos. El conexionado se realiza a través del bus I2C, con lo que sólo hay que conectar la alimentación de 5V al módulo y los pines SDA y SCL (20 y 21) del Arduino Mega a los correspondientes del sensor. En la imagen (59) puede verse la localización aproximada del sensor, junto con todos los demás sensores que van a ser usados en un montaje inicial de prueba y su conexión en el Arduino Mega (imagen 60).

5.3.5 Análisis de resultados y conclusiones

La prueba realizada ha consistido en enfrentar un objeto a distintas distancias con el robot, que no tiene las ruedas apoyadas sobre el suelo y tiene fijado el movimiento de ir hacia adelante. Se fueron tomando lecturas de la variación experimentada en la velocidad de desplazamiento del robot y se compararon dichas variaciones con lo que se ha establecido en el programa de control realizado a tal efecto (figura 61).

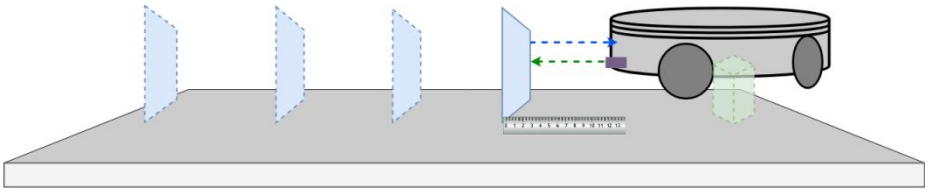


Imagen 61. Comprobación de la adaptación automática de la velocidad a la distancia en que se encuentra el objeto.

En la siguiente tabla (13) puede verse los resultados obtenidos sobre las variaciones de velocidad experimentadas en función de la distancia al objeto.

Distancia (cm)	5	20	35	45	60	90	110	120 <Dist<130	Dist >130 (no mide distancias)
Nivel PWM Motores (0 a 255)	100	100	140	140	180	180	180	220	120

Tabla 13. Variación de la velocidad en función de la distancia al objeto.

6. Sistema de control del robot vía *wifi* (Módulo 3)

6.1 Análisis de requerimientos

Para dirigir el robot se ha creado un sistema de control manual, a través de una APP para Android, que permite realizar las siguientes acciones:

- Conectar o desconectar el sistema, en su conjunto, mediante dos botones, uno de encendido y otro de apagado. Estos botones se deben quedar en el estado asignado con una única pulsación. Por otro lado se deben crear otros dos pares de botones (conexión y desconexión), del mismo tipo que en el caso anterior, para activar/desactivar una cámara IP y cambiar de estado de conducción manual al estado evasor de obstáculos, respectivamente.
- Para poder guiar el robot manualmente se deben de implementar unos pulsadores que controlaran el movimiento del robot mientras se mantengan pulsados, parándose cuando deje de ejercerse acción sobre los mismos.
- Se ha de poder establecer distintas velocidades de desplazamiento mediante un control deslizante.

Asimismo, la aplicación deberá cumplir los siguientes requisitos:

- Todas las acciones mencionadas deben poder ser realizadas a distancia, es decir, el método de control del robot se va a realizar mediante una señal de radio *wifi*.
- No se desarrollará ni construirá ningún tipo de mando a distancia (Hardware) específico para controlar el robot sino que se utilizará algún dispositivo que disponga de una conexión *wifi*, como un Smartphone o una Tableta.
- La aplicación de control deberá ser compatible con el sistema operativo Android, ya que los dispositivos móviles de los que dispone el autor funcionan todos bajo este sistema operativo.
- Para activar la alimentación del robot, en el chasis sólo existirá un interruptor de encendido de la alimentación que le permitirá estar en estado de espera hasta que se produzca la activación de los sistemas mediante el mando a distancia.
- Para realizar las acciones de control será necesario crear una APP que una vez instalada en el dispositivo móvil sirva para esos propósitos.

6.2 Realización de la aplicación (APP) para el control del robot

Para el control del robot se ha diseñado un mando distancia (APP para el sistema operativo Android) compuesto por una serie de botones de un solo toque para la activación/desactivación de relés en el chasis del robot y para activar el modo auto o evasor de obstáculos, de otros botones del tipo pulsadores para el control manual del robot y un control deslizante para regular la velocidad de desplazamiento.

La APP se ha realizado mediante MITT App Inventor que es un entorno de desarrollo de software desarrollado por el MIT (Massachusetts Institute of Technology) para la elaboración de aplicaciones destinadas al sistema operativo Android.

Desde su página web http://appinventor.mit.edu/explore/, puede entrarse a través de la pestaña "Create apps!" en la interface de creación de las aplicaciones, con sólo disponer de una cuenta de correo. Dispone de un set de herramientas bastante completo que permiten crear una gran diversidad de aplicaciones. Los trabajos realizados pueden guardarse en la nube, en el ordenador y también compartirlos.

El entorno de desarrollo tiene dos pantallas principales, la pantalla de diseño que sirve para crear los botones y otras funcionalidades que se quieran implementar y otra pantalla para definir las acciones que ejecutaran cuando se pulsen las teclas (Bloks).

Una vez creado los botones, en la pantalla de diseño, hay que "casarlos" con los bloques de creación del programa en la siguiente pantalla. En esencia, lo que se hace es adjudicar a cada botón una función (código) que se realiza mediante los bloques disponibles (sistema Scratch). En la figura (62) siguiente pueden verse la pantalla el aspecto final de la APP creada.

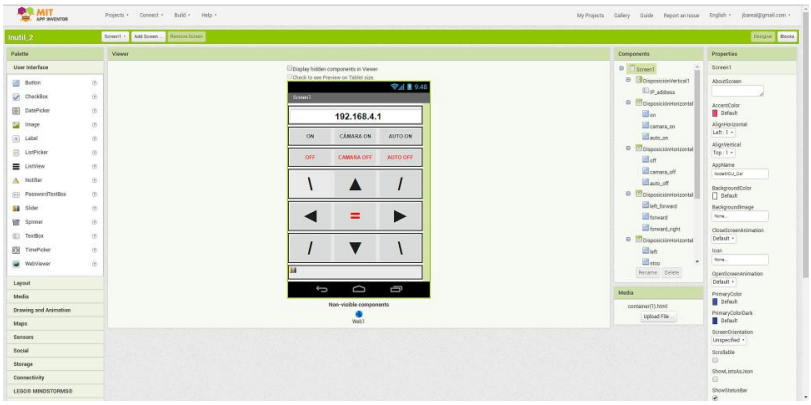


Imagen 62. Imagen de la APP para el control del robot.

La siguiente figura (63) muestra distintos conjuntos de bloques para distintas funciones que se han de realizar cuando se presionan las teclas de la APP:

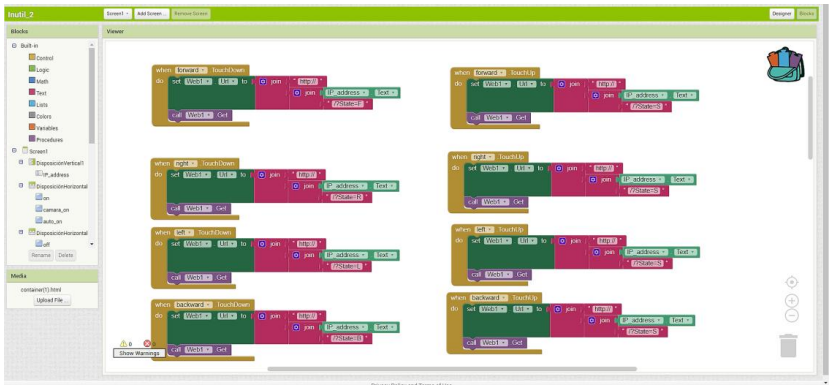


Imagen 63. Imagen de los bloques utilizados para la creación de la APP.

La implementación del código es muy sencilla e intuitiva gracias a la utilización de bloques. A modo de ejemplo, el grupo de bloques de arriba a la izquierda, y el de arriba a la derecha, sirven para enviar los comandos que hacen que el robot vaya hacia adelante y que se pare cuando se deje de presionar la tecla. Como puede verse en la imagen (63), esta función se realiza con el condicional “when” (cuando) y “do” (hacer) que implica que cuando se presiona la tecla “fordward”, que en el mando seria la flecha con dirección hacia arriba, el dispositivo emisor, (el Smartphone) conectado con la URL de la *wifi* utilizada, está enviando el comando “F” que hará que el robot se mueva hacia adelante. Como se trata de un pulsador, cuando se deje de presionar este (TouchUp) se enviará el comando “S” que en todos los casos implicará que el robot debe detenerse (stopRobot).

Una vez se ha finalizado la realización de la APP, hay que proceder a descargar el archivo generado con extensión .apk, e instalarlo en el dispositivo elegido (Smartphone o tableta con sistema operativo Android).

6.3 Programa a instalar en la placa nodeMCU

Para comprobar el correcto funcionamiento de la APP se debe instalar un software de control en la placa nodeMCU, que será la encargada de conectar mediante su módulo WiFi el Smartphone o tableta que se haya elegido como mando a distancia del robot. El programa a instalar en la placa nodeMCU es una adaptación de los ejemplos contenidos en las librerías del módulo WiFi ESP8266, las cuales han sido instaladas previamente en la IDE de Arduino.

