uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería de Tecnologías Industriales
(2017-2018)

Trabajo Fin de Grado

"DISEÑO DE UN PROTOTIPO DE UN ROBOT ASPIRADORA CON COMPONENTES OPEN SOURCE"

Sandra Ramos Gutiérrez

Tutor

Juan Manuel Alonso Weber

Leganés, 2018

Abstract

This project corresponds to the design, construction and testing of an autonomous vacuum cleaner totally made with open source components. The present document will review the key vacuum cleaners that are sold in the market nowadays and it will also give a brief look to the main open source components that are compatible with Arduino.

In order to meet the requirements of this project, first of all, it was needed to perform a detailed investigation of the main motors and sensors that were compatible with Arduino. Some experiments were made with each sensor just to understand the way they work. Later on, it was necessary to build a preliminary prototype for the autonomous collision avoidance car.

In order to test if everything was correct, this prototype was done by parts, testing it every time that something new was added. After it was all working properly, the final prototype which also includes a suction motor was built using the circuits, parts and codes of the debugged preliminary prototype. This last prototype was a working autonomous vacuum cleaner made with open source products.

The document also provides several studies, including a market survey with the best-selling vacuum cleaners in 2017 in Spain, a budget to estimate the costs of the project and of the prototype, and a brief summary of the European legislation about open software and hardware.

Keywords

Vacuum cleaner, Arduino, open source, Roomba.

Resumen

Este proyecto corresponde al diseño, construcción y prueba de un prototipo de una aspiradora autónoma realizada totalmente con componentes *open source*. En el presente documento se revisarán las aspiradoras clave que se venden en el mercado hoy en día y también se expondrán los principales componentes *open source* compatibles con Arduino necesarios para este proyecto.

Para cumplir con los requisitos de este proyecto, en primer lugar, fue necesario realizar una investigación detallada de los principales motores y sensores compatibles con Arduino. Se realizaron algunos experimentos con cada sensor para comprender cómo funcionan. Más tarde, fue necesario construir un prototipo preliminar de coche autónomo capaz de evitar colisiones.

Para comprobar si todo era correcto, este prototipo se hizo por partes, probándolo cada vez que se agregaba algo nuevo. Una vez que todo funcionaba correctamente, se construyó el prototipo final, que también incluye un motor de succión, utilizando los circuitos, componentes y códigos del prototipo preliminar depurado. De esta forma se logró un último prototipo de una aspiradora autónoma que funcionaba con productos *open source*.

El documento ofrece también varios estudios, incluyendo un estudio de mercado con los robots aspiradoras más vendidos en 2017 en España, un presupuesto para estimar los costos del proyecto y del prototipo, y un breve resumen de la legislación europea sobre el *software* y *hardware* libre.

Palabras clave

Robot Aspirador, Arduino, código abierto, Roomba.

Tabla de contenido

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

1.1 Descripción general del proyecto

Por medio de la realización de este proyecto, se pretende conseguir un prototipo funcional de un robot aspirador que funcione de forma autónoma. Para ello se aplicarán los conocimientos adquiridos a lo largo del grado con el entorno *open source* y las facilidades que éste proporciona.

1.2 Motivación

A finales de los 90 surgió en California una nueva forma de entender el acceso a la tecnología [1]. La idea principal era que se puede aprender más unos de otros en un entorno en el que la información esté compartida y sea accesible de forma gratuita. De esta forma, todo el mundo puede innovar y desarrollar nuevos proyectos. La mayor implicación de esta nueva corriente de pensamiento consiste en que si todo el mundo tiene acceso a la misma información, se podrán solucionar los problemas de una forma más rápida y eficaz. Además, se fomenta la meritocracia y que todo el mundo pueda aportar su idea y compartirla globalmente [2]. Serán los propios usuarios de este movimiento los que decidan qué ideas son las mejores y también podrán hacer cambios para optimizarlas.

A través del trabajo comunitario, se puede aprender fabricando prototipos de una forma barata y sencilla, por lo que, aunque se fracase en algún momento, se podrá encontrar una solución al problema más adelante. Además, al permitir hacerse de forma grupal es posible aportar ideas y tratar de perfeccionar prototipos de otras personas. Todo esto permite una mejora y un aprendizaje continuo, basado en el trabajo propio y en el compartido con otros.

Este entorno se ha desarrollado mucho en los últimos años y se ha podido extender la robótica y la electrónica a mucha gente de una forma muy eficaz. Además, proporciona unas enormes ventajas educativas ya que todo el mundo puede fabricar sus propios prototipos sin necesidad de haber cursado carreras técnicas ni disponer de un gran presupuesto.

1.3 Objetivos

Aprovechando la comunidad *open source*, y aplicando los conocimientos transversales que se han ido adquiriendo a lo largo del grado, se opta por usar Arduino para obtener un prototipo funcional de un robot aspirador. Se elige este producto por ser un aparato tecnológico muy demandado actualmente y se comparará con los modelos que hay en el mercado.

El principal objetivo de este proyecto consiste en diseñar un prototipo que tenga un funcionamiento aceptable, usando solo productos *open source* en la medida de lo posible.

Para cumplir con este objetivo, se han definido otras subtareas que se detallan a continuación:

- Construir un prototipo que pueda evitar obstáculos de forma autónoma.
- Incorporar un módulo de aspiración para aspirar mientras avanza por la habitación.
- Tratar de desarrollar un prototipo usando solo productos *open source* que se puedan encontrar fácilmente por internet.
- Poder controlarlo con un mando externo.
- Conectar el prototipo a un módulo wifi que permita activarlo desde fuera del domicilio.
- Tratar de obtener un producto lo más ecológico posible. Para ello deberá tener una batería recargable y consumir lo mínimo posible.

1.4 Estructura del documento

Capítulo 1- Introducción: contiene una breve introducción del proyecto desarrollado, explicando la motivación y los objetivos que se pretenden conseguir. También se nombran los diversos apartados de este documento

Capítulo 2- Estado del arte: se analiza la situación actual del entorno en el que se ubica este proyecto, así como las diferentes alternativas de las tecnologías disponibles para llevar a cabo este proyecto.

Capítulo 3- Análisis del sistema: se detallan las especificaciones que debe tener el prototipo y se determinan las opciones que se usarán para llevarlo a cabo.

Capítulo 4- Diseño del sistema: se detalla paso a paso todos los prototipos elaborados con sus problemas hasta llegar al prototipo definitivo.

Capítulo 5- Pruebas experimentales y de validación: se detallan las pruebas realizadas para comprobar si el funcionamiento del prototipo es correcto.

Capítulo 6- Gestión del proyecto: se realiza un análisis de los recursos empleados para llevar a cabo el proyecto. Por un lado, se muestra la planificación en el tiempo de cada una de las fases del proyecto a través de un diagrama de Gantt. Por otro lado, se desglosa el coste de todos los materiales usados en cada uno de los prototipos. También se analizan los costes de mano de obra y de las herramientas empleadas.

Capítulo 7- Entorno socioeconómico: se analizan los modelos más vendidos actualmente en el mercado y se extraen algunas conclusiones.

Capítulo 8- Marco regulador: se realiza una revisión de la normativa a nivel europeo en el entorno *open source: open software* y *open hardware* y con algunos aspectos relacionados con los robots aspiradores comerciales.

Capítulo 9- Conclusiones: presenta un repaso general al proyecto, analizando y comparando los resultados con los requisitos planteados en el *Capítulo 1: Introducción*. Además, se plantean posibles futuros trabajos de mejora.

Capítulo 10- Glosario: se presenta un glosario de siglas para facilitar la lectura del proyecto.

Capítulo 11- Bibliografía: se exponen las fuentes de las que se ha obtenido la información y el conocimiento necesario para la elaboración de este proyecto.

Capítulo 12- Anexos: se muestra el código para la Arduino Mega correspondiente al prototipo V2.0. También se incluye el esquema electrónico final ampliado de los prototipos V1.3 y V2.0

2. ESTADO DEL ARTE

En este apartado se abordan las dos cuestiones principales del presente proyecto. Por un lado, se expondrá la situación actual de los robots aspiradores disponibles en el mercado como contexto del problema planteado en este proyecto.

Por otro lado, se desarrollarán y se explicarán las distintas tecnologías usadas para llevar a cabo este trabajo.

2.1 Robots aspiradores

Son un producto estrella en el mercado actual por la facilidad y comodidad que proporcionan a sus compradores. La mayor parte de las personas tienen un ritmo de trabajo que da lugar a muy poco tiempo libre, por lo que muchas se apoyan en una serie de aparatos tecnológicos que sirven para simplificar las tareas en el hogar.

Un ejemplo son los robots aspiradores. Estos aparatos permiten limpiar de una forma rápida, eficiente y sobre todo autónoma superficies grandes con una baja necesidad de mantenimiento y supervisión.

2.1.1 Criterios de evaluación de robots aspiradores

A la hora de elegir qué robot aspirador comprar, hay distintas cuestiones que se deben tener en cuenta. En general, es importante considerar que las características dependen en gran medida de su precio, por lo que nos centraremos en analizar las dos grandes categorías externas:

1. Robots de gama baja: robots *low cost* con pocas prestaciones tecnológicas.
2. Robots de gama alta: robots de categoría superior con un precio mayor y nuevas prestaciones más sofisticadas que los encarecen.

Aunque existe un gran segmento que corresponde a robots de gama media, en general suelen presentar una combinación de características existentes en las otras dos gamas, por lo que no se tendrán en cuenta en esta exposición.

Teniendo en cuenta estos dos tipos de categorías de robots aspiradores, se pueden enumerar los siguientes criterios de evaluación:

- Grado de autonomía: que el robot pueda sortear todos los obstáculos (cables, muebles, puertas…) y avanzar y aspirar en el máximo número de superficies posibles (suelo, alfombras, moquetas…).
    - Robots de gama baja: suelen tener un sistema básico de detección de obstáculos y motores menos potentes por lo que es más probable que se queden atascados o que haya zonas en las que no puedan aspirar. Esto implica que el usuario deba apartar los posibles obstáculos para que no se

quede atrapado y que en caso de que lo esté deba desplazarlo manualmente a otro sitio.

  - Robots de gama alta: debido a las mejores técnicas de detección de obstáculos y a unos motores más potentes son capaces de aspirar más zonas de forma autónoma. Este tipo de robots es probable que no se quede atrapado y que pueda aspirar toda la habitación por su cuenta sin necesitar intervención humana.

- Calidad de la limpieza: en el caso de que no haya tenido ningún problema para recorrer una superficie, determina si el aspirado ha sido óptimo o no.
  - Robots de gama baja: en general tienen motores de succión menos potentes que dan lugar a un peor aspirado de la zona, teniendo que pasar más veces por ella.
  - Robots de gama alta: debido a los potentes motores son capaces de succionar toda la suciedad sin excesivas dificultades. Son más eficientes puesto que necesitan una sola pasada, economizando tiempo y batería.

- Capacidad de buscar la base de carga: muchos de los robots actuales son capaces de volver por si mismos a la base de carga para recargarse cuando su batería se agota.
  - Robots de gama baja: no tienen la capacidad de volver por si mismos a la base de carga. Esto requiere un mayor grado de supervisión al ser necesario que su dueño esté atento de cuánta batería le queda y de llevarlo a recargar.
  - Robots de gama alta: pueden volver por si mismos a la base. Con dejar su base de carga en el suelo en un punto accesible, pueden ir a recargarse cuando la carga de la batería sea reducida. En general saben dónde han dejado de aspirar y después de la recarga retoman el aspirado en ese mismo punto.

- Duración de la batería: es un parámetro que depende del modelo y de la marca del robot. Es importante tenerlo en cuenta en base a la superficie total que se desea limpiar.
  - Robots de gama baja: como en general no tienen la capacidad de volver solos a la base de carga y luego al punto exacto donde han dejado de limpiar es posible que se queden sin batería y sin una persona que los lleve la base de carga no pueden seguir aspirando el resto de zonas.

    - Robots de gama alta: como pueden retomar la limpieza en el mismo punto después de la recarga es menos probable que dejen zonas sin limpiar.

- Navegación: podríamos definir el rendimiento de un robot aspirador como la capacidad para limpiar una zona por completo, con la menor trayectoria y en el menor tiempo posible. Es decir, se pretende disminuir las zonas por las que pasa más de una vez, para lo que es imprescindible una buena técnica de navegación [3].
    - Robots de gama baja: suelen emplear algoritmos de navegación aleatoria, por lo que no se puede asegurar que limpien toda la superficie por igual. Es decir, puede haber zonas donde ha pasado más de una vez y otras donde no ha llegado a pasado.
    - Robots de gama alta: emplean técnicas de navegación elaboradas que les permiten pasar por todas las zonas al menos una vez e incluso más en caso de que sea necesario una limpieza a fondo.

- Facilidad de mantenimiento: es un parámetro fundamental puesto que todos los robots aspiradores, sean de los modelos más caros o de los más baratos, necesitan un mantenimiento periódico para vaciar el depósito de residuos y cambiar los filtros. Por tanto, las distintas empresas fabricantes han desarrollado técnicas que permiten realizar fácilmente dichas tareas.

- Programación: es uno de los principales parámetros para que estos robots tengan un amplio grado de autonomía.
    - Robots de gama baja: algunas marcas están incluyendo mandos de infrarrojos que permiten programarlos para que se enciendan, limpien y vuelvan a la base de carga todos los días a cierta hora.
    - Robots de gama alta: todos tienen alguna forma de poder programarlos tanto con un mando de infrarrojos como una App específica para el móvil. A través de esta App algunas marcas ofrecen la posibilidad de avisar de los mantenimientos necesarios o de mostrar un mapa de la superficie con las zonas de mayor suciedad o de mayor dificultad para el robot por el número de obstáculos.

Se ha presentado una lista de los principales parámetros que se deben tener en cuenta para comprar un robot aspirador. Es importante saber qué es lo que se desea para poder sopesar y elegir que gama se va a comprar. A continuación, se exponen algunas de las características mencionadas.

2.1.2 Técnicas de navegación

Actualmente, los algoritmos SLAM (*Simultaneous Location and Mapping*) son uno de los métodos más usados para conocer con precisión la posición y la trayectoria de los robots. Casi todos los robots aspiradores de últimas generaciones usan estos algoritmos para saber con certeza en qué posición de la habitación se encuentran, por dónde han pasado ya y por dónde deben pasar para una limpieza más rápida y eficiente. Se basan en crear y actualizar un mapa de un robot móvil, gracias a un sistema de medida, mientras éste se va desplazando.

Hay distintos tipos de sistemas de medida como [4]:

- Láser: son muy precisos y eficientes, aunque pueden tener problemas para detectar ventanas y otros objetos de cristal. El precio suele ser muy elevado.

- Sónar: eran muy usados hasta hace poco puesto que son baratos. Como se pueden usar bajo el agua, puede ser interesante para algunos robots autónomos como los limpia fondos de piscina. Por otro lado, son menos precisos al emitir haces mucho más amplios que el haz de un láser.

- Cámaras: es lo que más se parece al ojo humano. Proporciona una gran cantidad de información que hay que procesar, lo que puede generar un cuello de botella. Esto se va solucionando según aumenta la potencia computacional de los procesadores. Uno de sus principales inconvenientes es que no funcionan en la oscuridad.

2.1.3 Aspiración

La mayor parte de los robots aspiran de una forma muy similar. Como se muestra en la ilustración 1 tienen unos cepillos laterales que giran y empujan la suciedad hacia la parte inferior, donde otros cepillos rotatorios (agitador) desplazan la suciedad hasta su interior. Dentro del robot hay un motor de aspiración que se encarga de succionar la suciedad y el polvo queda almacenado en el depósito.

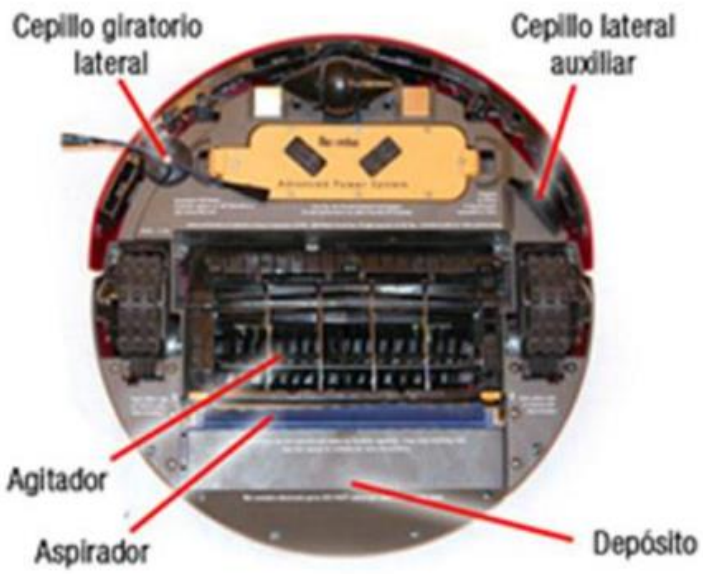


Ilustración 1 Ejemplo de la parte inferior de un robot aspirador

Fuente: https://www.andorobots.com/blog/como-funciona-un-robot-aspirador-sistema-de-limpieza

En la siguiente imagen (Ilustración 2) de una patente de una Roomba se ven los cepillos laterales marcados con el número 76 y los cepillos rotarios marcados con los números 92 y 94.

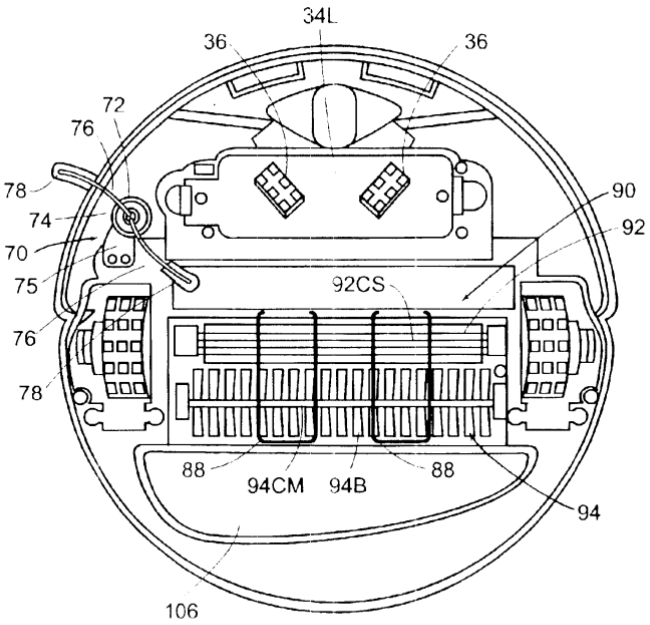


FIG. 2A

Ilustración 2 Esquema de la estructura de una Roomba según la patente.

Cepillos laterales: 76, cepillos rotarios: 92 y 94

Fuente https://www.google.com/patents/US6883201

2.1.4 Principales marcas

Actualmente, hay numerosas empresas que se dedican al diseño y fabricación de robots aspiradores, por lo que la competitividad ha aumentado y los precios han ido disminuyendo mucho, creándose todo un mercado para los compradores.

2.1.4.1 iRobot

Fue de las primeras marcas en crear este tipo de robots aspiradoras y sigue teniendo un gran éxito en el mercado. Esta marca fue creada en 1990 por el MIT (*Massachusetts Institute of Technology)* y buscaba acercar los robots a la gente e intentar hacer sus vidas más sencillas. En el año 2002 lanza al mercado el primer robot aspirador para "ofrecer un dispositivo de limpieza que opere sin intervención humana para limpiar las áreas designadas" [5].

A lo largo de estos 15 años, ha seguido invirtiendo en I+D llegando a lanzar distintas series de robots aspiradoras [6]:

- Serie 400 y Serie 500: fueron las primeras. Actualmente se consideran obsoletas y sólo se venden ciertos recambios.

- Serie 600: es la serie más económica a día de hoy puesto que no tiene ningún extra destacable. Los últimos modelos de esta serie como el 650 ya incluye la posibilidad de programarlo para que limpie todos los días a una hora determinada. Una de las estrategias comerciales de la marca, es mantener estos modelos en el mercado. Aunque son tecnológicamente inferiores comparándolos con las últimas versiones, su precio es bajo y de esta forma iRobot puede competir contra otras empresas que apuestan por modelos *low cost*.

- Serie 700 (2011): todos los modelos de esta serie incluyen la programación y un mando de infrarrojos. También tienen filtros más sofisticados, motores de succión más potentes y una nueva gama de sensores (los *Dirt Detect 2*).

- Serie 800 (2013): consigue una mayor limpieza, mayor rendimiento y ofrece mayor facilidad para el mantenimiento.

- Serie 900 (2015): además de mayor potencia de succión y una gran mejoría en la duración de la batería, incluye un sistema de navegación con cámara y una App para el móvil que proporciona mayor facilidad para programarlo. Los modelos más recientes son de esta serie.

Además de la gama Roomba tienen otros robots de limpieza como los Scooba que pueden fregar el suelo o la gama de Braava que lleva mopa incorporada. Sin embargo, no hay ninguno que combine varias cosas y que pueda aspirar y fregar, por ejemplo.

iRobot tiene una serie de complementos en todas las series que se venden actualmente en el mercado como la base de carga a la que vuelven de forma autónoma o los *virtual wall*.

Estos son barreras generadas con infrarrojos (IR) que se colocan en puertas o zonas donde no se desea que pase el robot.

Actualmente los distintos modelos de Roombas siguen siendo de los más vendidos y puesto que fueron los originales, se van a comentar de una forma más detallada.

2.1.4.1.1 Sensores Roomba

Al igual que los seres humanos tenemos los sentidos, una de las principales características de los robots es que necesitan sensores para interactuar con el entorno. En general los sensores que emplean dan un tipo de información muy limitado. Es decir, como si el oído humano solo pudiese escuchar una frecuencia o el ojo ver un solo color.

La complejidad radica en tener que combinar toda la información de cada uno de los sensores y elaborar una respuesta instantánea de que es lo que debería hacer en cada instante el robot, actualizando los datos de los sensores según se vaya desplazando.

Como se ve en la siguiente figura (Ilustración 3), la mayor parte de los sensores están situados en la parte frontal, ya que la Roomba suele avanzar de frente.

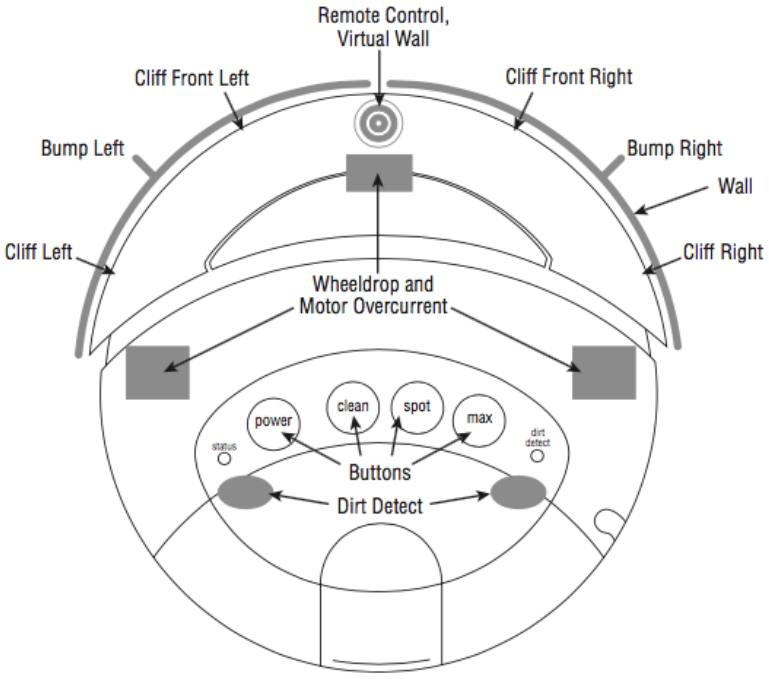


Ilustración 3 Sensores Roomba

Fuente: Libro "Hacking Roomba" de T. E. Kurt

Casi todos los sensores son infrarrojos (cada uno con un par de emisor y receptor) debido a que así se minimiza el desgaste y el rozamiento típico de sensores de contacto y aumenta su vida útil. Sin embargo, suelen ser más caros y complejos de controlar.

En [7] se presentan los sensores ópticos habituales de los Roomba clasificados en tres categorías:

1) Acoplador óptico (*optoisolator*, representado en la Ilustración 4a): de forma general, un acoplador óptico consiste en un LED, en este caso de infrarrojos, como emisor y un fototransistor como receptor. Si usamos un acoplador óptico cuando se aplica una corriente al LED éste emite un haz que llega hasta el receptor que lo convierte en una señal eléctrica. Sirve para comunicar circuitos que deben estar aislados por emplear tensiones de alimentación diferentes. En el caso de las Roomba se utiliza para detectar y localizar la base de carga. Para ello habrá un emisor en la base y el receptor estará situado en la parte frontal del robot.

2) Interruptores ópticos (representado en la Ilustración 4 b): se caracterizan por interponer una barrera física entre el emisor y el receptor. Hay de dos clases:

    - En las ruedas: hay un disco dentado situado entre el emisor y el receptor, de forma que contando los pulsos que recibe el receptor se puede saber los grados de rotación, lo cuál facilita conocer la posición y el movimiento que ha seguido la Roomba.
    - En el parachoques (*bump sensor*): cuando hay una colisión se mueve la barrera y corta el haz entre el emisor y el receptor de forma que la Roomba detecta que ha colisionado con un obstáculo.

3) Detector óptico de objetos (representado en la Ilustración 4c) (*Cliff and Wall detection*): el emisor no apunta directamente al receptor, sino que la luz del emisor rebota cuando hay un objeto próximo y es recibida por el receptor. Por ello se puede emplear tanto para detectar obstáculos, como la ausencia de suelo en el caso de un escalón (*Cliff Sensor*).

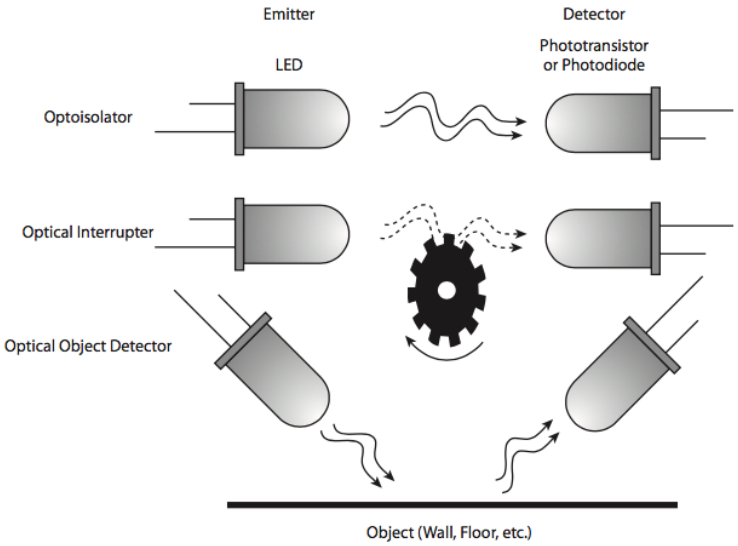


Ilustración 4 Configuraciones típicas de emisor/receptor ópticos (a, b, c)

Fuente: Libro "Hacking Roomba" de T. E. Kurt

Otros sensores que tienen menos uso y, por tanto, al tener menos desgaste no tienen por qué ser ópticos son:

1) Micro interruptores (*Wheeldrop and Buttons*): la unidad de las ruedas empuja el interruptor cuando la rueda está elevada y libera el interruptor cuando baja. Así se puede conocer la posición de las ruedas en caso de coincidir con algún desnivel.

2) Detectores de suciedad *(Dirt detector sensors)*: son sensores piezoeléctricos que generan tensión cuando muchas partículas golpean contra él en zonas de mucha suciedad. Esto permite a la Roomba que determine si debe volver a aspirar por una zona.

2.1.4.1.2 Sistemas de navegación

Al contrario de lo que sucede con los modelos actuales, los primeros modelos de iRobot limpian en recorridos aleatorios, moviéndose por las habitaciones y esquivando obstáculos. Primero empiezan a moverse en espiral hasta que el *bumper* detecta un objeto o una pared. Entonces siguen la pared con su *wall sensor* y van limpiándola, hasta que llega a la esquina con otra pared. Ahí giran noventa grados y prosiguen en línea recta a ras de la pared o hacen otra espiral si no tienen posibilidad de seguir la nueva pared debido a algún obstáculo [8].

Por el contrario, los nuevos modelos como el 980 de Roomba son mucho más eficientes puesto que son capaces de trazar mapas de las habitaciones e incluso trasmiten un mapa a la App para móvil. Esta nueva generación, emplea algoritmos SLAM (*Simultaneous Localization and Mapping*) gracias a la cámara que tienen incorporada [9]. Esto tiene un gran potencial a la hora de limpiar, puesto que le permite asegurar que sólo pase una vez por cada punto. Además, puede interrumpir la limpieza para ir a recargarse y posteriormente retomarla en el mismo sitio en que lo había dejado.

Esta nueva tecnología también da lugar a posibles problemas de privacidad, porque uno de los términos y condiciones que el usuario debe aceptar antes de empezar a usar el robot es que la marca puede vender la información a terceros. Es un tema polémico que se tratará más a fondo en el *Capítulo 8: Marco Regulador*.

2.1.4.2 Cecotec

Esta empresa valenciana creada en el 2007 se caracteriza por ofrecer productos de una categoría bastante buena a un precio reducido. Tiene una amplia variedad de productos: desde colchones hasta microondas. Dentro de tan amplio catálogo, también venden aspiradoras de diversos tipos: de mano, de vapor, verticales y por supuesto robots aspiradores.

Al contrario que iRobot, Cecotec lleva poco tiempo desarrollando este producto y sólo tiene 5 modelos distintos de robot aspirador bajo la denominación genérica Conga, cada uno con la opción de que sean sólo aspiradoras (modelo *dry*) o también fregonas (modelo *wet*) dependiendo del botón que se pulse.

En orden cronológico los modelos son [10]:

- Conga
- Conga Slim
- Conga Slim 890
- Conga Excellence
- Conga Excellence 990

La principal diferencia entre los modelos es que con cada nueva generación ha ido mejorando su autonomía y su capacidad de succión. Además, ha habido mejorías en su sistema de navegación.

Estos robots aspiradores se caracterizan porque por un precio muy económico ofrecen una aspiración con una buena capacidad de succión combinada con la opción *wet* que permite fregar el suelo. Es importante destacar que no aspira y friega a la vez, sino que hay que conmutar entre el modo aspiración y el modo fregona. Por otro lado, es cierto que carece de la variedad de los extras que tienen los robots aspiradores de otras marcas como aplicaciones para el Smartphone, o un sistema de navegación sofisticado. El Conga Excellence 990, aun siendo el mejor modelo de la marca, es comparable con los modelos de la serie 600 de iRobot, es decir, cumplen de sobra con su función de aspiración y tienen un precio muy asequible, pero no son demasiado eficientes.

2.1.4.3 ILife

Esta marca surge en China en torno al año 2007 y empieza a vender robots aspiradores en 2012. Sus robots son similares a los de Cecotec, puesto que tienen unas especificaciones y un precio equivalente. Tiene dos series que incluyen diversos modelos [11]:

- Serie A: han sido diseñadas para una limpieza profunda, sobre todo en alfombras. Para ello cuentan con una enorme capacidad de succión capaz incluso de eliminar alérgenos, y llevan ruedas especiales para poder cambiar con facilidad de superficie y circular rápidamente por las alfombras.

- Serie V: han sido diseñados para suelos de superficie dura. Además, tienen un perfil bajo para poder pasar por debajo de muebles. Algunos modelos como el V5 S Pro incluyen la función de fregado.

Son modelos que generan poco ruido y que tienen una autonomía elevada. Son de los mejor valorados actualmente en webs como Amazon debido a su buena relación calidad-precio. Al igual que los de Cecotec, no tienen un gran sistema de navegación, sino que funcionan de forma aleatoria en las habitaciones. La principal desventaja de este tipo de robots es que son menos eficientes y puede que dejen zonas sin aspirar, pero, por otro lado, el precio es mucho más asequible y cumplen en gran medida con su función de aspirado.

2.1.4.4 LG

Esta marca coreana creada en 1958 empezó a fabricar la gama "Hombot" en 2012. Los Hombots son robots aspiradores muy característicos, con una gran diferencia frente a sus competidores puesto que tienen un diseño cuadrado que les permite aspirar mejor las esquinas [12].

Respecto a su sistema de navegación, las nuevas generaciones tienen una cámara en su parte superior que toma 30 imágenes por segundo, generando automáticamente información para saber dónde se encuentra y poder trazar un mapa. Combinando una serie de algoritmos SLAM (*Simultaneous Localization and Mapping*) con las imágenes de la cámara permite una aspiración eficiente.

Además, las últimas versiones disponen de un modo que permite usar el robot aspirador como cámara y como sistema de video vigilancia. Esto da la opción al usuario de ver en tiempo real lo que ocurre en su casa.

2.1.4.5 Dyson

El británico James Dyson revolucionó el mercado en 1993 con una aspiradora innovadora que no necesitaba bolsas de recambio. Gracias a la singularidad de este producto su empresa despegó, llegando a ser una gran multinacional actualmente.

Hoy por hoy, dentro de su lista de productos, cuenta únicamente con un robot aspirador: el Dyson 360 eye. Es uno de los más potentes del mercado y en vez de ruedas tiene "ruedas de oruga" (como los tanques militares) que le permite cambiar de una forma más eficiente de superficie [13]. Además, cuenta con una cámara que le permite hacer fotos a toda la habitación y mejorar la ruta de aspiración.

Su cámara *Live Vision* permite capturar 30 imágenes por segundo de forma que el robot puede crear y actualizar una representación de su entorno. Esto, combinado con algoritmos SLAM permite triangular la posición del robot en la habitación donde está aspirando, creando un patrón de recorrido más eficiente.

En conclusión, Dyson tiene un único robot aspirador, pero de una gama muy alta. Con su tecnología de succión puede eliminar toda la suciedad pasando solo una vez por cada punto y con las bandas de tracción que tiene puede superar obstáculos como alfombras o cables con mayor facilidad.

2.1.4.6 Neato

Esta empresa americana compite de una forma más reciente en el mercado de los robots aspiradores con diversos modelos. El modelo crea un mapa de las viviendas que se puede ver en una App en el móvil y permite seleccionar desde la pantalla zonas que no se desean limpiar o que se prefiere que el robot evite [14].

Su sistema de navegación se basa en un láser combinado con algoritmos SLAM. El láser se encuentra en la parte superior del robot, y va girando para obtener un escaneo de distancias de la habitación de 360º incluso cuando esté en la oscuridad.

Gracias a estos algoritmos el robot puede crear y actualizar un mapa de la vivienda según va aspirando.

2.1.4.7 Comparativa de los mejores modelos de cada marca

A continuación, se presenta un análisis del mejor modelo de cada marca, representado en la Tabla 1.

Tabla 1 Tabla comparativa del mejor modelo de cada marca

Ordenados por precio ascendente.

	Conga Excellence 990 [15]	**ILife A6 [16]**	**LG Hombot Turbo serie 12 [17]**	**Neato Botvac D7 Connected [18]**	**Roomba 980 [19]**	**Dyson 360 [20]**
Año	2017	2017	2016	2017	2015	2014
Precio Amazon (€)	219	249,99	799	n.d. 899 en su web	958,65	1215,42
Potencia de succión (W)	25	22	40	50	33	n.d
Autonomía de la batería (min)	130	160	100	120	120	80
Peso (Kg)	4,7	4,9	3	3,5	3,94	2,45
Voltaje (V)	14,8	n.d.	14,4	14,4	14,4	n.d.
Vuelve solo a la base	Si	Si	Si	Si	Si	Si
Programable	Mando	Mando	App móvil	App móvil	App móvil	App móvil
Altura (cm)	12	7	8,9	10	9	11
Cámara	No	No	Si	No, láser	Si	Si

Se observa una diferencia enorme de precio entre los modelos comparados en función de la marca. Es cierto que las marcas más caras incluyen ciertas mejoras respecto a las más baratas, sobre todo en términos de sistemas de navegación y eficiencia a la hora de limpiar. Estos modelos disponen de cámaras o sistemas similares que les permiten trazar un patrón de limpieza acorde a diversos parámetros teniendo un funcionamiento óptimo. Por otro lado, los modelos de otras marcas más baratas como la Conga o la iLife A6 funcionan en base a patrones aleatorios lo que los hace menos eficientes.

Además, todos ellos se pueden controlar a través de un mando o de una App para móvil. En el caso de las marcas más caras, disponen de la opción de usar el móvil para programarlos directamente a través de una red Wifi y proporcionan unas estadísticas de limpieza llegando a avisar del mantenimiento que requieren.