El programa se ha realizado mediante la IDE Arduino ya que placa nodeMCU es completamente compatible con este entorno de desarrollo, utilizando las librerías específicas para el módulo Wifi como ya se ha indicado.

Librerías, variables y funciones	void setup()	void loop()

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
const char* host = “WiFi_Inutil2_NodeMCU";
const char* ssid = "WiFi_Inutil2";
ESP8266WebServer server(80);
 	
void HTTP_handleRoot(void) {
if (server.hasArg("State")) {
Serial.println(server.arg("State"));
}
server.send (200, "text/html", "");
}
	{
	Serial.begin(115200);
	WiFi.mode(WIFI_AP);
	WiFi.softAP(ssid);
	server.on ("/", HTTP_handleRoot);
	server.onNotFound (
	HTTP_handleRoot);
	server.begin();
	}
		{
		server.handleClient();
		delay(50);
		}

El programa, como puede verse arriba, consta de la declaración de las librerías, la declaración de constantes (en este caso el nombre del host) y el puerto a utilizar que habitualmente suele ser el puerto 80. Se implementa, también, una pequeña función que es la encargada de comprobar la conexión entre la placa y dispositivo, y establecer el envío de comandos una vez se haya comprobado que existe comunicación. El bloque central se encarga de inicializar el puerto de comunicaciones y el servidor. Por su parte, el método “handleClient()”, implementado en server, funciona en bucle dentro de loop() “refrescando” la conexión cada 50 ms.

A la hora de comprobar su correcto funcionamiento se ha procedido a cargar, primero, el programa en la placa nodeMCU mediante la IDE de Arduino, y en segundo lugar hay que conectar el teléfono móvil con el módulo *wifi* de la nodeMCU. Una vez conectados los dos dispositivos se comprueba si se reciben los datos enviados por la aplicación en la placa nodeMCU, abriendo el monitor serie de Arduino que debe estar sincronizado a la misma velocidad que se ha especificado en el programa cargado en la placa.

En la figura (64) pueden verse los resultados obtenidos: al pulsar una de las teclas en el Smartphone se visualiza un comando determinado y una vez se suelta la tecla, si el botón es del tipo “touch”, automáticamente se recibe el comando “S”. Cuando el botón pulsado no es un pulsador sino un interruptor, el comando recibido se mantiene (comando “A” por ejemplo) aunque deje de pulsarse. También se ha constatado el correcto funcionamiento del control deslizante que establece las distintas velocidades (los números 3, 4,...).

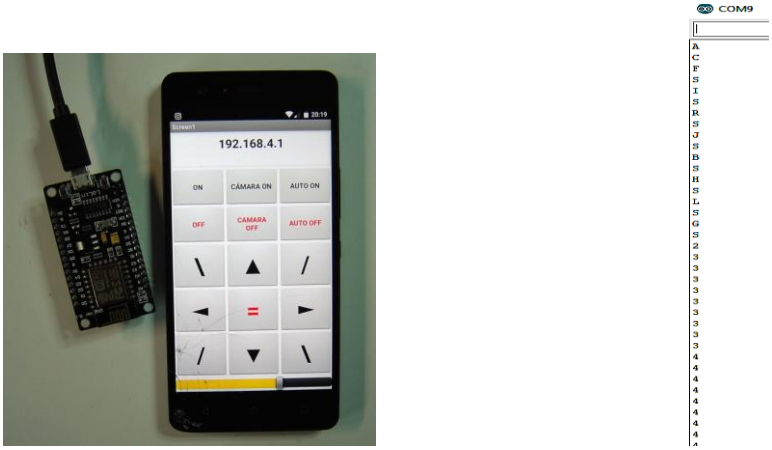


Imagen 64. Vista de la aplicación instalada en el teléfono (izquierda) y comandos recibidos en el puerto serie de la IDE (derecha).

6.4 Programa para el control WiFi a instalar en la placa Arduino Mega

Una vez cargados en la placa nodeMCU el programa y las librerías necesarias para poder enviar comandos mediante la APP, será necesario implementar unos métodos específicos en la placa Arduino Mega que permitan controlar al robot mediante los comandos recibidos por el módulo *wifi*.

6.4.1 Transferencia de datos entre las dos placas

Los datos recibidos en la nodeMCU, a través de su módulo ESP8266, deben ser transferidos a la placa Arduino Mega para poder transformar los comandos recibidos a acciones para controlar el robot. La transferencia de datos se ha realizado conectando los puertos serie de las dos placas (UART/USART). El protocolo de comunicaciones Serie, o Serial, consiste en transmitir la información bit a bit a través de unos puertos especiales denominados Rx (recepción) y Tx (transmisión) que están conectados internamente a un chip etiquetado como UART(Universal Asynchronous Reciver-Transmiter). La UART toma los bytes de datos y los convierte a una secuencia serie de bits que serán transferidos a través del puerto Tx. En la recepción, puerto Rx, la UART realizará el proceso inverso, es decir, transformará la secuencia entrante de bits en un bloque de bytes completo. El modo de conexionado de las dos placas puede verse en la figura (65).

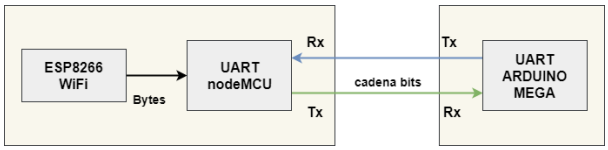


Imagen 65. Conexión de las UART de las dos placas a través de los puertos Tx/Rx.

6.4.2 Realización del código para el tratamiento de los datos recibidos

Los datos que llegan a través de la UART del Arduino deben servir para controlar las funciones del robot y para ello se debe crear unos métodos específicos. El diagrama de flujos del programa creado puede verse en la figura (66), el cual será explicado a continuación.

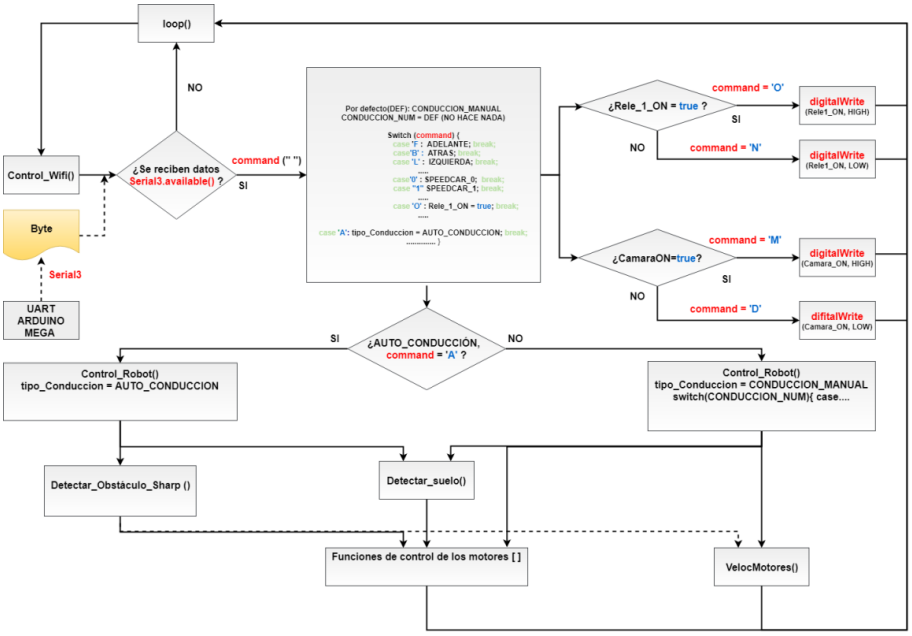


Imagen 66. Diagrama de flujo del programa del control vía WiFi instalado en el Arduino Mega.

En el esquema puede verse, arriba a la izquierda, un bloque que representa a la función “Control_Wifi()”, llamada por “loop ()”**.** Esta es la función principal que se encarga de comprobar la recepción de datos a través del puerto serie (Serial3) y de establecer una correspondencia entre los datos recibidos (comandos) y las funciones a realizar por el robot.