Por otro lado, no podemos olvidar que hay una diferencia de unos 700 euros entre las marcas más caras y las más baratas. Si lo único que se desea es un robot que aspire la casa

sin mayores prestaciones y sin importar que tarde más tiempo, las segundas son opciones mucho más rentables que las primeras.

Otro tema a tener en cuenta es el peso, ya que las más pesadas son más costosas de trasladar de una habitación a otra. También conviene reflexionar sobre la altura puesto que lo ideal es que se puedan meter por debajo de sofás y muebles, pero no siempre es posible en el caso de tener muebles más bajos que el robot aspirador.

2.2 Componentes electrónicos de control

En este apartado se comentan las principales opciones disponibles para implementar sistemas de control mediante componentes *open source*.

2.2.1 Arduino vs Raspberry Pi

De cara a realizar el prototipo para el proyecto, es necesario evaluar las principales diferencias entre dos de los grandes competidores del mercado *open source*: Arduino y Raspberry Pi [21]:

- Arduino: son placas basadas en microcontroladores. Éstos son circuitos integrados programables formados por un núcleo que se encarga de ejecutar los programas cargados previamente en la memoria y una serie de unidades de gestión de periféricos.
  Por lo tanto, son necesarios actuadores y sensores que conectar al microcontrolador que realizará diversas acciones dependiendo del programa cargado previamente en su memoria por el usuario. Tienen muchas limitaciones de memoria, velocidad y versatilidad en cuanto a operaciones complejas. Sin embargo, son muy útiles para interactuar con el exterior a través de actuadores o sensores.

- Raspberry Pi: son placas basadas en un microprocesador. Tienen su propio Sistema Operativo y una mayor capacidad para tratar datos y hacer cálculos, pero también con una mayor limitación de conexión con sensores.

Tabla 2 Comparación Arduino Uno vs Raspberry Pi 3 Modelo B

	Arduino Uno (Original) [22]	**Raspberry Pi 3 Modelo B [23]**
Precio Amazon España (€)	20,72 [24]	34,40 [25]
Microprocesador	ATmega328	ARM Cortex-A53 de 64 bits
Pines digitales	14	40 GPIO
Pines analógicos	6	0
Memoria	SRAM 2Kb + EEPROM 1 Kb	RAM 1 GB + Micro SD
Velocidad	16 MHz	1200 MHz

Puesto que se quiere realizar un prototipo con un entorno *open source* con una gran variedad de periféricos sin una necesidad excesiva de potencia de cálculo, se determina usar Arduino.

A continuación, se analizarán las placas Arduino alternativas y algunas opciones de sensores que ofrecen. Para evitar duplicidad en la exposición de la variedad de sensores compatibles con este entorno, esta parte se tratará en el *Capítulo 3 Análisis del Sistema*.

Arduino es el nombre de una compañía basada en el *open source* que diseña, produce y vende una serie de microcontroladores, ofreciendo un software gratuito para programarlo.

Es una herramienta popular para el desarrollo de productos de IoT (*Internet of Things*) y para la enseñanza de conocimientos STEM (*Science, Engineering; Technology and Mathematics*), ya que ayuda a los estudiantes y a personas sin demasiado conocimiento en programación de microcontroladores a crear prototipos, y a un precio muy asequible.

Debido a la alta demanda, muchas empresas se han introducido en este mercado de forma que se han creado numerosos estándares y periféricos que se pueden conectar gracias a los pines multiplexados (cada uno se puede reprogramar para desempeñar distintas funciones).

El software gratuito que usan se llama Arduino IDE (*Integrated Developed Environment*) y se puede instalar en los principales sistemas operativos: Windows, Macintosh OSX y Linux. Además, se ha vuelto muy popular puesto que usa librerías derivadas del lenguaje C++. Esto facilita la programación ya que no es necesario tener conocimiento detallado de cómo programar microcontroladores, sino que basta con tener conocimientos mínimos de programación.

2.2.1.1 Variantes de Arduino analizadas

A continuación, en las Tablas 3 y 4 se mostrará una comparación de las tres placas de Arduino analizadas para la realización de este proyecto.

Tabla 3 Especificaciones técnicas: Arduino Uno, Arduino Mega y Arduino Nano

Resaltadas en negrita las ventajas de la Arduino Mega

Nombre Arduino	**Uno [22]**	**Nano [26]**	**Mega [27]**
Microcontrolador	ATmega328	ATmega328	ATmega2560
Tensión operativa	5 V	5 V	5 V
Tensión de entrada recomendada	7 - 20 V	7 - 20 V	7 - 20 V
Límites de tensión	6 - 20 V	6 - 20 V	6 - 20 V
Corriente DC por pin I/O	40 mA	40 mA	40 mA
Memoria flash	32 KB	32 KB	**256 KB**
SRAM	2 KB	2 KB	**8 KB**
EEPROM	1 KB	1 KB	**4 KB**
Velocidad de reloj	16 MHz	16 MHz	16 MHz

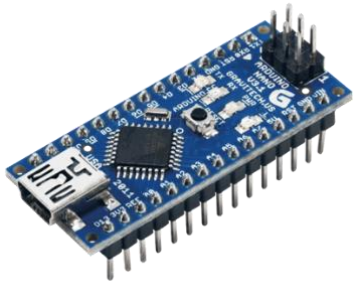
Tabla 4 Números de pines de cada tipo: Arduino Uno, Arduino Mega y Arduino Nano

Resaltadas en negrita las ventajas de la Arduino MegaTipos de pines	**Uno**	**Nano**	**Mega**
Pines digitales I/O (con PWM)	14 (6)	14 (6)	**54 (14)**
Pines analógicos	6	6	**16**
Interrupciones externas	2 (2 y 3)	2 (2 y 3)	**6 (2, 3,18,19,20 y 21)**
Two-Wire Interface /Inter-Integrated circuit (TWI/I2C)	2 (A4 y A5)	2 (A4 y A5)	2 (20 y 21)

En la ilustración 5 se muestra una imagen de cada una de las placas.

Arduino Uno

Arduino Nano



Arduino Mega

Ilustración 5 Comparación de imágenes entre Arduino Uno, Nano y Mega

Por lo tanto, el Arduino Uno y el Nano tienen las mismas características técnicas, pero en un tamaño mucho menor en el segundo caso. Por otro lado, el Arduino Mega dispone de mejores prestaciones tecnológicas y de un mayor número de pines.

2.2.1.2 Tipos de Pines en los Arduino

Las tres placas de Arduino empleadas para este proyecto tienen distintas características, tamaño y número de pines. Sin embargo, los tipos de pines son iguales en los tres casos:

- Pines digitales: pueden usarse como entrada para leer sensores o como salida para enviar información a los actuadores. Toman de valor lógico 0 o 1. Además, pueden actuar con otras funciones especiales como:
    - Pines de PWM: los Arduino usan esta técnica para generar salidas analógicas, puesto que no tienen conversor digital analógico.
    - Pines de interrupción: se pueden configurar para iniciar una interrupción en la rutina del microcontrolador.
- Pines analógicos (de entrada): usan un conversor analógico digital y permiten leer sensores analógicos.

2.2.1.3 Programación en Arduino

El software libre que ofrece el entorno de desarrollo de Arduino (Arduino IDE) se puede instalar en diversos sistemas operativos y permite elaborar programas teniendo conocimientos elementales de C/ C++.

En los programas en lenguaje Arduino debería haber siempre al menos dos funciones:

- Una primera función (*set up*): permite identificar y configurar los diversos pines conectados como *outputs* o *inputs*. Sólo se ejecuta una vez.
- La segunda función (*loop*): actúa como rutina principal y es llamada después de ejecutarse la función de *set up*. Contiene el código que se repetirá de forma secuencial en un ciclo infinito.
  En el caso de estar vacía, el Arduino ejecutaría una vez lo que haya en la función *set up* y luego pararía. Esto puede ser útil si se desea ejecutar el código una única vez y no de forma permanente.

De cara a facilitar la programación de los elementos electrónicos existen algunas facilidades que permiten la interacción con diversos periféricos. A continuación, se detallan algunas de estas facilidades teniendo en cuenta los periféricos que se han usado en este proyecto, en concreto: PWM, comunicación I2C, Interrupciones y Conversión A/D.

2.2.1.3.1 *Pulse-Width Modulation* (PWM)

Dado que la mayoría de los Arduino no disponen de conversor Digital a Analógico, es necesario recurrir a una técnica alternativa para generar salidas analógicas. Esta técnica es la denominada PWM (*Pulse-Width Modulation* = modulación por ancho de pulso).

Dado que las salidas son digitales y sólo pueden adoptar valores lógicos altos (5V) o bajos (0V), la técnica PWM se basa en generar una alternancia entre ambos valores. De esta forma se logra crear ondas cuadradas donde llamamos ancho de pulso al tiempo en el que la onda está en 5V.

Esto es especialmente útil de cara a controlar la velocidad de un motor de corriente continua a través de un Arduino ya que cambiando el ancho del pulso se puede variar la corriente que aportamos al motor y por tanto la energía que recibe. A mayor energía recibida mayor velocidad de giro tendrá el motor.

El *duty cycle* o ciclo de trabajo representado como porcentaje nos permite determinar el tiempo que el pulso está en el estado activo durante un ciclo, como se ve en la ilustración 6.

De esta forma podremos controlar la velocidad de un motor a través de un pin digital con PWM del Arduino. Si aumentamos el *duty cycle* a un 100% la señal estaría siempre activa y el motor girará a máxima velocidad y si lo bajamos a un 0% la señal estaría siempre a 0V y el motor estará parado. [28].

Para aplicar esta técnica es necesario usar uno de los pines con la función PWM combinado con la función *analogWrite(pinNumber,number)*, donde *number* puede ir de 0 (*duty cycle* de 0%) a 255 (*duty cycle* de 100%).

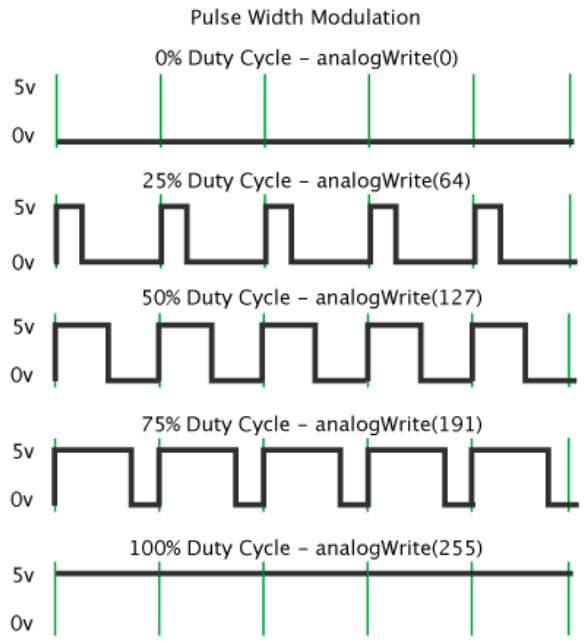


Ilustración 6 PWM con distintos Ciclos de Carga (Duty Cycle).
Fuente https://www.arduino.cc/en/Tutorial/PWM

2.2.1.3.2 Comunicación entre Arduinos (I2C)

El I2C es un protocolo síncrono que permite conectar varios Arduino usando solo dos cables. Uno para el reloj (pines SCL) y otro para los datos (pines SDA). Se basa en una configuración maestro-esclavo en la que el dispositivo maestro controla el cable por el que circulan los datos a los dispositivos esclavos [29].

El maestro genera la señal del reloj e inicia la transferencia de datos, indicando al esclavo si desea recibir información de éste o enviarle algo. Todo esto se pude realizar a través de los pines A4 y A5 (en el caso del Arduino UNO y el Nano) y gracias a la librería *Wire* del IDE de Arduino.

2.2.1.3.3 Interrupciones

Hay situaciones en las que integrar en un bucle la supervisión de todos los sensores de un dispositivo supone muchas complicaciones. En tales casos, conviene disponer de algún mecanismo capaz de interrumpir el flujo de procesamiento normal (tareas básicas) mediante señales generadas por eventos que requieren de un tratamiento urgente. Por ejemplo, en el caso de colisión del prototipo mientras el Arduino está pendiente de medir distancias con los ultrasonidos se puede llamar a una función que detenga el robot y que permita esquivar el obstáculo.

Cada placa de Arduino dispone de pines de interrupción. A cada uno de ellos se le puede asociar una función que será invocada siempre que suceda algún evento independientemente de la parte del código que esté procesando el microcontrolador. Lo ideal es que esta función sea lo más breve posible, para que el Arduino pueda retomar la tarea principal que quedó interrumpida.

Además, es necesario usar un tipo de variables especiales declaradas *volatile* para indicar al compilador que pueden cambiar de valor en cualquier momento. Esto evita que el compilador asigne una variable a un registro interno del microcontrolador, con lo cuál se operará siempre accediendo a memoria. Si en algún momento una rutina de servicio de interrupción modifica el valor de dicha variable, no habrá inconsistencia. El resto de las variables sí podrán ser asignadas a los registros del microcontrolador al ser más rápido que el acceso a la memoria.

Los Arduino son capaces de detectar los siguientes eventos en los pines de interrupción [30]:

- RISING: si el flanco de subida va de LOW a HIGH
- FALLING: si el flanco de subida va de HIGH a LOW.
- CHANGING: tanto si el flanco va de HIGH a LOW como si va de LOW a HIGH.
- LOW: se detecta si el valor del pin es LOW

2.2.1.3.4 Conversores A/D

Permite a un Arduino leer la información de los sensores conectados en los pines analógicos. Sólo es necesario un único conversor A/D de Arduino ya que a través de un multiplexor se puede atender a varios pines analógicos.

El conversor A/D permite convertir la señal analógica que recibe el pin de entrada en un valor binario que puede ser procesada por el Arduino. Tiene una resolución de 10 bits y una frecuencia de muestreo de 8,928 KHz [31].

2.2.1.4 Sensores, motores y otros componentes para Arduino

Para evitar la duplicidad estos elementos se explicarán en el *Capítulo 3 Análisis del Sistema*

3. ÁNALISIS DEL SISTEMA

El objetivo de este proyecto era diseñar un prototipo funcional de un robot aspirador autónomo capaz de evitar en la medida de lo posible obstáculos y paredes. Todo esto sería diseñado y creado usando productos *open source* en general, y en particular sensores, motores y otros módulos necesarios compatibles con Arduino.

Aunque la eficiencia de un robot aspirador esté directamente relacionada con el sistema de navegación y los algoritmos que usa, debido a motivos de presupuesto y de tiempo, este prototipo avanzará por las habitaciones de forma aleatoria, mientras evita chocarse y quedarse atascado en obstáculos o paredes.

Como es un prototipo, se ha dejado para futuros trabajos el refinado del algoritmo de navegación, así como un diseño propio de un módulo de aspiración extraíble.

Otro requisito importante, es que se pueda alimentar con baterías recargables para promover la sostenibilidad y reducir el costo a largo plazo al no tener que comprar nuevas pilas.

3.1 Elección de las tecnologías para el desarrollo del proyecto

Aquí se analizarán los elementos tecnológicos disponibles y los motivos para elegirlos frente a otras alternativas.

3.1.1 Microcontrolador

Como se ha mencionado en el *Capítulo 2: Estado del Arte*, *Sección 2.2 Componentes electrónicos de control* el microcontrolador seleccionado será el Arduino, debido a la gran flexibilidad que ofrece dentro de los entornos *open source* y a su precio que es bastante reducido. La alternativa con una Raspberry Pi se descarta dado que no es necesaria gran capacidad de cómputo, y Arduino facilita las conexiones con sensores, de lo cuáles hay disponibles una gran variedad.

Respecto a las placas Arduino, se han ido probando en los sucesivos prototipos las tres alternativas mencionadas (Uno, Nano y Mega) y se han evaluado las ventajas e inconvenientes de cada una de ellas para elegir la alternativa óptima en el prototipo final. Esto se expondrá en el *Capítulo 4: Diseño del Sistema*.

3.1.2 Motores

Hay tres tipos principales de motores para Arduino [32]:

- Servos: son motores eléctricos que pueden ser programados para girar un número determinado de grados.
- Motores paso a paso (*stepper*): son motores eléctricos que se mueven un paso (un número determinado de grados) por cada pulso que se les aplique.

- Motores de corriente continua (DC): son motores eléctricos en los que no hay control sobre los grados que giran. Cuando se aplica una tensión mayor gira más rápido, y si se cambia la polaridad gira en sentido contrario. Además, conviene usar transistores para gobernarlos y así evitar sobretensiones en el Arduino.

Tabla 5 Comparación de motores.
Fuente https://www.luisllamas.es/tipos-motores-rotativos-proyectos-arduino/

	Características		Control	
Motor	Velocidad	Par	Posición	Velocidad
DC	Alto	Bajo	Bajo	Bajo
DC con reductora	Medio	Alto	Bajo	Bajo
Servo	Bajo	Alto	Absoluto	Absoluto
Paso a paso	Medio	Medio	Absoluto	Absoluto

Para este proyecto se usarán motores de corriente continua, puesto que no es primordial tener un control absolutamente preciso del giro que realizan. Dado que no se requiere una velocidad alta y sí que es necesario que tenga alto par para poder superar obstáculos como alfombras y desplazar todo su peso, lo más apropiado será usar motores DC con reductora.

Hay numerosos motores de continua con reductora para Arduino, pero todos tienen unas características muy similares. La reductora consiste básicamente en un engranaje con un piñón que mueve a una rueda con un mayor número de dientes, por lo que se reduce la velocidad de giro del eje de la rueda. Por otro lado, sucede lo contrario con el par y este aumenta en el eje de la rueda.

Principales parámetros para elegir un motor DC [33]:

- Par/Torque: es la cantidad de fuerza rotatoria que el motor puede aplicar a una carga. En el caso de este proyecto, tiene que ser lo suficientemente alto para que el robot pueda superar cables y pequeños desniveles sin tener demasiados problemas.
- Velocidad (rpm): es la velocidad angular con la que rota el motor. Para este proyecto, es necesario tener en cuenta algunos datos como por ejemplo la velocidad a la que van los robots aspiradores que hay en el mercado, y así hacerse una idea de la velocidad ideal a la que debería ir.
- Potencia: es necesario encontrar un motor con una potencia aceptable para los requerimientos del prototipo. La potencia depende de la tensión y de la corriente según la siguiente ecuación:

$$P = V * I$$

  - Corriente: aumentar el amperaje es necesario para incrementar el par.

- Tensión: aumentar el voltaje va ligado con incrementar la velocidad de giro del motor (rpm).