Antes de analizar el funcionamiento de esta función es necesario indicar que se han declarado dos modos de control del robot en la enumeración “enum DS” que por defecto se inicia como conducción manual (tipo_Conduccion = CONDUCCION_MANUAL), es decir, será necesario utilizar el mando para activar las distintas funciones implementadas en el robot. Por otra parte, los tipos de movimientos y los distintos grados de ajuste de la velocidad se han declarado mediante la enumeración “enum DN”, que por defecto se inicializa a DEF que no se asocia con ninguna función en concreto, por lo que el robot no hace nada al inicializarse.

tipo_Conduccion	CONDUCCION_NUM
enum DS { CONDUCCION_MANUAL, AUTO_CONDUCCION } tipo_Conduccion = CONDUCCION_MANUAL;	enum DN { ADELANTE, ATRAS, IZQUIERDA, DERECHA, ADELANTE_DERECHA, ADELANTE_IZQUIERDA, ATRAS_DERECHA, ATRAS_IZQUIERDA, STOP_STOP, SPEEDCAR_0, SPEEDCAR_1, SPEEDCAR_2, SPEEDCAR_3, SPEEDCAR_4, SPEEDCAR_5, SPEEDCAR_6, SPEEDCAR_7, SPEEDCAR_8, SPEEDCAR_9, DEF } CONDUCCION_NUM = DEF;

El funcionamiento del sistema se explicará estructurándolo en tres fases:

- En primer lugar el condicional “if” comprueba si el puerto Serial3 se encuentra habilitado (método “avilable”) y entonces se leen (método “read”) los datos recibidos guardados en “command”:

```
void Control_Wifi() {

        if (Serial3.available() > 0) {
          command = Serial3.read();
      ..............
```

- En segundo lugar los comandos recibidos se interpretan mediante una estructura tipo “switch case”, donde se asocian los datos recibidos con un tipo de conducción (tipo_Conduccion) y una acción (CONDUCCION_NUM). Cada comando (“letras” o “números”) se relaciona con un modo de conducción (conducción manual o auto conducción), por una parte, y con una instrucción relacionada con los distintos métodos implementados para hacer que el robot se mueva.

```
      switch (command)
{
    case 'F': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = ADELANTE; break;
    case 'B': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = ATRAS; break;
    case 'L': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = IZQUIERDA; break;
    case 'R': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = DERECHA; break;
    case 'I': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = ADELANTE_DERECHA; break;
    case 'G': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = ADELANTE_IZQUIERDA; break;
    case 'J': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = ATRAS_DERECHA; break;
    case 'H': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = ATRAS_IZQUIERDA; break;
    case '0': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = SPEEDCAR_0; break;
    case '1': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = SPEEDCAR_1; break;
    case '2': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = SPEEDCAR_2; break;
    case '3': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = SPEEDCAR_3; break;
    case '4': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = SPEEDCAR_4; break;
    case '5': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = SPEEDCAR_5; break;
    case '6': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = SPEEDCAR_6; break;
    case '7': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = SPEEDCAR_7; break;
    case '8': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = SPEEDCAR_8; break;
    case '9': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = SPEEDCAR_9; break;
    case 'S': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = STOP_STOP; break;
    case 'O': Rele_1_ON = true; break;
    case 'N': Rele_1_ON = false; break;
    case 'M': CamaraON = true; break;
    case 'D': CamaraON = false; break;
    case 'A': tipo_Conduccion = AUTO_CONDUCCION; break;
   case 'C': tipo_Conduccion = CONDUCCION_MANUAL; CONDUCCION_NUM = STOP_STOP; break;
    default: break;
 }
```

- En tercer lugar, dependiendo del tipo de conducción y la instrucción asociada a cada comando, se establecen las acciones que han de ser realizadas. Los casos ‘O’, ‘N’, ‘M’ y ‘D’ se resuelven directamente en la estructura “case” descrita en la relación anterior ya que solo se trata de activar, a través de dos pines habilitados como salidas, dos relés que servirán para conectar/desconectar el sistema y para activar/desactivar una cámara IP, respectivamente, y estas acciones son independientes del modo de conducción. Mencionar, finalmente, el caso donde se recibe el comando ‘A’, que establece el modo de conducción AUTO_CONDUCCION y el caso ‘C’ que desactiva este modo llevándolo de nuevo al tipo de conducción CONDUCCION_MANUAL, parando los motores (STOP_STOP).

  Llegados a este punto se hace necesario explicar cuál es la tarea asignada a la función “Control_Robot()”, llamada por “loop()”. Esta función se encarga de establecer una correspondencia entre el valor asignado a CONDUCCION_NUM (ADELANTE, ATRAS, IZQUIERDA… SPEEDCAR_1, SPEEDCAR_2, etc.), y los métodos creados para

controlar los distintos movimientos del robot y la velocidad de desplazamiento. Para conseguirlo, al igual que en el punto anterior, se ha optado por crear una estructura tipo "switch case":

```
void Control_Robot()
{
 if (tipo_Conduccion == CONDUCCION_MANUAL)
 {
  switch (CONDUCCION_NUM)
  {
   case ADELANTE: goAhead(); break;
   case ATRAS: goBack(); break;
   case IZQUIERDA: turnLeft(); break;
   case DERECHA: turnRight(); break;
   case ADELANTE_DERECHA: goAheadRight(); break;
   case ADELANTE_IZQUIERDA: goAheadLeft(); break;
   case ATRAS_DERECHA: goBackRight(); break;
   case ATRAS_IZQUIERDA: goBackLeft(); break;
   case STOP_STOP: stopRobot(); break;
   case SPEEDCAR_0: VelocMotores(100, 100, 100); break;
   case SPEEDCAR_1: VelocMotores(120, 120, 120); break;
   case SPEEDCAR_2: VelocMotores(135, 135, 135); break;
   case SPEEDCAR_3: VelocMotores(155, 155, 155); break;
   case SPEEDCAR_4: VelocMotores(170, 170, 170); break;
   case SPEEDCAR_5: VelocMotores(185, 185, 185); break;
   case SPEEDCAR_6: VelocMotores(195, 195, 195); break;
   case SPEEDCAR_7: VelocMotores(215, 215, 215); break;
   case SPEEDCAR_8: VelocMotores(235, 235, 235); break;
   case SPEEDCAR_9: VelocMotores(255, 255, 255); break;
   default: break;
  }
  CONDUCCION_NUM = DEF;
 }
 else if (tipo_Conduccion == AUTO_CONDUCCION)
 {
  Detectar_obstaculo_Sharp();
 }
}
```

Como puede verse en el cuadro anterior, cuando el estado de conducción es el de "AUTO_CONDUCCION" se llama a la función "Detectar_obstaculo_Sharp()" que es la función que engloba los métodos creados que permiten al robot moverse evadiendo obstáculos. En esta función se han unificado, en un único método, las funciones que se describen en el apartado correspondiente a la detección de obstáculos mediante sensores Sharp:

```
void Detectar_obstaculo_Sharp()
{
 detectarObstaculo();
 Marchar();
}
```

- Para finalizar esta descripción, hay que recalcar qué el sistema encargado de la detección de falta de piso, "Detectar_suelo()", funcionará independientemente de cuál sea el estado de conducción, es decir, detectará tanto en el modo de "CONDUCCION_MANUAL" como en el modo "AUTO_CONDUCCION", tal y como puede observarse en el diagrama de flujos de la imagen (66).

6.5 Implementación práctica

La conexión física de los dos placas se llevará a cabo tal y como se mostró en el esquema del punto 6.4.1. Además de la interconexión de los puertos Tx y Rx (el puerto transmisor de una se conecta con el puerto receptor de otra, y viceversa) de cada una de las placas hay que conectar en común las masas de ambas placas, ya que en caso contrario el sistema no funcionará convenientemente. En la imagen (67) puede observarse el modo de conectar las dos placas.

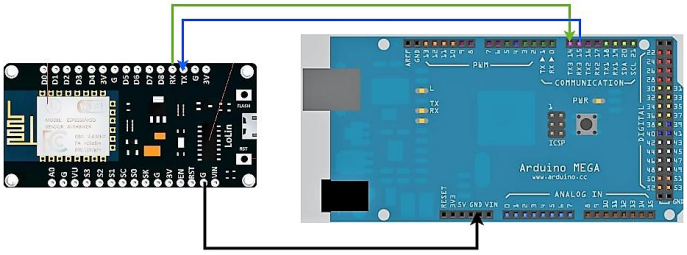


Imagen 67. Conexión de los puertos Tx y Rx entre Arduino y nodeMCU.

6.6 Análisis de resultados y conclusiones

Las modificaciones más importantes realizadas en el código, para poder comprobar el control a distancia mediante el teléfono, ya han sido señaladas en el apartado correspondiente por lo que no se cree necesario profundizar más en este tema. Dicho esto, se procedió a cargar el programa realizado en la placa del Arduino y realizaron las conexiones entre los dos puertos serie de las dos placas.

Para testear el funcionamiento del sistema se han realizado las siguientes comprobaciones:

- **Tiempo de respuesta de la conexión entre robot y dispositivo móvil:** la conexión es correcta, no se han detectado errores cuando se conectan los dos dispositivos.
- **Probabilidad de desconexión entre robot y dispositivo móvil:** el sistema no se desconecta del Smartphone cuando se mantiene a la espera sin presionar ninguna de las teclas, se ha comprobado manteniendo el robot inactivo durante varias horas y al pulsar de nuevo en el teclado respondió correctamente.
- **Tiempo de reacción del sistema (retardos) ante la pulsación de cualquiera de las teclas:** la respuesta es correcta, no se observan retardos en la ejecución de las órdenes enviadas.
- **Respuesta del sistema cuando se envían los comandos para encender o apagar los relés:** estas instrucciones son independientes del tipo de conducción en que se encuentre el robot y en cualquiera de los casos se ejecutan las órdenes correctamente sin que se vea afectado el modo de conducción.
- **Transición entre tipos de conducción:** en el modo evasión de obstáculos, o auto-conducción, cualquier tecla que se presione, con excepción de las de activación/desactivación los relés, modifican el modo a conducción manual. Por el contrario, en el modo de conducción manual sólo se pasa al estado de auto conducción presionando la tecla “ON” correspondiente.