3.1.2.1 Motores para Arduino

Se ha seleccionado unos tipos de motor DC con una reductora acorde con la velocidad mínima de una Roomba y con un par lo suficientemente grande para lograr mover los prototipos. Para ello, se ha tenido en cuenta las distintas reductoras que hay en este tipo de motores de continua [de 1:48 y 1:120] de cara a los prototipos iniciales (Ilustración 7a) y de 1:224 para el prototipo final (Ilustración 7b).

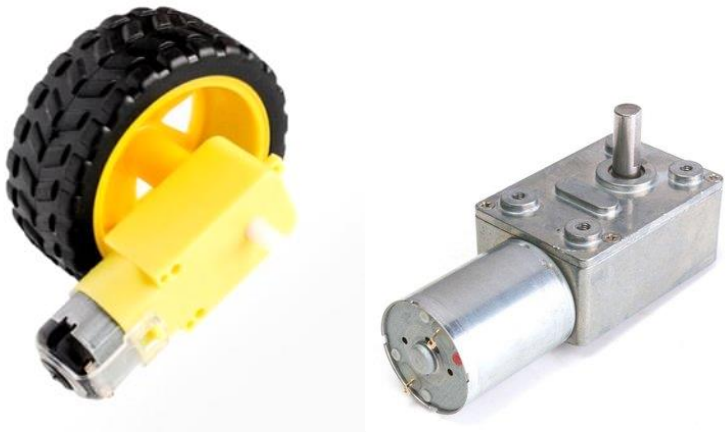


Ilustración 7 Motores con reductora (a) de plástico y (b) de metal

La velocidad de una aspiradora Roomba es entre 10 mm/s y 500 mm/s [7], sin embargo, de cara al prototipo es más importante que logre una velocidad lo suficientemente lenta para que tenga cierto margen de maniobra y pueda evitar colisionar contra los obstáculos.

Dado que las ruedas escogidas tenían un radio R=32,5 mm, podemos calcular la velocidad angular (ω) a la que debería girar el eje del motor de teniendo en cuenta la reductora:

$$v = \omega * R$$

Por lo tanto, necesitamos unas revoluciones por minuto (rpm) de la rueda de:

$$\omega_{Rueda\ min} = \frac{v_{min}}{R} = \frac{10}{32{,}5} = 0{,}307\,\frac{rad}{s} = 2{,}93\ rpm$$

$$\omega_{Rueda\ max} = \frac{v_{max}}{R} = \frac{500}{32{,}5} = 15{,}38\,\frac{rad}{s} = \ 146{,}86\ rpm$$

En el caso de los motores de plástico (Ilustración 7a) la velocidad final del eje dependerá de la reductora. Si el motor lleva una reductora de 48:1, su eje debe girar a:

$$\omega_{Motor\ \ min} = 48 * \omega_{Rueda\ min} = \ 140{,}64\ rpm$$

$$\omega_{Motor\ max} = 48 * \omega_{Rueda\ max} = 7049{,}28\ rpm$$

Si el motor lleva una reductora de 120:1, su eje debe girar a:

$$\omega_{Motor\ min} = 120 * \omega_{Rueda\ min} = 351{,}60\ rpm$$

$$\omega_{Motor\ max} = 120 * \omega_{Rueda\ max} = 17.863{,}2\ rpm$$

Como es un prototipo, se puede asumir que debería ir a la velocidad mínima para que tenga mayor margen de maniobra, aunque sea menos eficiente. De forma que es necesario encontrar un motor que gire a una velocidad en torno a 140 rpm, en el caso de que tenga de reductora 48:1, o de 350 rpm, en el caso de que tenga reductora 120:1.

Por otro lado, los motores de metal (Ilustración 7b) ofrecen en su *datasheet* la velocidad a la que gira el eje del motor por lo que no será necesario realizar cálculos con la reductora (1:224). En este caso se usará un motor que gire a una velocidad en torno a 100 rpm con 12V y sin carga. Por lo tanto, una aproximación de la velocidad máxima a la que avanzaría el prototipo teniendo en cuenta que recibe una tensión de unos 6V es de:

$$v_{max} = 50\ rpm * 32{,}5mm = 5{,}23\frac{rad}{s} * 32{,}5mm = 169{,}97\ mm/s$$

Hay que tener en cuenta otras circunstancias, además de que se trata de una aproximación. Por el hecho de tener que desplazar el peso del prototipo final, los motores girarán necesariamente más despacio. La regulación de la velocidad final se realizará mediante el *PWM*.

3.1.2.2 Motor driver board

Para controlar los motores, es necesario usar un *driver* de potencia. De los modelos estudiados, se utilizará el modelo L298N (puente de H doble) debido a que viene ya ensamblado con disipadores de calor, y conexiones para insertar cómodamente los *inputs* y los *outputs* como se ve en la ilustración 8. Otra posible opción sería el L293 (4 medios puentes H), pero puesto que sólo se van a controlar dos motores DC con el L298N bastará.

Ilustración 8 L298N

El L298N es un circuito integrado formado por dos puentes H (un puente H para controlar cada motor) [34] que permite controlar las dos principales variables de un motor de continua: el sentido y la velocidad de giro. Para lograr el cambio de sentido podemos cambiar la polaridad de los terminales del motor. Esto se puede conseguir gracias a los puentes H que son dispositivos formados por transistores y diodos que pueden controlar la polaridad en función de unas entradas lógicas como se ve en la ilustración 9.

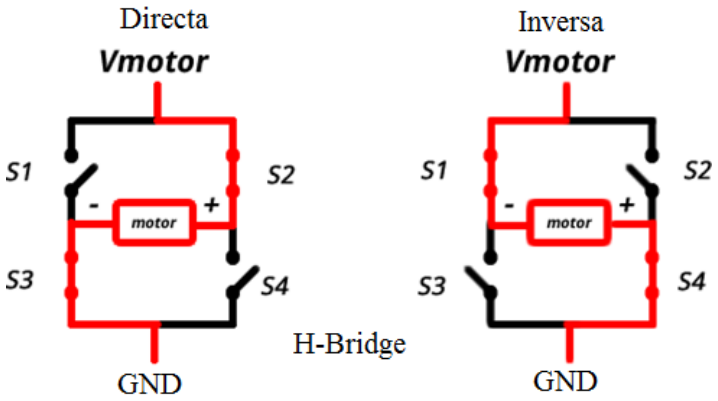


Ilustración 9 Ejemplo de funcionamiento de un puente H

Fuente: https://www.prometec.net/hbridge/

En la tabla 6 se representan los valores lógicos que deben aplicarse en la entrada del driver de potencia para que el robot avance hacia delante, hacia atrás, gire a la derecha o a la izquierda:

Tabla 6 Valores lógicos de entrada al driver de potencia L298N

El Arduino las activará para determinar el sentido de avance

	IN1	IN2	IN3	IN4
Girar derecha	L	H	L	H
Girar izquierda	H	L	H	L
Avanzar	L	H	H	L
Retroceder	H	L	L	H

Con este driver se pueden controlar 2 motores de DC de menos de 2 A y de una tensión de 3V a 35V. Está formado por 4 pines que se encargan del sentido de giro de dos motores. Por ejemplo, por un lado, IN1 e IN2 sirven para controlar el motor A y, por otro lado, IN3 e IN4 para controlar el motor B. También tiene dos entradas conectadas a pines de tipo PWM del Arduino (ENA y ENB) que sirven para controlar la velocidad de los motores [35].

3.1.3 Sensores de Arduino

Antes de empezar con el diseño del prototipo, era necesario investigar sobre los distintos tipos de sensores, *drivers* y motores disponibles y compatibles con Arduino. Los robots que evitan obstáculos suelen usar dos tipos de sensores. Por un lado, están los sensores de colisión por contacto en los que el robot detecta un obstáculo una vez que ha colisionado con él. Por otro lado, están los sensores a distancia con los que el robot recibe una señal en presencia de un objeto o una pared y que le permite evitar el choque [32].

1) Sensores de colisión: el sensor funciona como un interruptor. Si no se produce ningún choque la señal que envía al Arduino es una señal digital de valor 0 y en el momento en el que se produce una colisión se presiona y envía la señal digital de valor 1 (Ilustración 10). Debe conectarse a una entrada digital del Arduino. Son

especialmente útiles al permitir que el robot pare cuando ha colisionado, aunque no evitan los golpes.

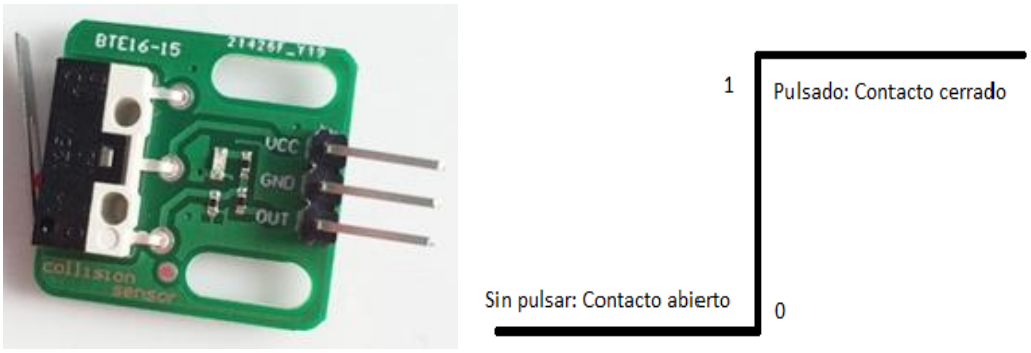


Ilustración 10 (a) Sensor de colisión y (b) ejemplo de señal que genera

2) Sensores a distancia: los sensores activan una señal cuando detectan un objeto y otra distinta en caso contrario, sin necesidad de tocar el objeto. Por lo tanto, son muy útiles para evitar en la medida de lo posible la colisión del robot.

   - Infrarrojos (IR): están formados por un emisor de infrarrojos y un receptor. Hay muchas variedades y se pueden encontrar numerosos modelos compatibles con Arduino. Se comentarán los tres tipos empleados para los prototipos:

     A) Sensores de proximidad de infrarrojos: en este caso el receptor genera una señal digital en nivel bajo cuando ha detectado un obstáculo y en nivel alto si no hay nada en su alcance. Suelen ser de corto alcance, en torno a unos 5 cm.
     El sensor de infrarrojos se compone de un transmisor de luz infrarroja y un receptor capaz de detectar los rayos reflejados en una superficie (Ilustración 11). Contienen también un potenciómetro que permite ajustar el umbral de detección, con lo que es posible regular la distancia de detección dentro del intervalo de las especificaciones del modelo. Se han estudiado modelos de diversos fabricantes (Sunfounder, Elegoo, …), pero casi todos presentaban un funcionamiento insatisfactorio.

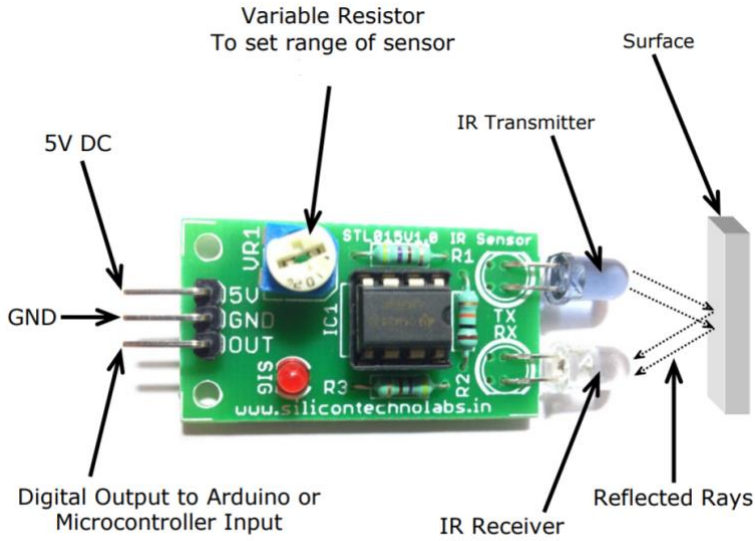


Ilustración 11 Componentes de un sensor de proximidad de infrarrojos.
Fuente: http://silicontechnolabs.in/upload/IR%20Proximity%20Sensor%20datasheet.pdf

B) Sensores Sharp [36]: este tipo de sensores son algo más complejos y miden la distancia exacta de los obstáculos. Son sensores con un *output* analógico (por lo que habrá que conectarlos a una entrada analógica del Arduino) formados por un receptor PSD (*Position Sensitive Sensor*), un diodo emisor de infrarrojos y un circuito procesador de señal. En la ilustración 12 se puede ver representado.

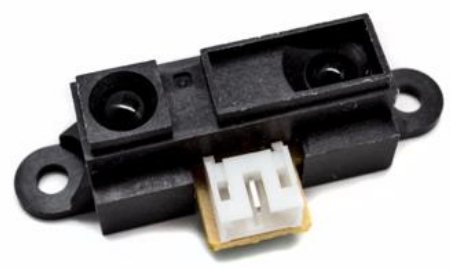


Ilustración 12 Sensor Sharp

Los hay de distinto tipo, siendo clasificados como sigue:

- Corto alcance: detectan obstáculos entre 4 y 30 cm.
- Medio alcance: detectan obstáculos entre 10 y 80 cm.
- Largo alcance: detectan obstáculos entre 20 y 150 cm.

- Ultrasonidos (US): disponen de dos pines digitales, *trigger* que se encarga de disparar el envío de una onda de sonido de una frecuencia determinada y *echo* que entregará una señal lógica al recibir la señal reflejada. Por lo tanto, la onda

sonora sale, rebota en un objeto y vuelve. Calculan la distancia midiendo el tiempo que tarda en regresar. Son relativamente precisos y al igual que los Sharp también se pueden clasificar por el alcance que tienen. Uno de los más comunes para Arduino y el que se usará en este proyecto es el HC-SR04 que se muestra en la ilustración 13. También se plantearon otros modelos como los US-015 y SF-SR02 con un mayor alcance que no era necesario, y un precio más elevado, por lo que no aportaban ventajas de interés respecto al HC-SR04.

Ilustración 13 Sensor de Ultrasonidos HC-SR04

3.1.3.1 Elección de sensores para el prototipo

En principio, los robots aspiradores patentados usan sensores infrarrojos para identificar los objetos y obstáculos que tienen alrededor y poder evitarlos. Sin embargo, la mayor parte de los prototipos de robots que evitan obstáculos se fundamentan en sensores ultrasonidos. Para investigar esta discrepancia y ver cuáles podían ser más útiles para el prototipo, se realizan una serie de pruebas con la placa Arduino y los sensores por separado que se detallarán en el *Capítulo 5: Pruebas experimentales y de validación*. Las conclusiones obtenidas se adelantan a continuación:

- Ultrasonidos HC-SR04: tienen un alcance útil de unos 400 cm [37]. En las pruebas llevadas a cabo se pueden detectar obstáculos hasta una distancia 30 cm de forma óptima. No se siguió ampliando la distancia de la prueba al no considerarse necesario para este proyecto.
- Infrarrojos de proximidad: tienen menor alcance, llegando solo a detectar obstáculos a menos de 4 cm de distancia. Sin embargo, en las especificaciones del fabricante, el rango del sensor es de 3 a 30 cm [38]. Esta inconsistencia entre el funcionamiento teórico y el real de alguno de los componentes *open source* se tratará en el *Capítulo 9: Conclusiones y futuros trabajos*.
- Infrarrojos Sharp: tienen un rango operativo generoso que depende del modelo y una buena precisión. Para este proyecto se usará el Sharp GP2Y0A21YK con un rango útil de 10 cm a 80 cm [39]. En las pruebas llevadas a cabo se detectaron obstáculos hasta unos 40 cm de forma óptima.

En resumen, las elecciones dependerán del problema que deba ser solucionado en cada caso. Se usarán tres sensores de ultrasonidos para detectar obstáculos frontales a distancia puesto que tienen un rendimiento similar a los infrarrojos de Sharp y son mucho más baratos. Por otro lado, se opta por instalar dos sensores de infrarrojos de proximidad para

detectar obstáculos laterales a corta distancia. Otros tres sensores de colisión evitarían que el robot siguiese avanzando en caso de golpe. Por último, se dispondrá de un sensor Sharp como *cliff sensor* debido a su gran precisión y exactitud para detectar y evitar los desniveles.

3.1.4 Alimentación

Existen diversas formas para alimentar las distintas placas de Arduino mencionadas en este proyecto:

- Conectar 5V a la placa:
    - A través del USB conectado al ordenador.
    - A través de una fuente de tensión conectada al pin V_{in}.
- Clavija Jack (no disponible en el Arduino Nano): se han de aplicar entre 6 y 20 V. Aunque se recomienda no aplicar más de 12V por posibles problemas de sobre tensión.
- Aplicar en el caso de un Arduino Nano entre 6 y 20V entre el pin *ground* y el pin RAW. Aunque se recomienda no aplicar más de 12V por posibles problemas de sobre tensión.

Es necesario alimentar el Arduino y el L298N con una fuente de alimentación. Puesto que una de las características de los robots aspiradores es que deben ser capaces de funcionar autónomamente, se opta por incluir baterías en el prototipo.

Es cierto que se podrían haber implementado otras alternativas más ecológicas como placas solares, pero debido a que en general proporcionan una tensión baja y a que los robots aspiradores no suelen estar expuestos a mucha luz solar, se descartó la idea.

De forma que para cumplir con el requisito de ser lo más ecológico posible, se opta por incluir baterías recargables que permitan numerosos usos. Para realizar este prototipo se han usado dos tipos de baterías para determinar las más convenientes:

- Baterías LiPo (Litio y Polímero) de tipo 18650: son baterías recargables que tienen una tensión de 3,7 V cada una, y 3400 mAH. En este proyecto se han usado dos conectadas en serie para disponer de una tensión de unos 7,4 V ya que 7V es la tensión mínima para que funcione correctamente el L298N conectado a los dos motores.
- Baterías de NiMh (Hidruro de Niquel) tipo AA con 1,2V nominales y 1600 mAh: se colocan en un portapilas de 6 unidades en serie para obtener una tensión de unos 7,2V.

Debido a que las baterías de NiMh son más seguras de cara a realizar prototipos, son las que finalmente se eligieron para los últimos prototipos.

3.2 Materiales auxiliares

En este apartado se analizarán los componentes estructurales que se usarán para desarrollar los prototipos.

Hay numerosas opciones posibles para fabricar un chasis lo suficientemente sólido y compacto para agrupar todos los componentes electrónicos y mecánicos:

- Metacrilato: es un plástico duro que viene incluido en numerosos packs de robots que evitan obstáculos. Un ejemplo es el que se ve en la ilustración 14.