Conexión WiFi entre el teléfono y el robot	Desconexiones del WiFi mientras no se actúa sobre el teclado	Respuesta de los pulsadores	Activación de los relés mediante botones	Respuesta del sistema a ordenes enviadas en cualquiera de los dos modos principales
CORRECTO	CORRECTO	CORRECTO	CORRECTO	CORRECTO

Tabla 14. Pruebas realizadas y resultados obtenidos del programa de control WiFi instalado en el Arduino.

7. Sistema de control del nivel de carga de las baterías del robot (Módulo 4)

El control de la alimentación de un robot es fundamental para su correcto funcionamiento. Si no se controla adecuadamente los niveles de carga de las baterías los distintos sistemas pueden empezar a fallar debido al incorrecto nivel de tensión. Por esta razón se ha creído conveniente

diseñar un sistema con el que poder visualizar de forma directa el nivel en que se encuentra las baterías, y que avise cuando el voltaje requerido no alcance unos límites mínimos preestablecidos.

7.1 Requisitos del sistema

El sistema de alimentación tiene que cumplir los siguientes requisitos:

- La tensión de alimentación debe ser de 12 Voltios, que es la tensión a la que deben alimentarse los motores.
- La placa Arduino Mega puede ser alimentada directamente o a través del shield para sensores que se le ha añadido, de 7 a 16 Voltios, por lo que la alimentación de 12 voltios elegida es correcta.
- La placa nodeMCU también ha sido montada en un shield especial de expansión, por lo que también puede ser alimentada directamente con 12 voltios.
- La placa de potencia controladora de los motores se alimentará con 12 V, conforme a lo requerido por los mismos.
- Todos los sensores y actuadores precisan de 5 voltios, por lo que se tomaran de los pines de la placa shield montada sobre el Arduino Mega.
- El sistema de alimentación debe ser capaz de suministrar el suficiente amperaje para poder alimentar todos los sistemas implementados en el robot. Atendiendo a las mediciones realizadas, con todos los elementos conectados, el consumo pico máximo medido es de 1,5 A, no llegando el consumo normal a sobrepasar los 0,8 A.

Aparte de los requerimientos mínimos de voltaje y potencia, deben resolverse otra serie de cuestiones:

- El sistema debe poder determinar el nivel de voltaje en la batería (no el de la salida del convertidor DC/DC que se explicará abajo), y el consumo durante su funcionamiento. A tal efecto se implementará un medidor digital que incorpora un voltímetro/amperímetro en el chasis del robot.
- El voltaje máximo a aplicar al conjunto no debe sobrepasar los 12 Voltios, por lo que la alimentación de todos los circuitos se realizará a través de un convertidor DC/DC (capaz de soportar el consumo de corriente del robot) que límite el voltaje suministrado por las baterías a 12 Voltios: la alimentación se llevará a cabo mediante dos baterías de 7,2V en serie, por lo que la tensión conseguida será de algunos voltios superior a lo requerido.
- El sistema deberá indicar cuál es el nivel de voltaje disponible a la salida del convertidor DC/DC, indicándolo visualmente mediante el uso de Leds de distintos colores y escribiéndolo en una pantalla LCD, donde se indicará cual es el nivel de carga disponible.
- Aprovechando el uso de la pantalla LCD, también se implementará un reloj que cuente el tiempo que el robot ha estado funcionando desde su conexión.
- Cuando el nivel de carga de las baterías sea insuficiente y deba procederse a su carga, será necesario que la acción se lleve a cabo sin tener que desconectar físicamente las baterías del resto del sistema.

7.2 Programa de control del nivel de carga

En la figura (68) se muestra el diagrama de flujo del programa creado para controlar la carga de la batería.

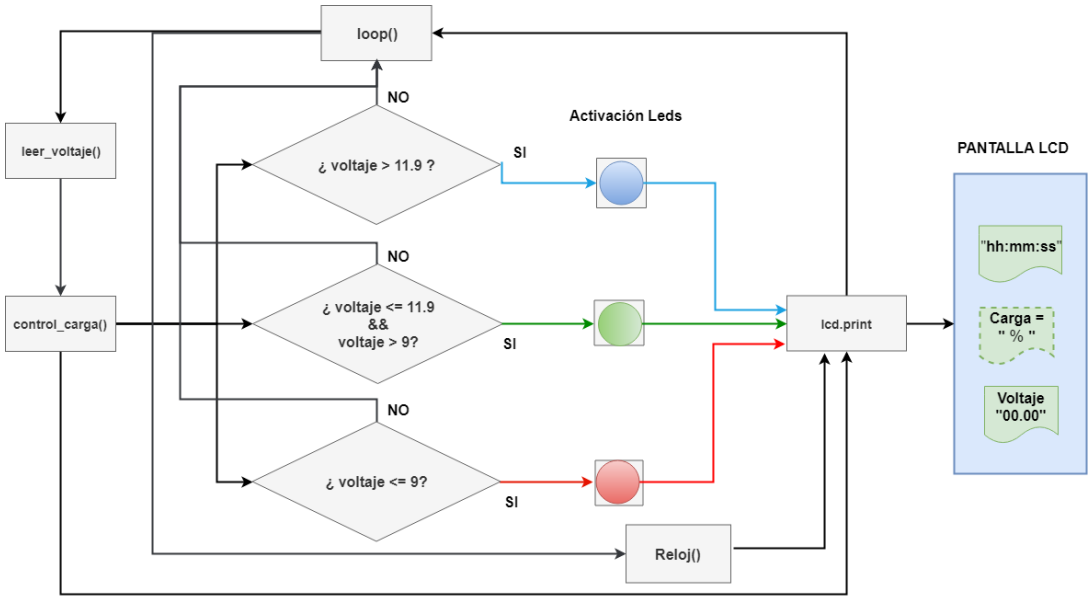


Imagen 68. Diagrama de flujos del sistema de control de carga de la batería.

Siguiendo el esquema, a la izquierda puede verse un bloque “leer_voltaje()” que representa un método que el nivel de tensión disponible, esta función toma de forma reiterada la lectura (medida) del valor de tensión existente en la fuente de alimentación. La lectura se realiza a través de entradas analógica del micro controlador y sobre este aspecto hay que indicar, aunque ya se hará una descripción más profunda en el apartado del montaje, que ha habido que adaptar los niveles de tensión disponibles en la batería a los de la entrada máxima admisible por el micro controlador. El valor leído es convertido a niveles lógicos, entre 0 y 1023, realizando el producto de la lectura entregada al pin analógico por 12 y dividiendo el resultado por los 1023 niveles del convertidor de 10 bits.

$$voltaje = \frac{medida \cdot 12}{1023}$$

El resultado se evalúa mediante una serie de condicionales que comparan el valor obtenido con un límite determinado mediante el método “control_carga()”. En este sistema se ha establecido que una medida de voltaje superior a 11,9 supone una carga máxima de la batería; un valor menor a 11,9 y mayor que 9 una carga normal; y un valor menor a 9 es un nivel de carga crítica. En cada caso se activará un pin habilitado a tal efecto (pasara de estado LOW a HIGH) que encenderá el Led azul, el verde y el rojo, respectivamente, cuando se cumpla la condición establecida para cada situación. Además, los datos correspondientes al voltaje y al porcentaje de carga serán escritos en la pantalla LCD.

Cuando la lectura corresponde a una carga crítica, además de encenderse el LED rojo será necesario desconectar el interruptor general que interrumpirá la conexión en serie de las dos baterías, habilitando la carga de las baterías a través de dos relés, y de este modo se podrá proceder a su carga sin necesidad de realizar ninguna desconexión de cables en el sistema.

Además de la lectura de la tensión mediante el método ya explicado, también se ha realizado (dentro del método “leer_voltaje”) una conversión a porcentaje de carga, basado en el parámetro voltaje. Se ha considerado, a tal efecto, que la batería se encuentra cargada totalmente cuando se alcanza el 88% (se supone que se ha alcanzado el máximo de voltaje). Se ha utilizado la siguiente expresión para conseguir el porcentaje de carga:

$$porcentaje(\%) = \frac{[(voltaje \cdot 100) - 320]}{10} = \frac{\left[\left(\left(\frac{voltaje \cdot 12}{1023}\right) \cdot 100\right) - 320\right]}{10}$$

En el programa se ha incluido un reloj para poder medir el tiempo. Este reloj se ha creado mediante una variable “t” donde se guarda el tiempo transcurrido contado por la función “millis()”, dividiéndolo por mil ya que el tiempo entregado por esta función viene dado en milisegundos, y después se han creado las horas, minutos y segundos como variables enteras, tal y como se muestra en la siguiente tabla (15).

long t	millis() / 1000
int horas	t / 3600
int minutos	(t % 3600) / 60
int segundos	(t - horas * 3600 – minutos * 60) % 60

Tabla 15. Código reloj

Por último, en el bloque de la pantalla LCD se ha identificado los datos que se mostrarán en la misma: los del reloj, los de la lectura del voltaje y el porcentaje de carga de la batería.