Ilustración 14 Chasis de metacrilato

Fuente: http://codeduino.colegionet.es/robots/97-chasis-coche-inteligente-para-arduino.html

- Metal: es una opción más robusta y pesada que el metacrilato. Para evitar un incremento excesivo del peso, una opción es hacer únicamente un esqueleto de metal usando segmentos alargados como las que ofrece la marca Eitech (Ilustración 15). Otras opciones más conocidas son de las marcas: Fischertechnik, Meccano o MakerBeam, que se descartaron debido a su elevado precio. Además, algunas piezas de Lego se han empleado de forma puntual (en el V1.3).

Ilustración 15 Piezas de metal de la marca Eitech para construir estructuras metálicas

Fuente: https://www.amazon.es/Eitech-11-25-piezas-planas-%C3%A1ngulos/dp/B0009GV5AE

- Plástico de una impresora 3D: es la mejor opción ya que ofrece un material robusto y totalmente personalizable que permite diseñar el chasis a medida. No se usará esta opción para este proyecto por no disponer de una impresora 3D, pero es una de las opciones más claras en caso de futuras mejoras del prototipo.

4.DISEÑO DEL SISTEMA

Se ha planteado un desarrollo progresivo e incremental en base a sucesivos prototipos que permitan ir estudiando diversos problemas y resolverlos en el siguiente prototipo.

4.1 Diseño del prototipo preliminar (V1.0 – V1.3)

El objetivo es desarrollar un primer prototipo para estudiar e implementar las distintas tecnologías mencionadas en el *Capítulo 3: Análisis del sistema*. Para comprobar que todo funciona como es debido, se irán realizando diferentes versiones de éste prototipo preliminar, añadiendo de forma sucesiva nuevos componentes. Además, se estudiarán varias alternativas para determinar la combinación de componentes más eficiente.

4.1.1 Prototipo V 1.0

Está formado por un chasis de metacrilato, dos motores de DC con reductora, con ruedas de caucho en sus ejes y una rueda libre delantera. Lleva también dos baterías de LiPo que proporcionan una tensión de 7,4 V, un Arduino Uno y el *driver* de potencia L298N conectado a los dos motores DC. En las ilustraciones 16 y 17 se muestran dos vistas del protipo V1.0.

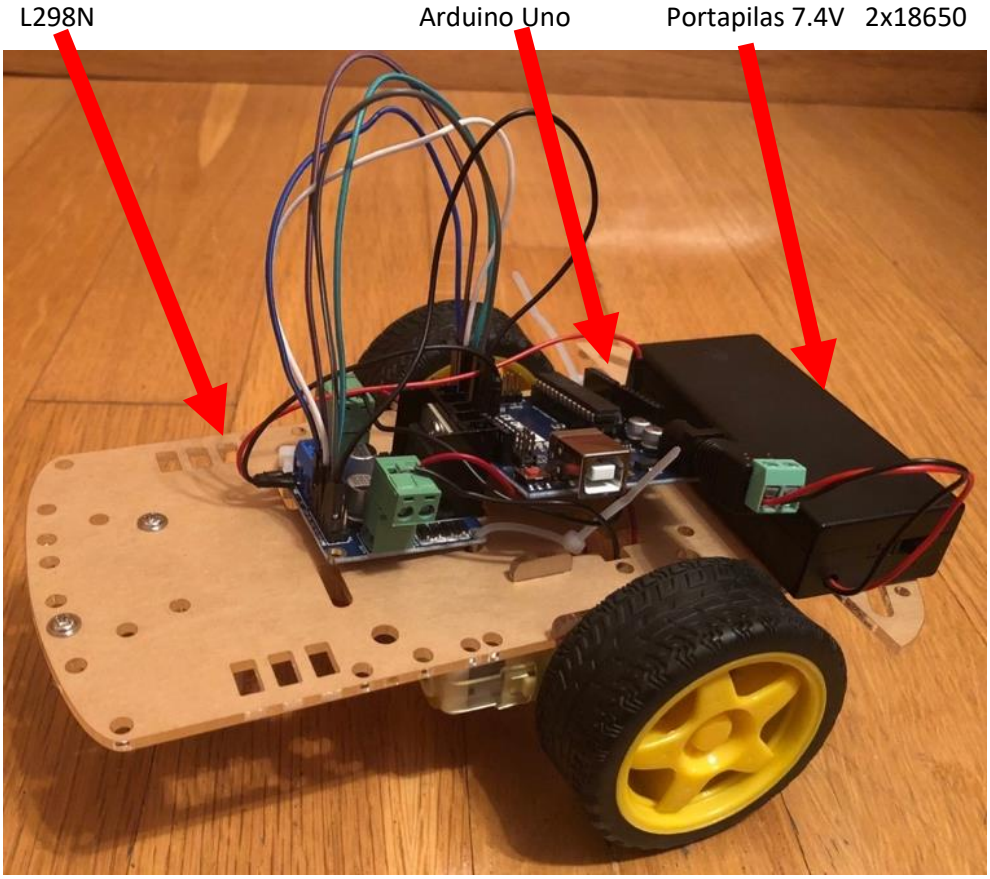


Ilustración 16 Prototipo V 1.0, vista superior

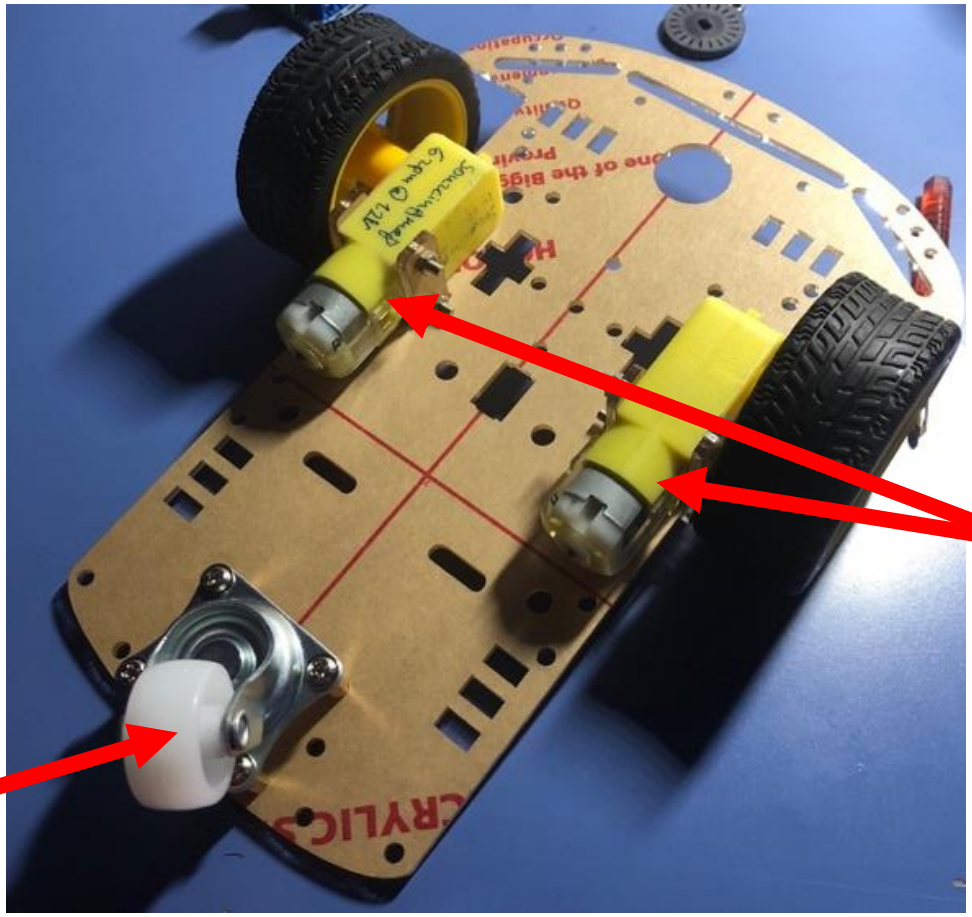


Ilustración 17 Prototipo V 1.0, vista inferior

El esquema electrónico se representa en la ilustración 18.

La alimentación está conectada directamente al L298N puesto que éste necesita una tensión alta para funcionar correctamente y la alternativa de usar el pin de salida de 5 V del Arduino implicaría una tensión insuficiente. Por otro lado, este módulo tiene un pin que proporciona una tensión de salida de 5V que es una tensión adecuada para el Arduino. De esta forma se puede conectar una tensión alta al L298N y obtener de él los 5V necesarios para el Arduino de manera que no se corre peligro de alimentarlo con una sobre tensión.

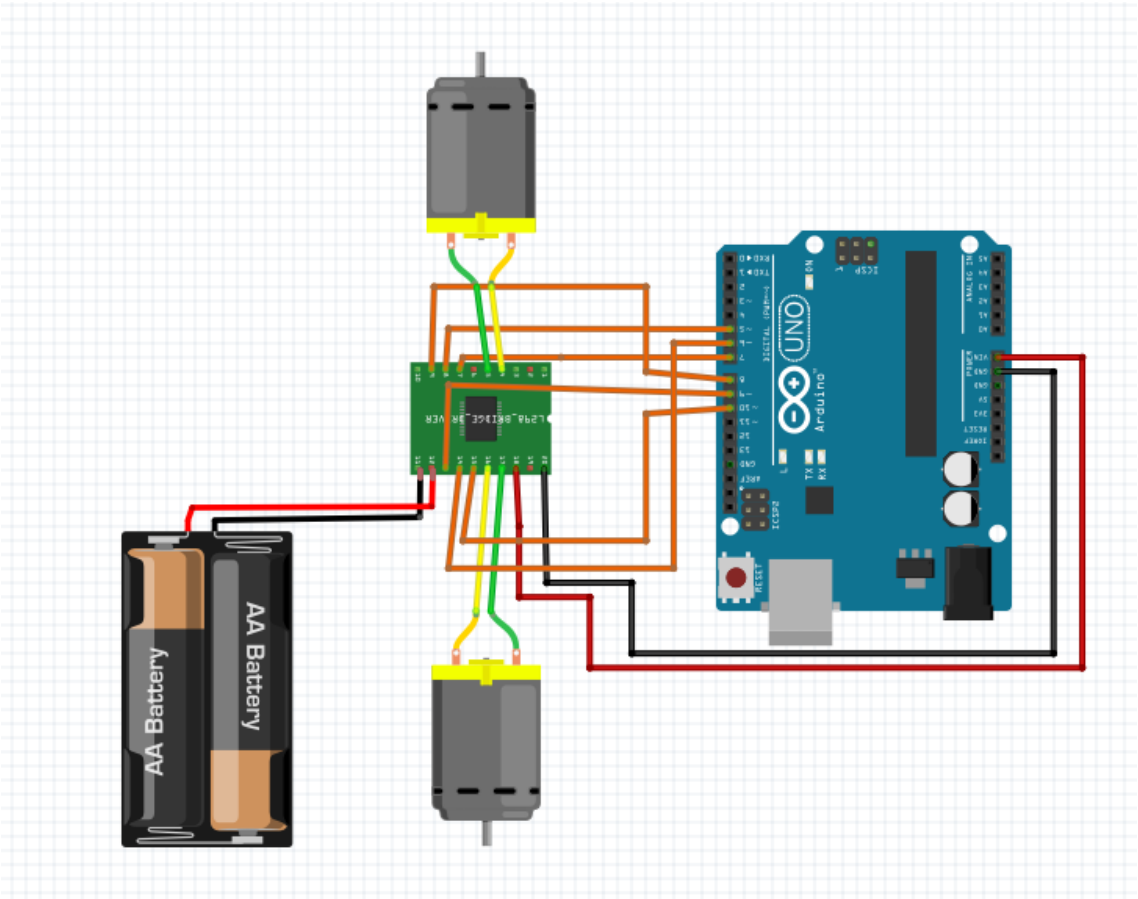


Ilustración 18 Prototipo V1.0, circuito electrónico

En la tabla 7 se representan las conexiones relevantes para el L298N. Las entradas ENI y END tienen que ir conectadas a los pines PWM del Arduino para regular la velocidad de los motores. La placa de Arduino Uno genera una frecuencia de 490 Hz para los pines de PWM excepto en los pines 5 y 6 en los que la frecuencia es de 980 Hz. Esto implica que los pines PWM no se pueden mezclar puesto que si no se estaría aplicando una mayor frecuencia a un motor que al otro lo que conllevaría que una rueda girase más rápido que la otra.

Tabla 7 Correspondencia entre pines del Arduino y pines del L298N

Pin Arduino	L298N
5 (PWM)	ENI
7	IN1
8	IN2
9	IN3
10	IN4
6 (PWM)	END

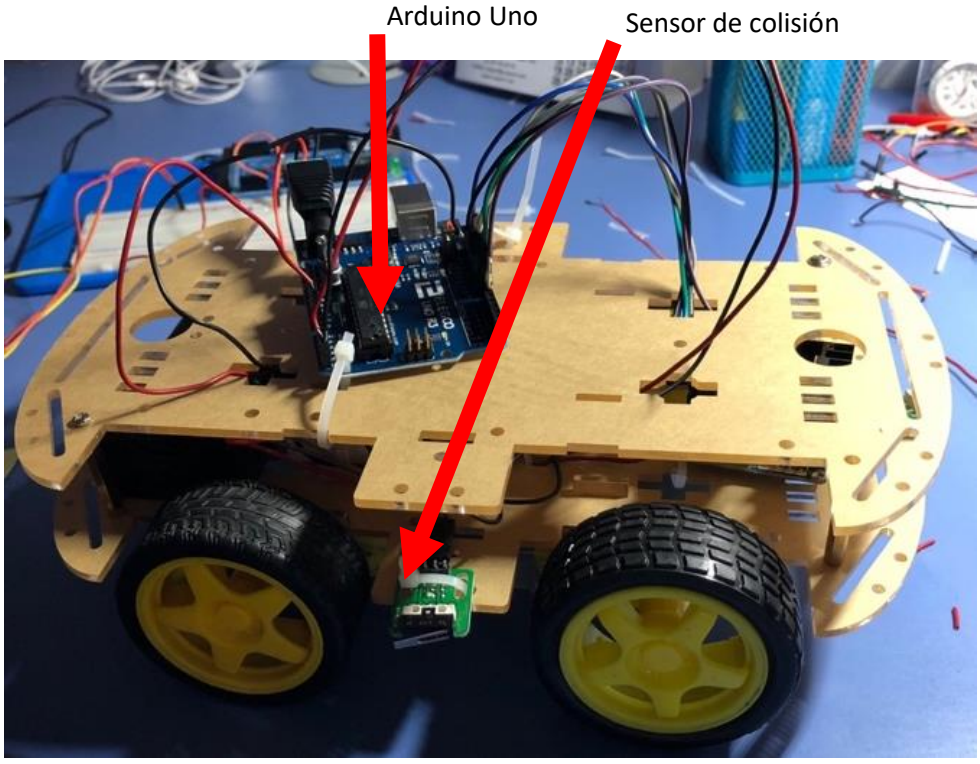
En esta versión, se obtiene un prototipo funcional de robot que se desplaza siempre hacia delante sin capacidad de esquivar obstáculos. Por lo tanto, se concluye con una primera aproximación válida.

4.1.2 Prototipo V1.1

Como el prototipo anterior funcionaba bien, se decide incorporar cuatro motores con sendas ruedas, para aumentar la estabilidad. Además, se incorporan sensores de colisión para que pueda parar en el caso de una colisión y que los motores no sigan girando. Puesto que el número de pines del Arduino Uno es limitado, se opta por ampliar el sistema con un Arduino Nano usando la comunicación I2C entre ambos. En este caso, el Arduino Uno sería el maestro, es decir, el que determina los tiempos y las acciones, y el Arduino Nano sería el esclavo.

Se conectan los tres sensores de colisión al Arduino Nano y se programa de forma que si se activa cualquiera de los tres se envíe una señal al maestro para que éste pare los motores.

Para ampliar el espacio disponible para los dispositivos electrónicos, se usan dos pisos de metacrilato, en el piso superior se coloca el Arduino Uno como se muestra en la ilustración 19.



lustración 19 Prototipo preliminar V1.1, vista piso superior

En el piso de abajo, se emplazan los cuatro motores (Ilustración 20), conectados los dos de la derecha en paralelo al L298N y al igual que los de la izquierda. Esta configuración permitirá que los motores operen de forma sincronizada para realizar giros a derecha y a izquierda, además de avanzar y retroceder.

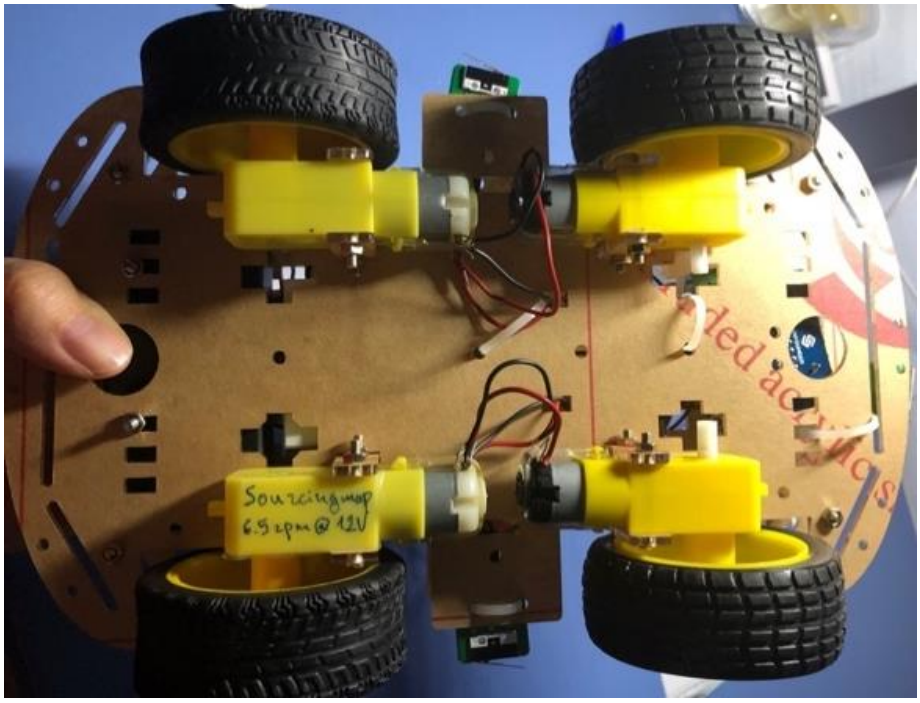


Ilustración 20 Prototipo V1.1, piso inferior, vista inferior

En la parte superior del piso de abajo también se sitúan las baterías, una *mini protoboard* con una regleta de conexión a tierra y otra a 5V, el L298N, y los tres sensores de colisión, junto con el Arduino Nano al que están conectados, como se aprecia en la Ilustración 21.

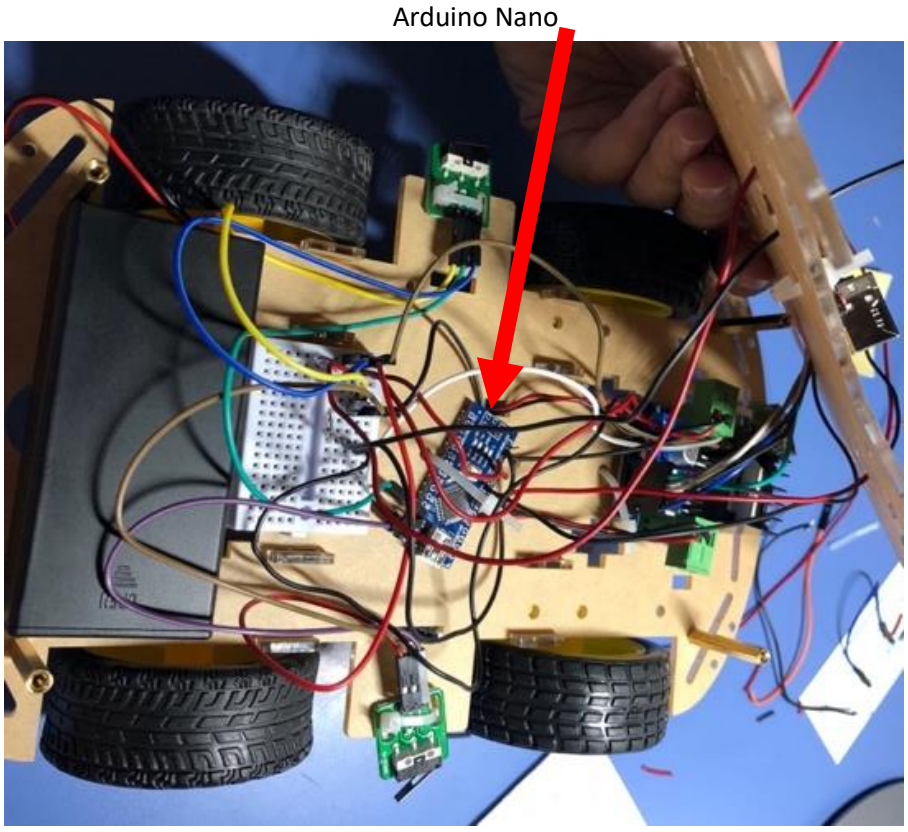


Ilustración 21 Prototipo V1.1, piso inferior, vista superior

El esquema electrónico viene representado en la ilustración 22:

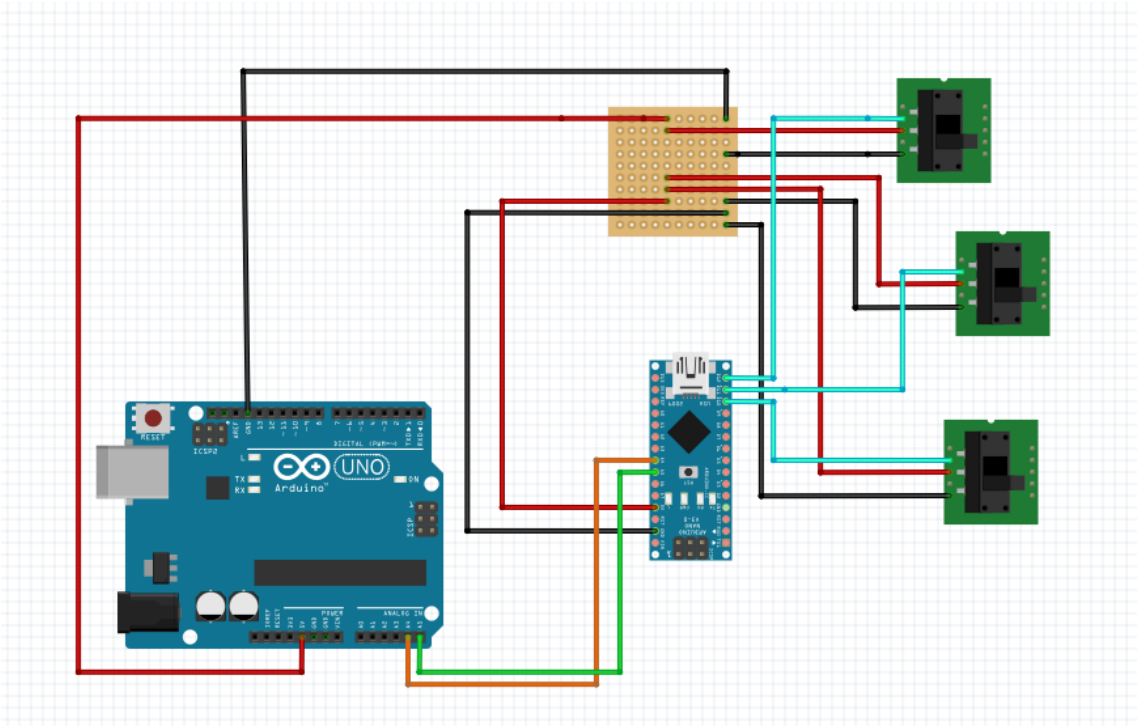


Ilustración 22 Circuito electrónico prototipo V1.1

El esquema de conexiones se representa también en la tabla 8.

Tabla 8 Pines Arduino Nano

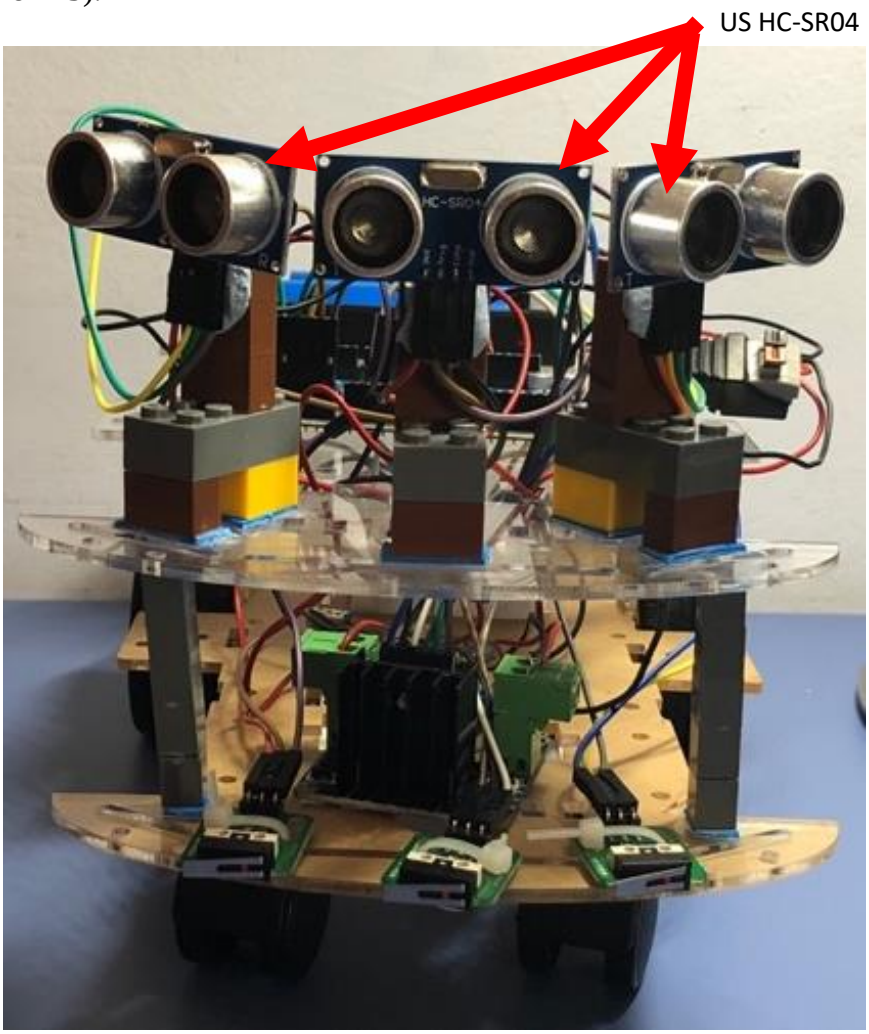
Pin NANO	Componente
D12	Sensor de colisión frontal
D11	Sensor de colisión frontal derecho
D10	Sensor de colisión frontal izquierdo
A4 (SDA)	A4 (SDA) del Arduino Uno (i^2C)
A5 (SCL)	A5 (SCL) del Arduino Uno (i^2C)

Como conclusiones para esta versión, funciona de forma correcta en caso de colisión, pero tiene una capacidad de reacción lenta. Es decir, aunque cumple con su función y en caso de colisión se paran los motores, el tiempo de reacción es demasiado grande debido a los retardos en la comunicación I2C entre los dos Arduinos. Por tanto, se optará por una aproximación distinta en la siguiente versión.

4.1.3 Prototipo V1.2

El siguiente paso consiste en incorporar sensores de detección a distancia que eviten las colisiones en la medida de lo posible.

Un tipo de sensor muy común para Arduino debido a su bajo precio es el sensor de Ultrasonidos HC-SR04. Para usarlo es necesario emplear dos pines digitales del Arduino. Se decide incorporar tres sensores en la parte frontal, dispuestos en ángulos de 45º entre sí (Ilustración 23).



lustración 23 Vista frontal prototipo V1.2

Se realizan dos cambios fundamentales frente al prototipo V1.1.

Por un lado, para optimizar su tiempo de reacción en el caso de colisión, se cambiará la combinación Arduino Nano y Arduino Uno comunicados vía I2C por un Arduino Mega que dispone de muchos más recursos en cuanto a pines y a memoria.

Por otro lado, se usarán solo dos motores de DC, como tienen casi todos los robots aspiradores en el mercado. De esta forma se ahorran dos motores con sus engranajes y su correspondiente consumo de energía. Por lo que tendrá dos ruedas traseras con tracción y dos ruedas delanteras sin motor asociado, que le proporcionan estabilidad (Ilustración 24).

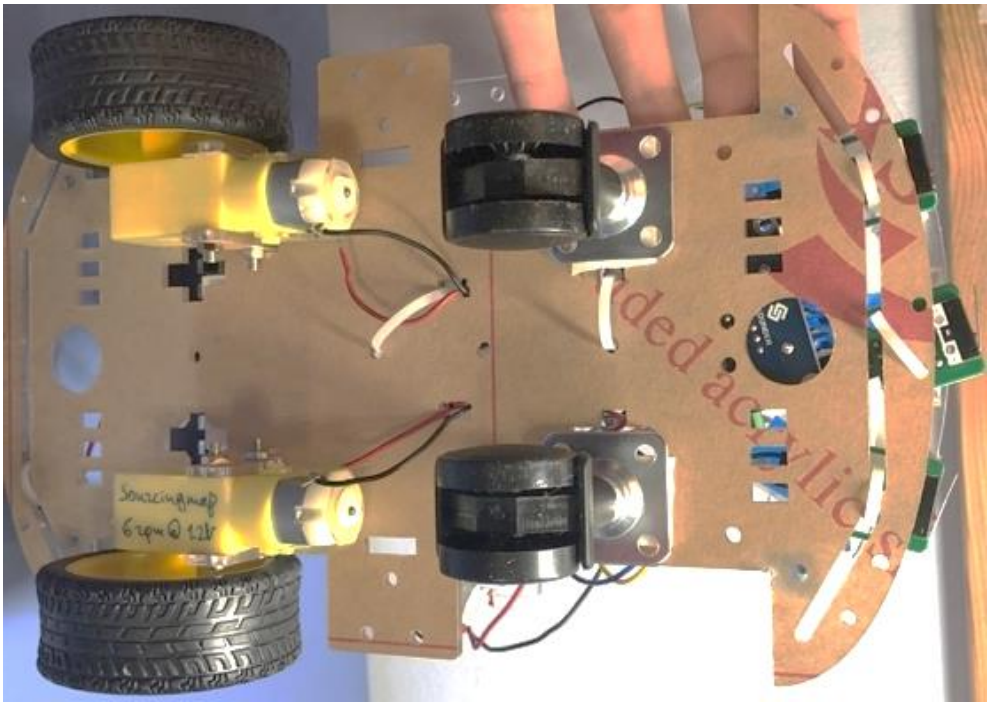


Ilustración 24 Prototipo V1.2, piso de abajo, vista inferior

El otro objetivo era cambiar la conexión I2C dado que induce un ligero retardo, por un único Arduino Mega (Ilustración 25) que dispone de muchos más pines que el Arduino Uno o el Arduino Nano.

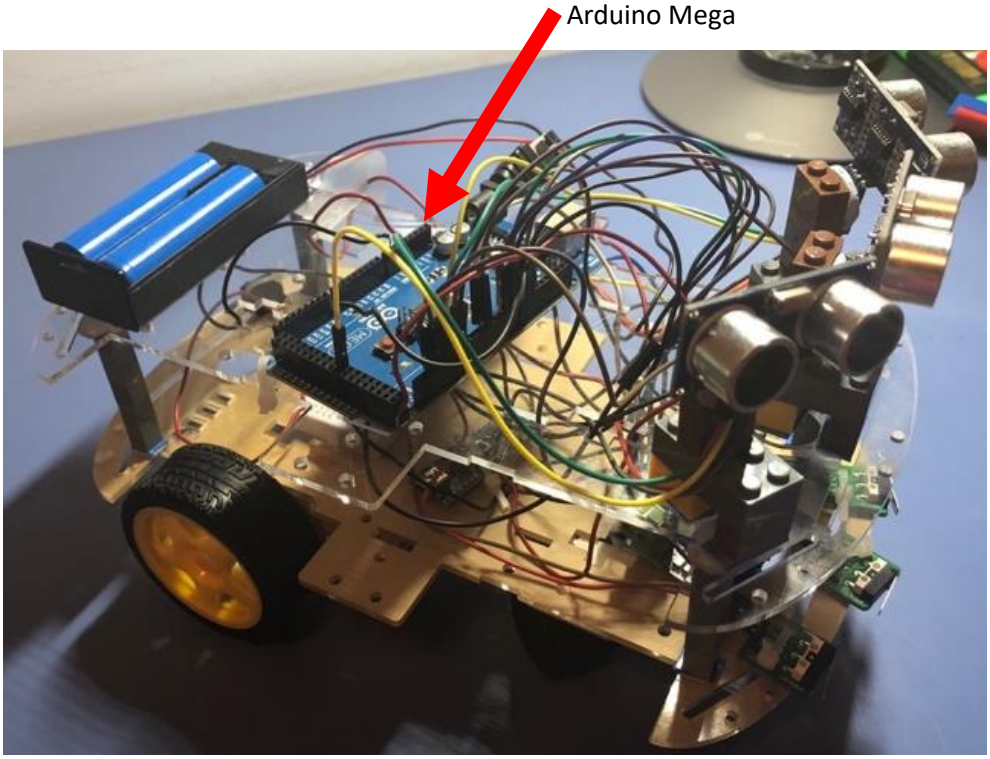


Ilustración 25 Prototipo V1.2, piso de arriba, vista cenital

De esta forma, éste prototipo dispone de un Arduino Mega para controlar los tres sensores de colisión y los tres sensores de ultrasonidos, además de controlar el L298N que se encarga de los dos motores.

Un problema que surge en este caso es que la necesidad de controlar varios sensores de forma simultánea conlleva un programa de control más largo y complejo. Un efecto secundario no deseado es que la señal de colisión se podría perder en los instantes en los que el microcontrolador estuviese pendiente de medir el retardo de los sensores de ultrasonidos, por lo que se decide emplear interrupciones en el Arduino, aprovechando los correspondientes pines de interrupción. El esquema electrónico se representa en la ilustración 26:

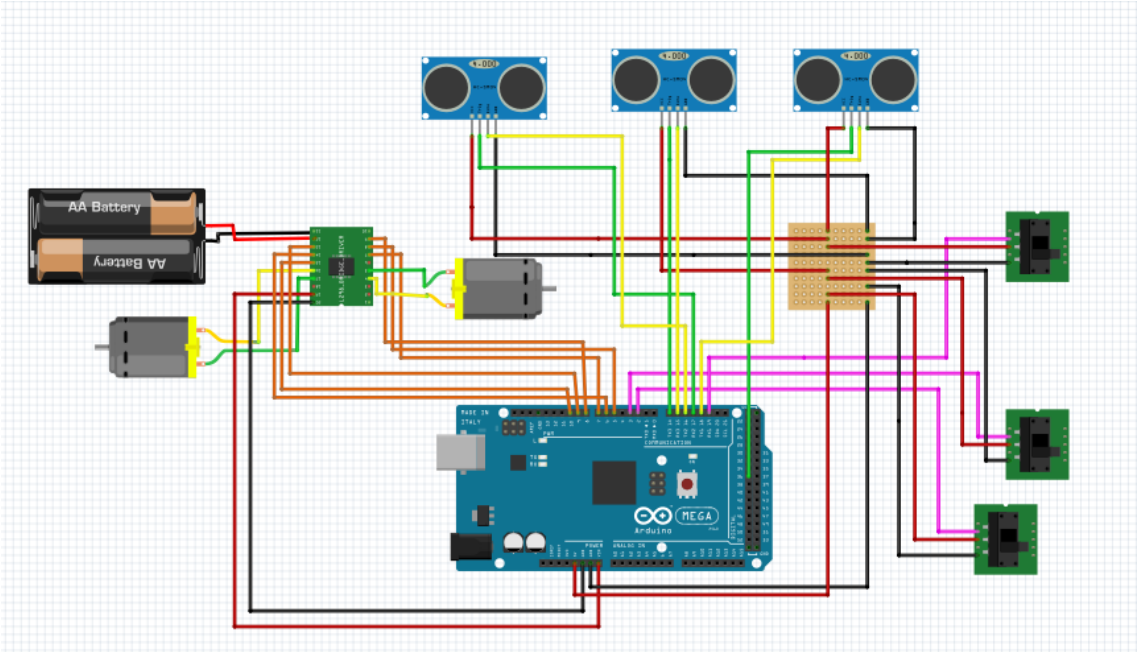


Ilustración 26 Circuito electrónico, prototipo preliminar V1.2

A los pines de la tabla 9 hay que añadir los pines de los sensores de colisión y los pines de los sensores de ultrasonidos que se conectan directamente a pines de la Mega:

Tabla 9 Conexión de los pines entre la Mega y los componentes en prototipo V1.2

Pin Arduino Mega	Componente	
3 (interrupción)	Colisión izquierda	
2 (interrupción)	Colisión central	
19 (interrupción)	Colisión derecha	
17	Echo izquierda	US izquierda
16	Trigger izquierda	
14	Echo centro	US centro
15	Trigger centro	
18	Echo derecha	US derecha
36	Trigger derecha	

En este caso, se usan los pines de interrupción para los sensores de colisión. Estos sensores proporcionan una salida digital con un nivel alto cuando se ha detectado un objeto y se ha presionado el interruptor, y un nivel bajo si no se detecta nada. Por lo tanto, se requiere que el microcontrolador interrumpa sus operaciones cuando detecta un objeto, es decir, cuando el pin pasa de 0 a 1 en un flanco de subida como se muestra en la Ilustración 27. Para ello se utilizará la función:

attachInterrupt(pinDigitalAInterrumpir,función,RISING).