7.3 Montaje práctico

En primer lugar hay que puntualizar que la tensión objeto de control no será la que entreguen las baterías directamente, sino la que se obtiene de un convertidor DC/DC que tendrá la función de evitar que el voltaje de alimentación pueda ser superior a 12 voltios. Se utilizará, entonces, un circuito comercial capaz de realizar esta función. Al igual que en apartados anteriores, la realización práctica se ha efectuado usando los elementos de los que se disponía.

7.3.1 Elementos utilizados

La relación de los elementos utilizados en este sistema son los siguientes:

- Dos baterías de 7,2 V y 4000 mAh, que se conectarán en serie para conseguir algo más de 14 Voltios.
- Voltímetro/amperímetro digital para controlar la tensión de las baterías y el consumo.
- Un convertidor DC/DC de 7-32 V a 0,8-28V, capaz de aguantar picos de 12 amperios y 300 W de potencia y otro convertidor DC/DC LM2596 para alimentar la placa de relés.
- Módulo Led de tres colores (Azul, Verde, Rojo).
- Una resistencia de 10KΩ, una de 5,6 KΩ y una resistencia variable de 4,7KΩ.
- Placa de circuito impreso para montar el divisor de tensión, cables, etc.
- Pantalla LCD.
- Dos de los relés de la placa de cuatro unidades.
- Shield expansión para nodeMCU.

7.3.2 Realización práctica

La placa nodeMCU se montará sobre un “shield” de expansión, por lo que el voltaje de alimentación de la misma podrá ser obtenido directamente de los 12 voltios de salida del convertidor DC/DC utilizado para alimentar todo el sistema. Para poder medir el voltaje de las baterías en el micro controlador es necesario realizar un pequeño circuito de adaptación de la tensión a la salida del convertidor a la máxima admitida por el pin analógico. Es decir, se tratará de “convertir” el valor máximo de referencia de la fuente de alimentación (12 V) al valor máximo de la entrada analógica de la placa, o dicho de otro modo, para poder medir el nivel de carga de la batería mediante el micro controlador de la nodeMCU, este deberá “entender” que una entrada analógica de 12 voltios son 3,3 V. Esta adaptación se ha realizado mediante un divisor de tensión con el fin de que en la entrada del pin analógico nunca llegue a haber más de 5 voltios, en el caso del Arduino, ó 3,3 voltios en el caso de la nodeMCU.

Partiendo de una resistencia fija de $10K\Omega$ (R1) como una de las ramas del divisor de tensión, se puede calcular fácilmente el valor de la otra resistencia (R2):

$$R_1 = \frac{R_2(V_{cc} - V_{out})}{V_{out}}$$

Donde V_{cc} es el voltaje de entrada que se quiere reducir y V_{out} el de la tensión que se quiere adaptar a la entrada analógica de la placa.

Sustituyendo valores, si se busca una salida para $V_{out} = 5\,V$, se obtiene un valor para $R_2 = 7143\,\Omega$. En el caso de querer obtener una salida $V_{out} = 3.3\,V$, la resistencia buscada $R_2 = 3793\,\Omega$. Para su implementación práctica los valores de resistencias comerciales no son suficientes, por lo que para conseguir un valor muy próximo a 7143 Ω (salida de 5V) se utilizarán una resistencia de $5{,}6\,K\Omega$ en serie con una resistencia variable de $4{,}7\,K\Omega$. Para el otro caso (salida de 3,3 V) basta con usar la resistencia variable $4{,}7\,K\Omega$ y ajustarla hasta el valor más próximo a los $3793\,\Omega$ buscados. El esquema de los dos circuitos, divisores de tensión, puede verse representados en la figura (69).

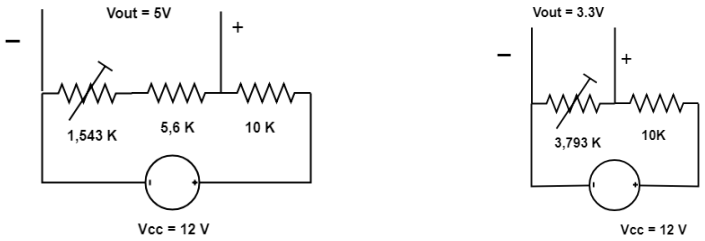


Imagen 69. Esquema eléctrico de dos divisores de tensión.

A continuación se describirá el conexionado entre todos los componentes que integran el sistema con la ayuda del siguiente esquema de la imagen (70). Como se puede observar en el diagrama de conexionado, se ha utilizado la tarjeta nodeMCU como controladora del sistema y de este modo aprovechar las capacidades de esta placa de desarrollo en la que solo se había cargado el programa de control del módulo *wifi*.

También puede verse la conexión de la pantalla LCD que se ha realizado mediante un bus serie I2C (módulo que incorpora la pantalla) con lo que solo se necesitan las conexiones para la alimentación y otros dos cables que conectan las salidas habilitadas en el micro controlador con las entradas SDA y SCL de la pantalla.

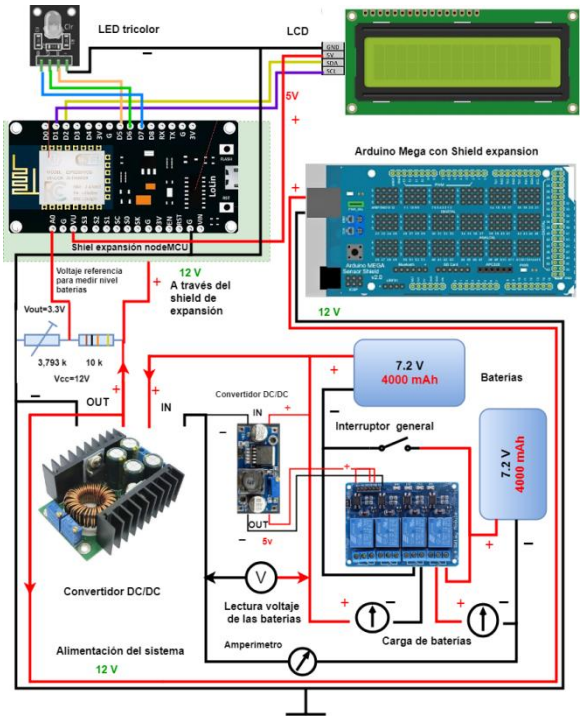
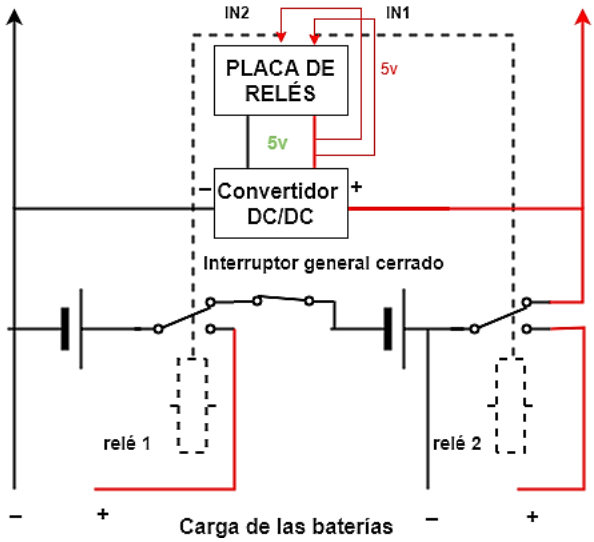


Imagen 70. Montaje práctico control alimentación.

Para finalizar, hay que indicar que la carga de las baterías debe hacerse con el interruptor general desconectado, ya que de este modo, al interrumpirse la alimentación de la placa de

relés, los contactos que permiten la alimentación del robot se conmutan para permitir que puedan cargarse las baterías sin que llegue corriente al sistema (imagen 71, izquierda).
El nivel óptimo de carga lo establecerá el cargador de las baterías. Una vez las baterías estén cargadas se procederá a conectar el interruptor general y el robot se encontrará en disposición de ser utilizado.



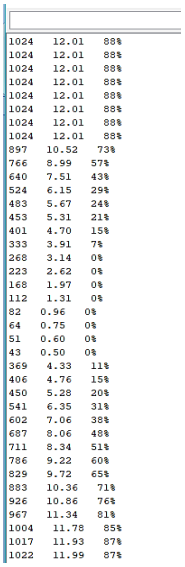


Imagen 71. Esquema funcionamiento relé (izquierda) e Impresión por puerto serie del valor digital (derecha).

7.4 Análisis de resultados y conclusiones

El montaje final ha sido sometido a un test de pruebas para detectar posibles fallos una vez instalado el programa en la placa nodeMCU. Las pruebas realizadas se han basado en comprobar los siguientes puntos:

- Correspondencia entre los valores de voltaje obtenidos con los realmente aplicados. En la Imagen (71, derecha) se pueden ver los valores impresos por el puerto serie de Arduino, donde la columna de la izquierda representa los niveles lógicos, la columna del centro es el voltaje y la columna de la derecha representa el porcentaje calculado. En la tabla se representa la correspondencia entre los voltajes aplicados al sistema, antes del convertidor DC/DC y después del mismo, medidos con un milímetro de 6 ½ dígitos de precisión, y el voltaje leído en el puerto serie de Arduino.

Voltaje aplicado (previo al convertidor DC/DC)	**Voltaje leído por el puerto serie (sin convertidor DC/DC)**	**Voltaje leído por el puerto serie (con convertidor DC/DC)**
14,568.91 V	(No procede)	12,01 V
13,069.81 V	(No procede)	12,01 V
12,016.70 V	11,98 V	11,98 V
11,023.52 V	10,93 V	10,93 V
10,017.01 V	9,96 V	9,96 V
9,018.260 V	8,97 V	8,97 V
8,000.720 V	7,96 V	7,96 V
7,091.150 V	7,06 V	7,06 V
6,025.010 V	6,02 V	6,02 V
5,043.840 V	5,07 V	5,07 V
4,063.470 V	4,08 V	4,08 V

3,032.530 V	3,03 V	0,04 V
2,010.490 V	2,06 V	0,04 V
1,009.001 V	1,04 V	0,04 V
0,00.000 V	0,04 V	0,04 V

Tabla 16. Correspondencia entre voltajes aplicados y valores obtenidos en la placa.

Como se desprende de la tabla (16) el voltaje aplicado se representa con bastante precisión en el puerto serie. En la columna de tensiones medidas desde la salida del convertidor se observa que para valores menores a 3 voltios el sistema es incapaz de medir el valor aplicado, esto es debido a que el convertidor DC/DC trabaja con una tensión mínima de conversión de algo más de 3 voltios. A efectos prácticos esto no es relevante ya que lo que se busca es que el sistema reacciones a valores bastante más elevados que ese mínimo.

- Comprobación del encendido correcto de los Leds indicadores de los distintos estados de carga (tabla 17):

Límite establecido para encender el Led Azul: (voltaje > 11,5 V)	**Límites establecidos para encender el Led Verde: (Voltaje <= 11,5V && voltaje > 9V)**	**Límite establecido para encender el Led Rojo: (voltaje <= 9V)**
Respuesta comprobada: (Voltaje > 11,596.87 V	Respuesta comprobada: (voltaje <=11,596.25 V && voltaje > 9,006.60 V)	Respuesta comprobada: (Voltaje < 9,006.60 V)

Tabla 17. Testeo de la correcta activación de los Leds indicadores.

En conclusión, los resultados obtenidos se ajustan a los objetivos y requerimientos establecidos.

8. Ampliaciones y mejoras (Módulo 5)

Como tareas secundarias a implementar en este proyecto se han realizado tres sencillas aplicaciones que serán descritas a continuación.

8.1 Activación de una cámara IP mediante el mando a distancia

Esta funcionalidad consistirá en activar/desactivar la alimentación de una cámara IP que podrá ser instalada en el chasis del robot. El funcionamiento de su activación ya fue explicado en el apartado 6.4.2 de este proyecto por lo que no se volverá a incidir sobre el mismo tema.

8.1.1 Implementación práctica

La cámara utilizada dispone de su propia APP por lo que sólo se entrará a detallar el esquema de conexión, que es muy simple, como puede verse en la figura (72). Aquí se observa que la señal de activación del relé se extrae del pin digital 43 del Arduino que cuando pasa a estado alto cierra el circuito, permitiendo la alimentación de la cámara.

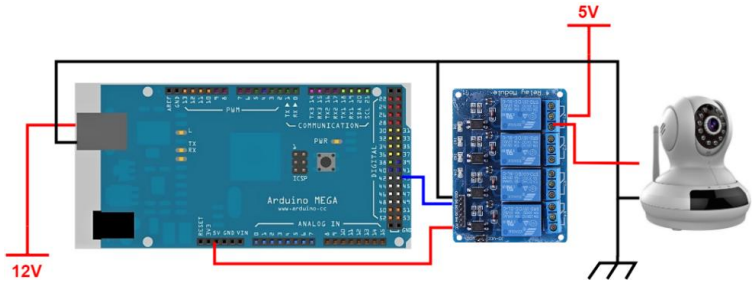


Imagen 72. Conexión de la alimentación de la cámara IP.

8.2 Señalización acústica mediante zumbador

Se ha creado un pequeño programa para controlar la activación de un zumbador que sirva para indicar ciertos estados. Para este proyecto sólo se va a utilizar para avisar acústicamente cuando el robot detecte la falta de suelo.

8.2.1 Código del programa

En la figura (73) se observa el diagrama de flujo de este sencillo programa.

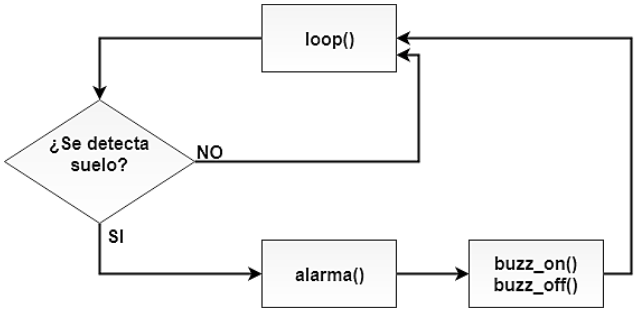


Imagen 73. Diagrama de flujos del programa del buzzer.

En este programa la función principal es “alarma()” que es llamada al bucle cuando el método “detectar_suelo()” detecta falta de piso en cualquiera de los sensores utilizados a tal fin. El método “alarma()” llamará primero a la función “buzz_on()” activándola durante 100 milisegundos durante los cuales el pin digital asignado para su funcionamiento se pondrá a nivel bajo (LOW), y después a la función “buzz_off()” pasando el pin al estado alto (HIGH), produciendo el sonido característico de este dispositivo. Por su simplicidad se ha reproducido el código en la siguiente tabla (18).

void alarma()	void buzz_on()	void buzz_off()

{
buzz_on();
delay(100);
buzz_off();
}
	{
	digitalWrite(buzzer, LOW);
	}
		{
		digitalWrite(buzzer, HIGH);
		}

Tabla 18. Código del programa del zumbador.

8.2.2 Montaje práctico

El esquema de conexionado se ha representado en la figura (74).

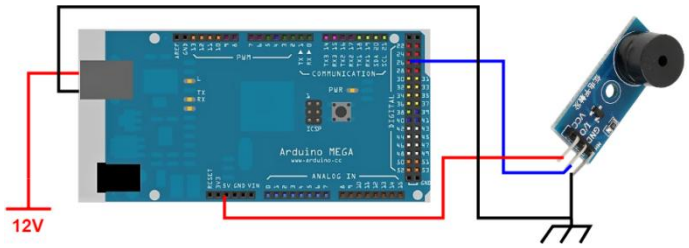


Imagen 74. Conexionado del zumbador.

El buzzer solo tiene tres conexiones, dos para la alimentación (5 voltios) y otra para la señal de activación que se conecta al pin digital 26 del Arduino Mega, tal y como puede verse en la figura .

8.3 Programa para obtener la velocidad de desplazamiento del robot

Cómo última tarea, se ha realizado un código para calcular el número de revoluciones por minuto así como la velocidad de desplazamiento del robot.

8.3.1 Requisitos

El programa tiene que poder calcular de modo aproximado las revoluciones por minuto de las ruedas y la velocidad de desplazamiento del robot, en base a los pulsos generados por un encoder.

8.3.2 Elaboración del Programa

El cálculo de la velocidad de desplazamiento del robot se puede aproximar relacionando el perímetro de la rueda y la resolución en pulsos del encoder, mediante la siguiente expresión:

$$\frac{Desplazamiento\ lineal}{pulso} = \frac{2\pi R}{Resolución}$$

Donde el lado izquierdo de la igualdad relaciona la distancia lineal recorrida por cada pulso y el lado derecho el perímetro de la rueda con la resolución del encoder (una vez aplicada la reducción de los engranajes). En lo que respecta a la resolución total del encoder (aplicada la educción del engranaje) esta tiene un valor de 990 pulsos por cada vuelta, los cuales se envían a uno de los pines con interrupciones del Arduino Mega (en este programa se utilizará el pin 18).

Por otro lado, a la hora de establecer la distancia recorrida hay que tener en cuenta la orientación de las ruedas con respecto a la dirección cuando el robot se desplaza en línea recta. El ángulo que ofrecen las ruedas sobre la dirección en línea recta es de 60°, por lo que habrá que corregir la distancia añadiendo una modificación en la anterior expresión:

$$\frac{Desplazamiento\ lineal}{pulso} = \frac{2\pi R}{Resolución} \cdot sen(60)$$

$$(Donde\ R = 2{,}9\ cm\ (radio\ de\ la\ rueda)\ y\ \ resolución = 990\ (resolución\ total\ del\ encoder)$$

Se tiene, entonces, que la distancia recorrida por cada interrupción, o pulso, es igual a:

$$1\ interrupción = 0{,}018\ \cdot sen(60) = 0{,}01380\ cm$$

La información que se pretende visualizar es la correspondiente a la velocidad de desplazamiento y las revoluciones por minuto, por lo que simplemente habrá que determinar el número de revoluciones de la rueda por unidad de tiempo, sin olvidar que el valor del tiempo entregado por la función “millis” está en milisegundos.