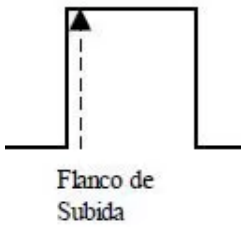


Ilustración 27 Flanco de subida en sensor de colisión

En las pruebas realizadas se concluye que el prototipo funciona de forma correcta, pero tiene ciertos problemas en el caso de obstáculos cercanos en los laterales, ya que no los puede detectar. Este problema se analizará en la siguiente versión.

4.1.4 Prototipo V1.3

Uno de los problemas del prototipo V1.2 consistía en que no detectaba bien los obstáculos laterales que escapan al ángulo de visión de los sensores de ultrasonidos de la izquierda y de la derecha. Por tanto, se decide instalar adicionalmente dos sensores de infrarrojos de proximidad con un alcance de unos 2 cm para detectar obstáculos más cercanos situados fuera del rango de visión de los sensores de ultrasonidos.

El siguiente paso era instalar un *cliff sensor* o un sensor para detectar escalones o desniveles y evitar caídas. Tras evaluar todas las alternativas de sensores se opta por instalar un sensor Sharp de corto alcance. Estos sensores son muy precisos y adecuados para detectar un desnivel y parar el robot, evitando que se caiga por ellos. En la ilustración 28 se puede observar dicho sensor.

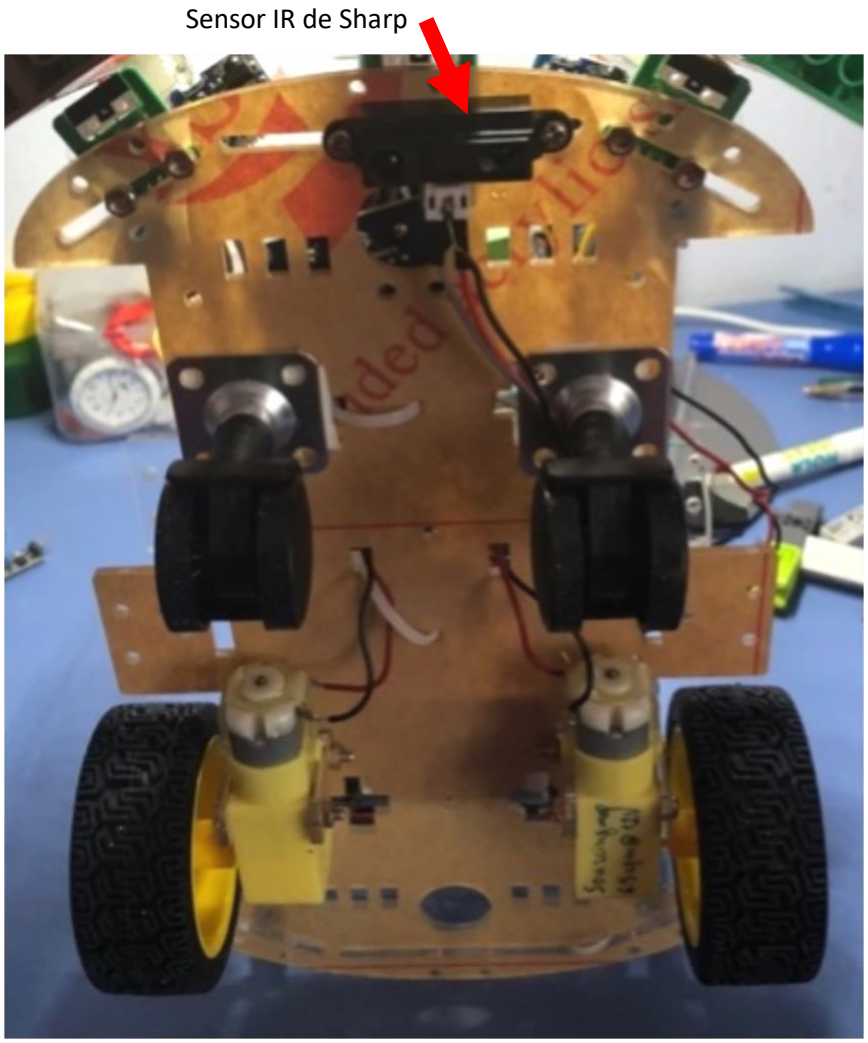


Ilustración 28 Prototipo V1.3, piso de abajo, vista inferior

Además, se quiere mejorar el rendimiento de los sensores de colisión del prototipo V1.2, puesto que solo se activaban si el robot chocaba de una forma muy específica contra los obstáculos, por lo que no estaba garantizado que funcionasen en todas las colisiones. Una forma sencilla de incorporar un parachoques consiste en unir piezas de lego para aumentar el área de detección del impacto, como se puede ver en la ilustración 29.

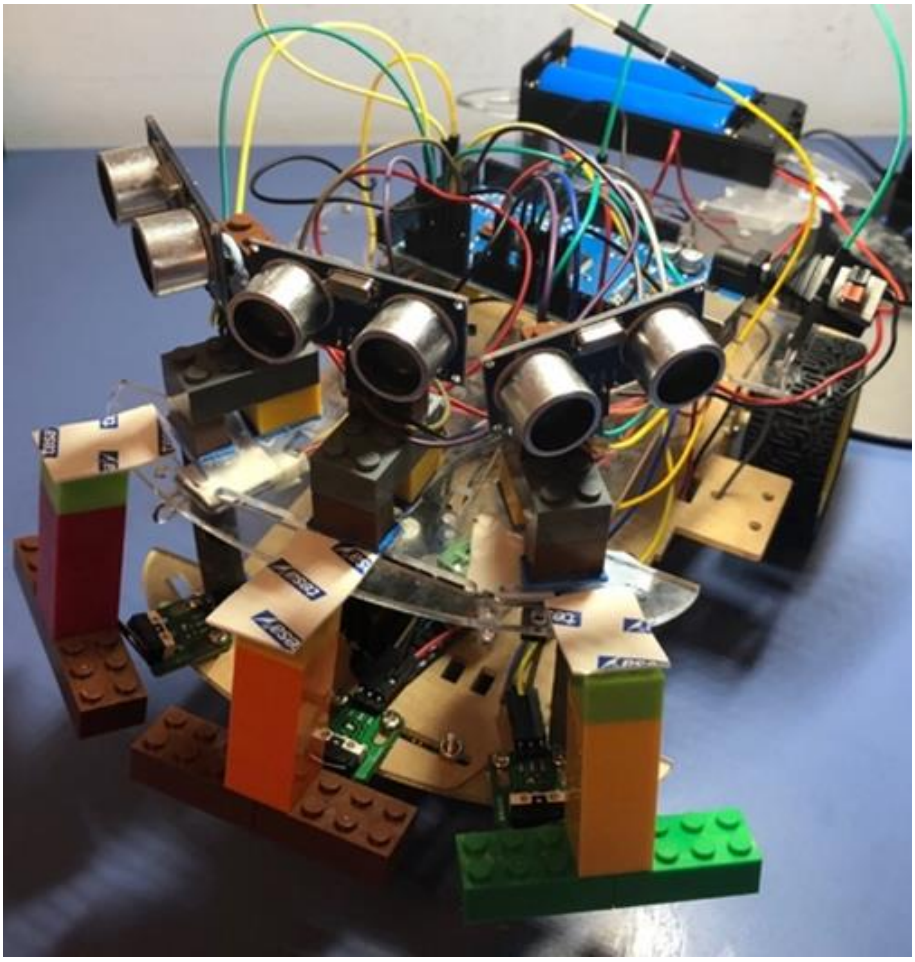


Ilustración 29 Prototipo V1.3 con parachoques

A los pines de la tabla 9 hay que añadir los sensores de proximidad de infrarrojos y el sensor Sharp como se ilustra en la tabla 10.

Tabla 10 Pines Mega y componentes añadidos al prototipo V1.3

Pin Arduino Mega	Componente	
3 (interrupción)	Colisión izquierda	
2 (interrupción)	Colisión central	
19 (interrupción)	Colisión derecha	
17	Echo izquierda	US izquierda
16	Trigger izquierda	
14	Echo centro	US centro
15	Trigger centro	
18	Echo derecha	US derecha
36	Trigger derecha	
42	IR izquierda	
44	IR derecha	
A1	Sharp	

El esquema electrónico se representa en la ilustración 30 que está ampliada en el anexo:

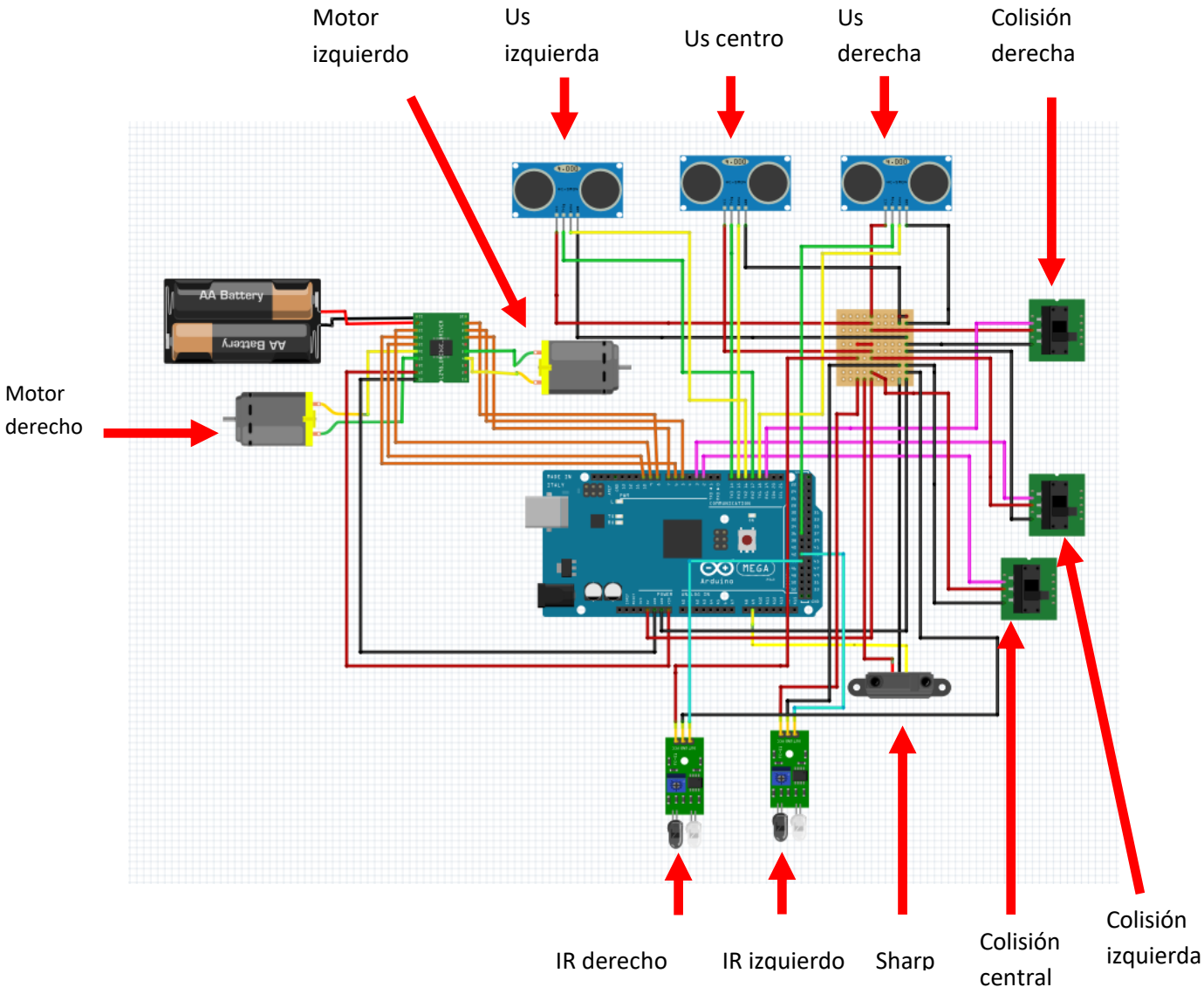


Ilustración 30 Circuito electrónico prototipo V1.3

Antes de pasar al diseño del modelo final del proyecto, se realizan cambios que se pueden apreciar en la ilustración 31:

- Sustituir las baterías LiPo por 6 baterías recargables de NiMh 1,2 V porque son más seguras a la hora de trabajar con ellas en los prototipos.
- Incorporar 3 LEDS en la parte posterior que permitan identificar cuál es la función que está siguiendo el prototipo 3 en cada momento: ir de frente, girar a la izquierda o a la derecha en función de lo que detecten los sensores de ultrasonidos. Esto permite verificar en tiempo real si el programa funciona correctamente.

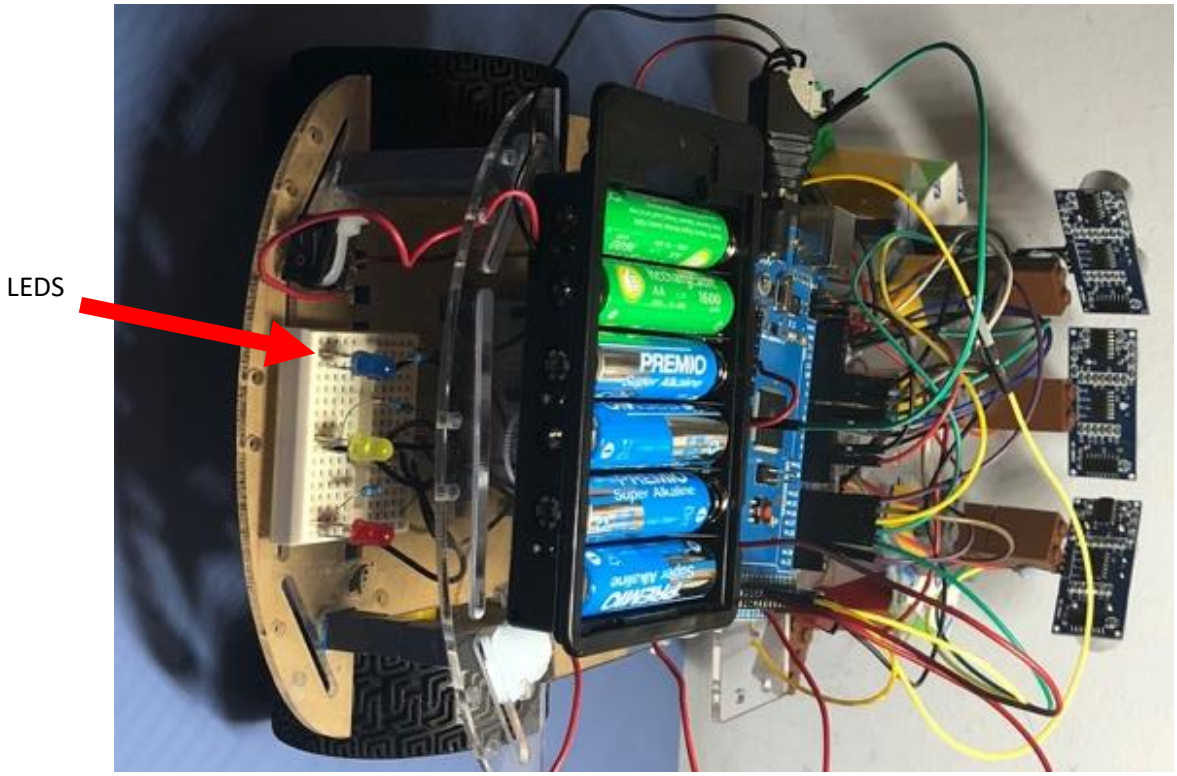


Ilustración 31 Prototipo V1.3 final

El esquema electrónico del prototipo V1.3 se representa en la ilustración 32.

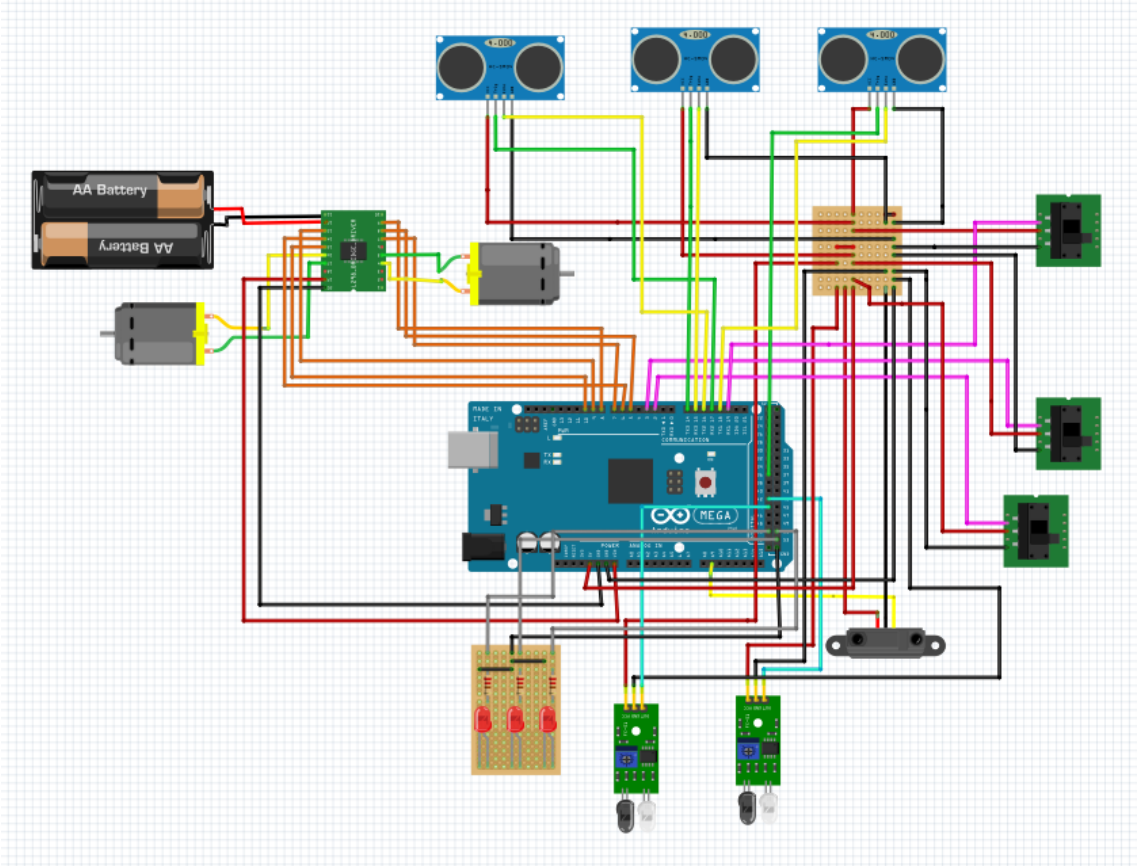


Ilustración 32 Circuito electrónico prototipo V1.3 final

Los pines empleados para los LEDS están incluidos en la tabla 11.