Al relacionar el número de vueltas con el tiempo se obtiene el parámetro frecuencia de revolución “f”:

$$f = n \cdot \frac{vuelta}{seg} = n \cdot \frac{990}{1000}$$

Se ha creado el parámetro “n” (número de vueltas) cuyo valor será determinado por un contador que llevará la cuenta de las interrupciones registradas hasta 990, equivalente a una vuelta de la rueda, y pondrá el contador a cero cada vez que se alcance esta cifra, incrementando el número de vueltas en una unidad.

Y la velocidad de desplazamiento, en m/s, será entonces:

$$v = n \cdot \frac{990}{10^3} \cdot 13{,}80 \cdot 10^{-2} = n \cdot 0{,}13662\ m/s$$

Para obtener las revoluciones por minuto (rpm), basta con multiplicar la frecuencia de vuelta por 60:

$$rpm = f \cdot 60 = n \cdot 0{,}99 \cdot 60$$

Para realizar estos cálculos se han creado dos funciones, una denominada “encoder()” y otro método denominado “contador2”, además de utilizar la función “millis”. En el siguiente diagrama de flujos (imagen 75) puede observarse su funcionamiento.

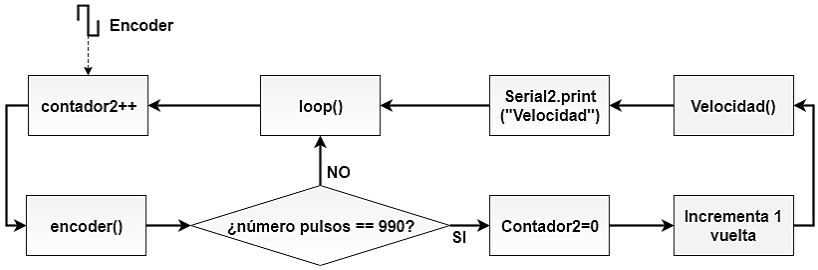


Imagen 75. Diagrama de flujos del programa para medir la velocidad.

8.3.3 Cálculo del ángulo de giro

Además de lo ya expuesto, en lo que respecta a la velocidad de desplazamiento, su uso también puede servir para calcular el ángulo en que gira el robot cuando las ruedas se mueven con la misma velocidad en sentidos opuestos, ya que el robot girará en torno a un eje vertical situado en su centro.

La longitud de la circunferencia que describen las ruedas del robot cuando giran alrededor de su eje central es de:

$$Perimetro\ de\ la\ circunferencia = \pi \cdot 25\ cm = 78{,}54\ cm$$

Teniendo en cuenta que por cada vuelta de rueda los encoders producen 990 PRR, se puede establecer la relación entre interrupción y recorrido, utilizando la siguiente expresión:

$$1\ interrupción = \frac{\pi \cdot 5.8\ cm}{990} = 0{,}01841\ cm$$

> Nota: En este caso no es necesario aplicar ninguna corrección sobre el desplazamiento que realizan las ruedas ya que la disposición de cada una de ellas es tangente a la circunferencia del cuerpo del robot.

Se deduce, entonces, que se necesitan 4266 interrupciones $\left(\frac{78{,}54}{0{,}01841} = 4266\right)$ para completar una vuelta, por lo que cada interrupción se corresponde con un ángulo igual a:

$$1\ interrupción\ \frac{360°}{4266} = 0{,}08439°$$

8.3.4 Aplicación en odometría

Los dos parámetros que habría que evaluar con el encoder son, por un lado, la distancia que avanza el robot (función de la velocidad y tiempo) y, por otro lado, el ángulo al que gira.

Para comprobar el primer parámetro (desplazamiento) se podría hacer avanzar al robot en línea recta hacia delante y hacia atrás: moviendo el robot hacia delante, el valor de los dos encoders (motor 1 y motor 2) se incrementa en una unidad cada vez que se produce una interrupción y, al contrario, cuando el robot se mueve hacia atrás los dos contadores reducen la cuenta en la misma cantidad. De este modo se lleva la cuenta de la distancia que el robot se ha desplazado (1 interrupción = 0,01841 cm) teniendo como referencia un punto determinado donde comenzó el movimiento. Ya que los tiempos de avance y retroceso serán iguales se podrán traducir los datos obtenidos de número de interrupciones a distancia en centímetros.

Por otro lado, si lo que se quiere es comprobar el ángulo al que gira el robot se tendría que hacer girar al mismo una vuelta completa y después hacerlo girar en sentido contrario. La contabilidad de las interrupciones (1 interrupción = 0,08439°) se verá incrementada cuando el robot gira en un sentido y se hará más pequeña cuando lo haga en sentido contrario.

Con estos cálculos, y otros de mayor complejidad, se podría diseñar un sistema de odometría con el cual guiar al robot a un punto concreto. Como ya se ha indicado estos cálculos aproximados se han realizado con la intención de experimentar y aclarar ideas para la realización, en un futuro, de ampliaciones y mejores del presente proyecto.

8.3.5 Resultados obtenidos

En las siguientes ilustraciones (imagen 76) se pueden ver las revoluciones impresas por pantalla correspondientes a las revoluciones por minuto la velocidad en metros por segundo.

Imagen 76. Mediciones de las rpm y la velocidad mediante el encoder.

9. Valoración económica

9.1 Presupuesto

Para poder determinar la viabilidad o no del proyecto se hará necesario analizar los costes derivados de su realización, analizando por una parte los costes materiales y por otra los relacionados con el trabajo dedicado a su estudio, diseño e implementación práctica.

9.1.1Costes materiales

Cantidad	Concepto	Precio unitario	Precio total
1	Chasis con motores y ruedas	80 €	80 €
2	Puente H doble L298N	1.38 €	2.76 €
4	Módulo sensor infrarrojos SHARP 2Y0A21	2.94 €	11.76 €
1	Convertidor DC/DC 12A	3.55 €	3.55 €
1	Interruptor de palanca	0.75 €	1.5 €

2	Batería Ni_MH 7.2V 4000mAh	20 €	40 €
1	Módulo 4 relés	1.75 €	1.75 €
1	Tarjeta Arduino Mega 2560	10 €	10 €
1	Tarjeta desarrollo NodeMcu V3	2 €	2 €
1	Cables de conexión, PCB perforada y conectores	5 €	5 €
1	Conversor DC/DC	3 €	3 €
1	Sensor Shield expansión para Arduino Mega	1.56 €	1.56 €
1	Sensor Shield expansión para NodeMcu	1.26 €	1.26 €
4	Sensores seguidores de línea TCRT5000	0.58 €	2.32 €
1	Voltímetro/amperímetro digital doble pantalla	1.63 €	1.63 €
1	Sensor Laser VL530X	2.63 €	2.63 €
1	Módulo LCD I2C	1.88 €	1.88 €
1	Módulo zumbador pasivo	2.29 €	2.29 €
	TOTAL		**172.60 €**

Tabla 19. Costes materiales del Robot.

9.1.2 Costes de desarrollo

El coste de desarrollo va a depender del número de horas empleadas en su desarrollo y el precio adjudicado a cada hora de trabajo efectivo empleadas. Suponiendo un coste medio de 15 €/h por mano de obra y un total de 286 h se tiene:

$$15\,\frac{€}{h} \cdot 286\ h = 4290\ €$$

Coste total (materiales + desarrollo): 4290 + 172,60 = 4462,60 €

9.2 Viabilidad

Para estudiar la viabilidad de de producir el robot con fines comerciales, se realizará una sencilla simulación en base al coste material de producción de una unidad y de los demás costes (fijos y variables) en los que se pueda incurrir.

La estimación de los costes fijos y variables, mínimos, se han representado en la siguiente tabla (8).

Tipo de coste	**Concepto**	**Costes (1)**
Coste fijo	Alquileres y otros	No se alquilará ningún local (se realizará en el propio domicilio)
Coste fijo	Seguros	Inicialmente no se contratará ningún seguro
Coste fijo	Amortización préstamos*	Se utilizarán los ahorros disponibles
Coste variable	Otros	150 €
Coste variable	Materiales consumidos	¿? Dependerá del número de unidades vendidas
Coste variable	Consumo electricidad, agua, etc. (una estimación proporcional sobre el consumo total del domicilio)	50 €
Coste fijo	Seguro autónomo titular	300 €

TOTAL mensual:	**500 €**

Tabla 8. Distribución de los costes de producción mensuales.

En función de esta estimación, para cubrir los costes totales habrá que comercializar un número mínimo de unidades que dependerá, además de los costes de producción fijos y variables indicados, del coste del material necesario para su realización. El número de unidades puede determinarse utilizando la siguiente ecuación:

$$n \cdot PVP > 500 + (n \cdot PM) \rightarrow n = \frac{Robots}{mes}$$

Donde PVP es el precio de venta al público, PM es el coste del material necesario para su construcción y "n" es el número de robots que hay que comercializar para cubrir los costes mínimos y el coste del material, teniendo en cuenta que no habrá beneficios y que este, en su caso, estaría constituido por el sueldo del autónomo.

En el supuesto de que inicialmente se obtuviese una rebaja en la adquisición del material, sobre el establecido en la tabla de costes materiales del robot, de por ejemplo un 30%, y que se estableciese un precio de venta al público, en base al estudio de mercado que ya se hubiese realizado de 200 €, se tendría:

$$n \cdot 200 > 500 + n \cdot \left(172.60 - 172.60 \cdot \frac{30}{100}\right)$$

$$n = 6{,}32 \frac{Robots}{mes} \rightarrow 7\ robots/mes$$

Con estos datos ya se puede evaluar el PVP mínimo y el coste total mensual correspondiente para no incurrir en pérdidas:

$$\boldsymbol{Número\ de\ unidades\ a\ vender = 7}$$
$$\boldsymbol{Ingresos\ brutos = 1400}\ €$$
$$\boldsymbol{Costes\ totales = 500 + (120{,}4 \cdot 7) = 1342{,}8}\ €$$

Mencionar, finalmente, que este supuesto no se han tenido en cuenta los impuestos, ya se trate de de impuestos indirectos como el IVA o los directos como el IRPF.

10. Conclusiones y trabajo futuro

Analizando los resultados del proyecto se puede concluir que:

- Se han cumplido los objetivos principales del proyecto y también los secundarios ya que se ha construido un prototipo con las funcionalidades que en un principio se habían planificado.
- La planificación temporal se ha seguido correctamente.
- Desde un punto de vista didáctico se ha entrenado el uso de distintas herramientas de diseño y programación.

Por otra parte, como **trabajo futuro**, el prototipo podría ser mejorado en los siguientes puntos:

- Integración de imagen de la cámara y controles en una única GUI.
- Realización de un programa de auto-guiado.
- Mejorar la calidad del Hardware empleado, en general.

11. Glosario

ADC: Analog to Digital Converter

APP: aplicación informática

Bootloader: gestor de arranque

CPU: Central Processing Unit

DAC: Digital to Analog Converter

DC: Direct Current

E/S: Entrada/Salida

EEPROM: Electrically Erasable Programmable Read Only Memory

GPU: Graphics Processing Unit

HTML: Hyper Text Markup Language

HTTP: Hypertext Transfer Protocol

ICSP: In Circuit Serial Programming

IDE: Integrated Development Environment

I/O: Input/Output

IP: Internet Protocol

I2C: Bus serie de datos

KB: kilobyte

LASER: Light Amplification By Stimulated Emission of Radiation

LED: Light Emitting Diode

MCU: Microcontroller Unit

PC: Personal Computer

PPR: Pulse Per Revolution

PCB: Printed Circuit Board

PWM: Pulse Width Modulation

RAM: Randon Acces Memory

SBC: Single Board Computer

SD: Secure Digital

SDL: Simple Direct Media Layer

SO: Sistema Operativo

SPI: Serial Peripheral Interface

ToF: Time-Of-Flight

UART: Universal Asynchronous Receiver-Transmitter

USB: Universal Serial Bus

URL: Uniform Resource Locator

12. Bibliografía

[1] Sistemas empotrados: Ignasi Vilajosana guillén, Introducción a los sistemas empotrados, UOC_PID_00177260. Consultado Mayo/2018

[2] Wikipedia, the free enciclopedia (2018). Raspberry_Pi. Consultado Diciembre/2018
https://en.wikipedia.org/wiki/Raspberry_Pi

[3] Arduino.cl (2018)- Arduino Mega 2560 R3. Consultado Octubre/2018
http://arduino.cl/arduino-mega-2560/

[4] Prometec.net (2018). Circuito con sensor de distancia. Tutoriales Arduino.
Consultado Marzo/2018.
https://www.prometec.net/curso-kit-inicio/

[5] Luis Llamas (2018). Medir Distancia con precisión con arduino y sensor láser VL53L0X Y VL6180X. Consultado Octubre/2018.
https://www.luisllamas.es/arduino-sensor-distancia-vl53l0x/

[6] Omar Sánchez (2018). Cinemática, modelos y conductas de Robots Móviles.
Consultado Marzo/2018.
https://es.slideshare.net/omarspp/cinematica-vehiculos

[7] Luis Ignacio Gracia Calandín. Tesis Doctoral. Modelado Cinemático y Control de Robots Móviles con Ruedas. (*Departamento de Ingeniería de Sistemas y Automática*. Universidad Politécnica de Valencia). [PDF online]. Consultado Octubre/2018.
https://riunet.upv.es/bitstream/handle/10251/1840/tesisUPV2519.pdf

[8] Luis Llamas (2018). NODEMCU, LA POPULAR PLACA DE DESARROLLO CON ESP8266. Consultado Octubre/2018.
https://www.luisllamas.es/esp8266-nodemcu/

13. Anexos

- **Enlace manual de utilización de nodeMCU V3 Lolin:**

http://www.handsontec.com/pdf_learn/esp8266-V10.pdf

- **Enlace Arduino Mega Datasheet:**

https://www.tiendaarduino.com/datasheet/arduino-mega_2560.htm#.XCP4tVxKiUk

- **Enlace Datasheet sensor infrarrojos Sharp GP2Y0A21:**

http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a21yk_e.pdf

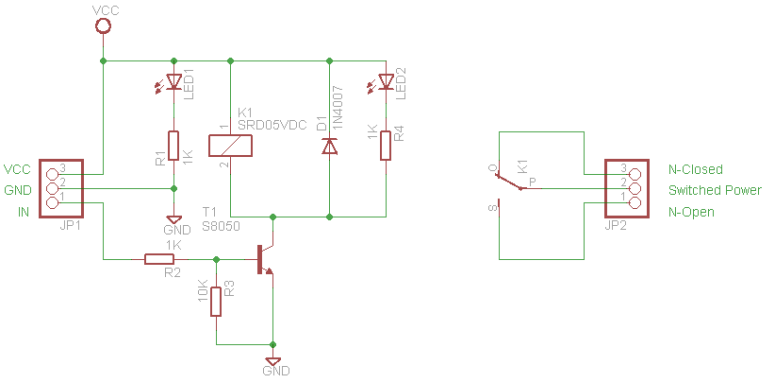
- **Enlace Datasheet sensor laser VL53L0X:**

https://www.st.com/resource/en/datasheet/vl53l0x.pdf

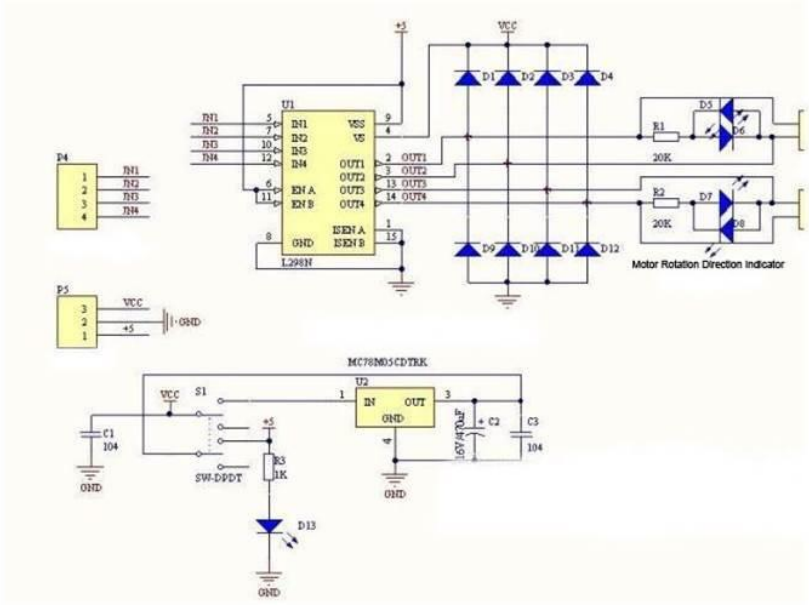
- **Enlace MIT APP inventor:**

http://appinventor.mit.edu/explore/

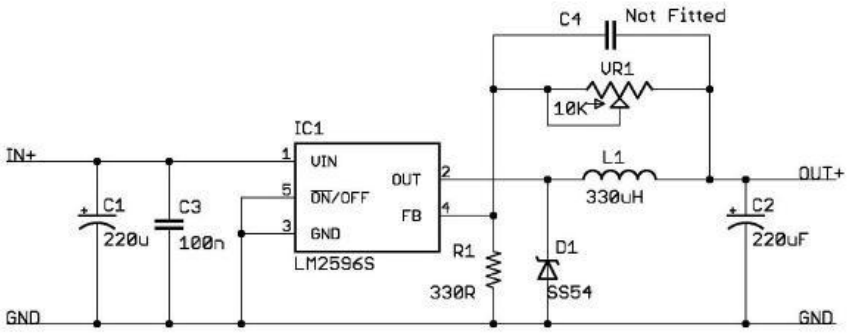
- **Esquema eléctrico modulo relés:**



- **Esquema eléctrico L298N:**



- **Esquema eléctrico Lm2596 convertidor DC/DC:**



- **Esquema eléctrico sensores infrarrojos seguidores de linea:**