Tabla 11 Pines Mega con sus nuevos componentes añadidos

Pin Arduino Mega	Componente	
3 (interrupción)	Colisión izquierda	
2 (interrupción)	Colisión central	
19 (interrupción)	Colisión derecha	
17	Echo izquierda	US izquierda
16	Trigger izquierda	
14	Echo centro	US centro
15	Trigger centro	
18	Echo derecha	US derecha
36	Trigger derecha	
42	IR izquierda	
A1	Sharp	
44	IR derecha	
A1	Sharp	
51	LED izquierda	
53	LED centro	
52	LED derecha	

En resumen, se obtiene un prototipo preliminar plenamente funcional, que es capaz de moverse de forma autónoma y de esquivar obstáculos. En caso de colisión, puede retroceder gracias a los sensores de colisión, girar y proseguir con su camino. Además, es capaz de evitar caídas por desniveles debido al *cliff sensor*.

Por lo tanto, se opta por dar por cerrado esta parte e iniciar el diseño y construcción del prototipo final. Éste se basará en los circuitos electrónicos y en el código del prototipo V1.3, pero además tendrá incorporado un módulo de aspiración. De esta forma, podrá aspirar mientras se va desplazando, que es uno de los requisitos planteados en el *Capítulo 1: Introducción.*

4.2 Diseño del prototipo final V2.0

Partiendo de los circuitos electrónicos (Ilustración 30) y del código del prototipo V1.3, se diseña el prototipo final. Éste nuevo modelo debe disponer de una mayor superficie para incorporar el módulo de aspiración.

Se decide reutilizar el módulo de aspiración de un robot aspirador comercial llamado Q7. De izquierda a derecha, la ilustración 33 consta de:

- Una primera caja hueca con escobillas que es la que debe ir a ras del suelo para recoger la suciedad. Lleva un motor incorporado que hace girar las escobillas para ayudar en la tarea de aspiración.
- Una segunda caja que sirve de depósito para la suciedad y que incorpora un filtro para que las partículas no pasen al motor de succión.
- Un motor de succión que opera con 15V. Se encargará de ayudar a la primera caja con escobillas a aspirar la suciedad del suelo.

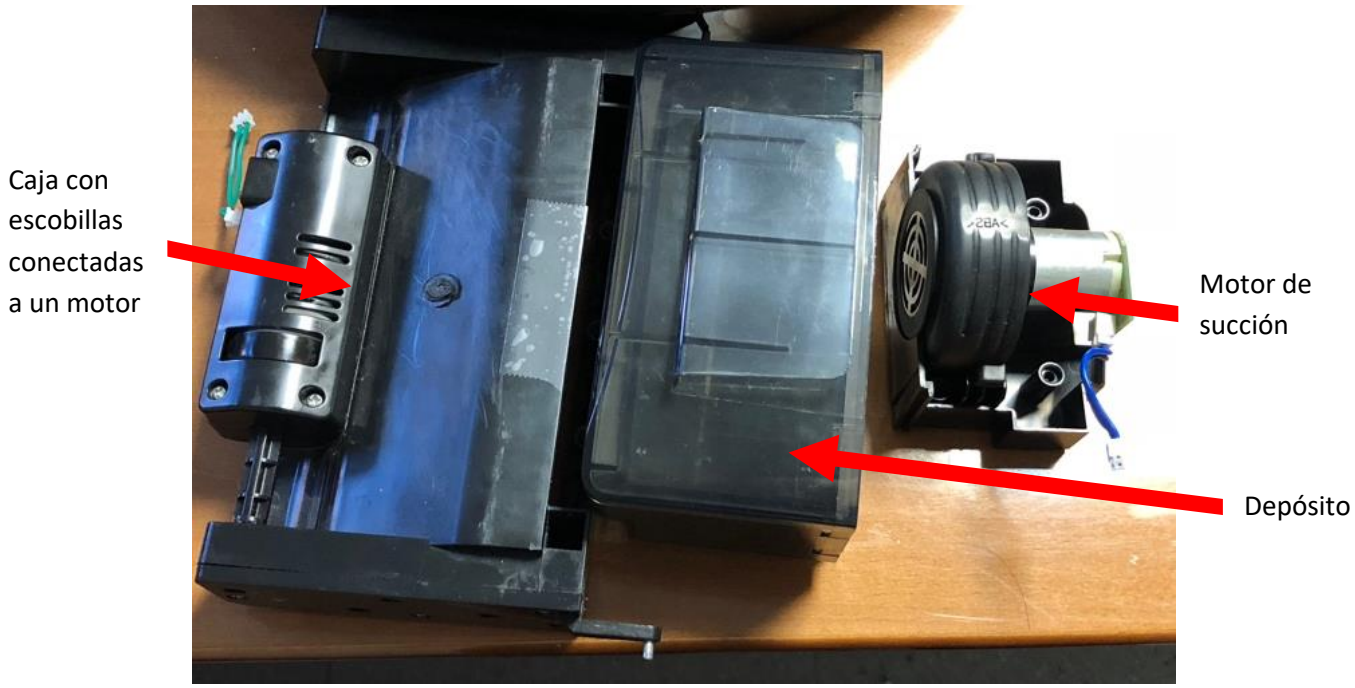


Ilustración 33 Módulo de aspiración reutilizado del Q7

Este módulo de aspiración incrementa considerablemente tanto la superficie como el peso que ha de soportar el chasis, por lo que se decide cambiar el metacrilato por piezas de metal de un mecano (marca Eitech) que se atornillan para construir una estructura hexagonal, como se ve en la ilustración 34. Además, los motores necesitarán tener torques mayores para mover este prototipo que es más pesado que el anterior. Es deseable que la caja con las escobillas esté lo más próxima posible al suelo para no perder eficiencia en la aspiración.

Como el prototipo anterior, constará de dos motores unidos por sus ejes a dos ruedas. En este caso los motores serán de metal y se sitúan de forma vertical para liberar espacio para

el módulo de aspiración. Sin embargo, en este caso no llevará dos “ruedas libres” sino una en disposición triangular respecto a las otras dos.

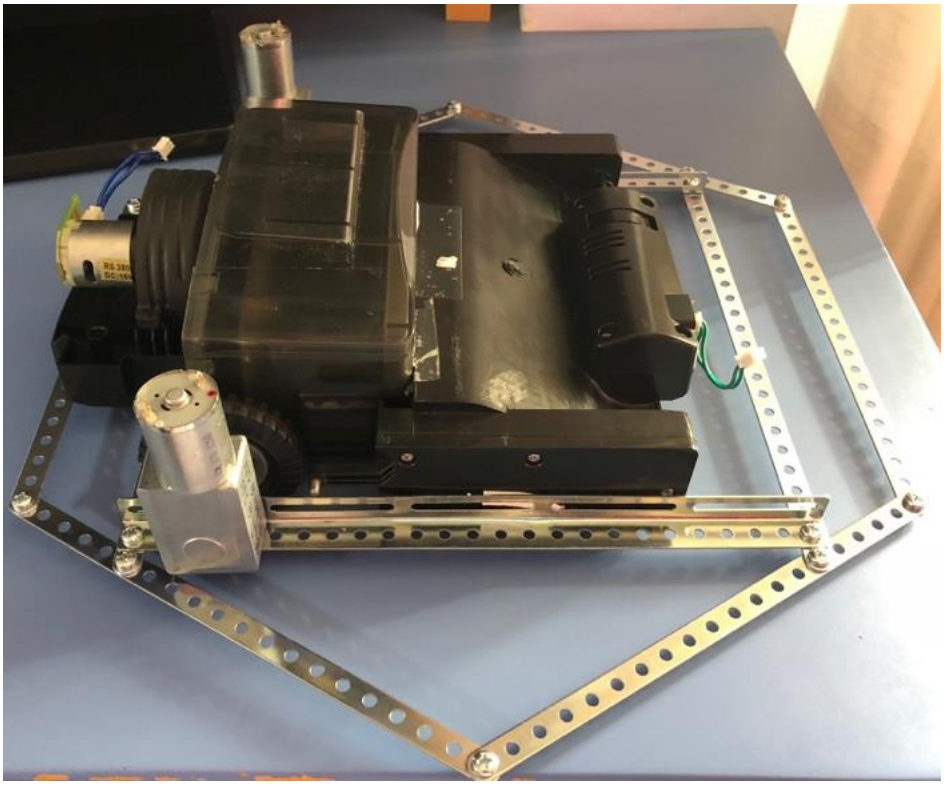


Ilustración 34 Módulo de aspiración y motores incorporados en el chasis de metal

En una primera aproximación del prototipo y puesto que su peso se ha incrementado notablemente llegando a alcanzar 2 Kg, se crea una base de cartón para el Arduino Mega (Ilustración 35), las baterías y el L298N. Adicionalmente servirá como material aislante para que no haya conexiones indeseadas de los componentes con los segmentos de metal.

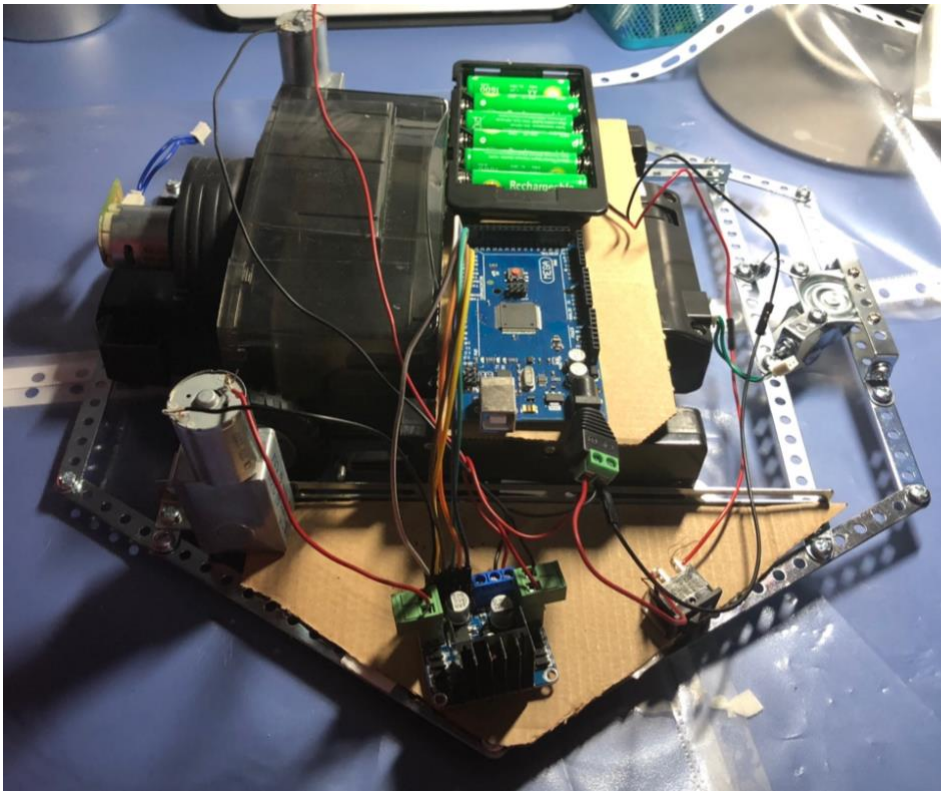


Ilustración 35 Prototipo V2.0 con el Arduino Mega y el L298N conectados

En la ilustración 36 se ve como se ha unido al chasis la rueda libre en la parte delantera de la estructura. Además, se pueden apreciar las conexiones entre los motores de las ruedas de tracción, el L298N y las baterías.

El siguiente paso es la colocación de los sensores de colisión y de ultrasonidos para evitar obstáculos. Para simplificar las conexiones se deciden soldar los cables a una PCB, con una regleta de conexiones de tierra y otra a los 5V que provienen del Arduino.

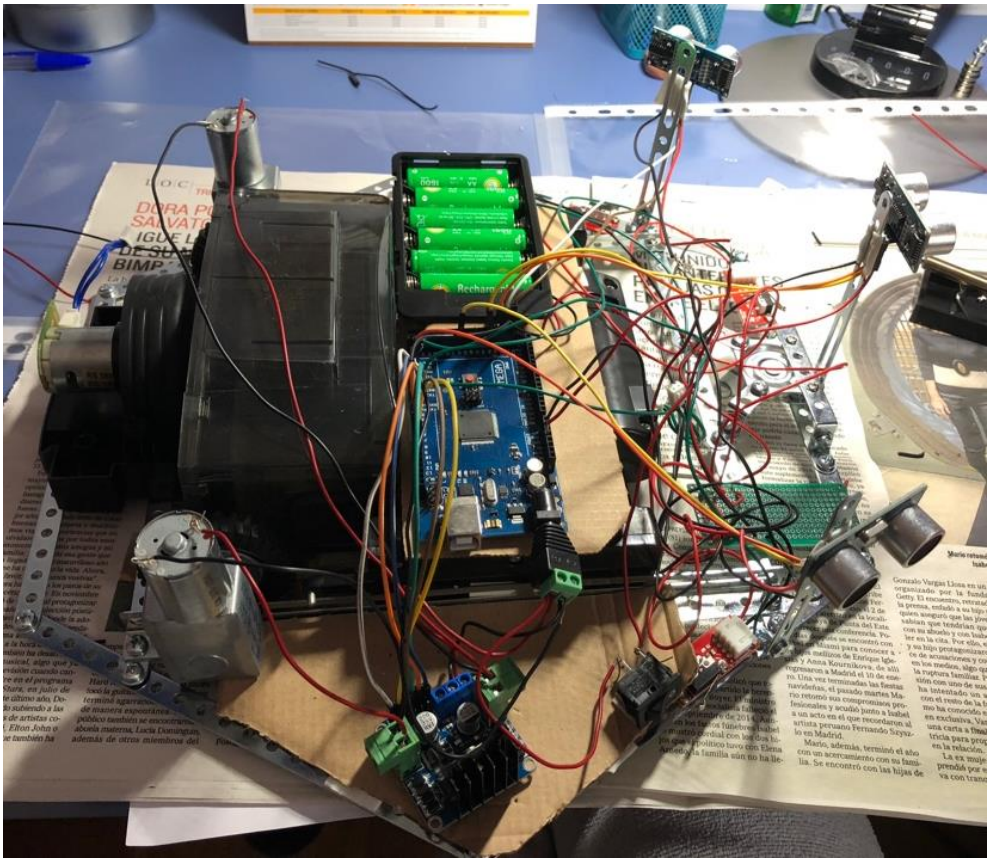


Ilustración 36 Prototipo V2.0 con sensores incorporados

Por último, para equipararlo al prototipo V1.3, falta la incorporación de los dos sensores de infrarrojos laterales. Además, de cara a optimizar su actuación frente a una colisión, era necesario incorporar parachoques tanto en el centro como en los laterales para facilitar la activación de los sensores de colisión. Tras probar diversas opciones, se escoge un diseño sencillo (Ilustración 37) aprovechando clips y su gran elasticidad y recuperación tras el golpe.

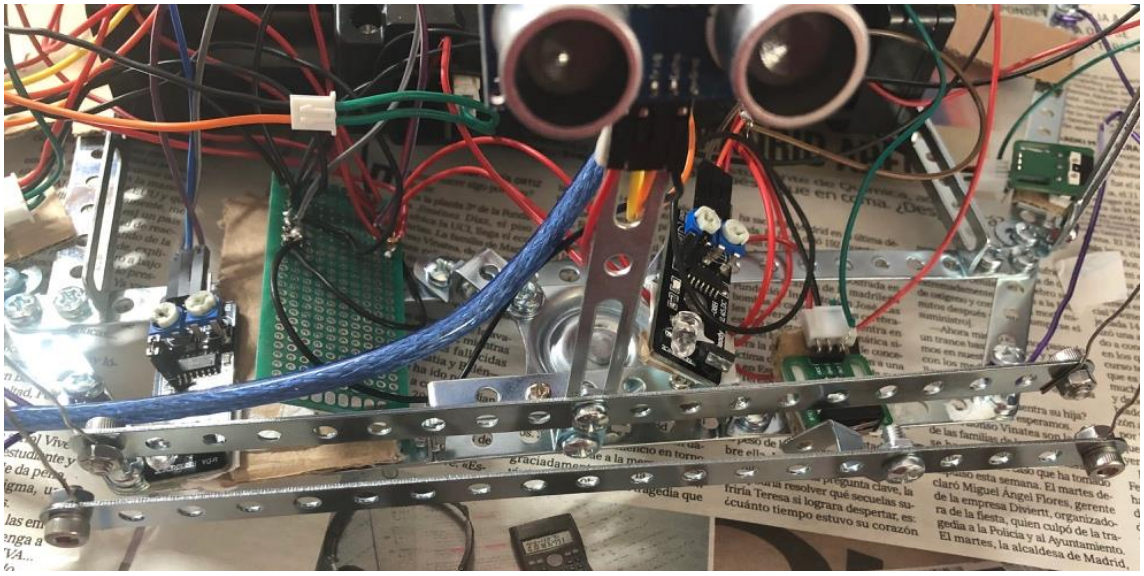


Ilustración 37 Parachoques frontal prototipo V2.0

Se añade también el sensor Sharp que actúa de *cliff sensor* como se ve en la ilustración 38.

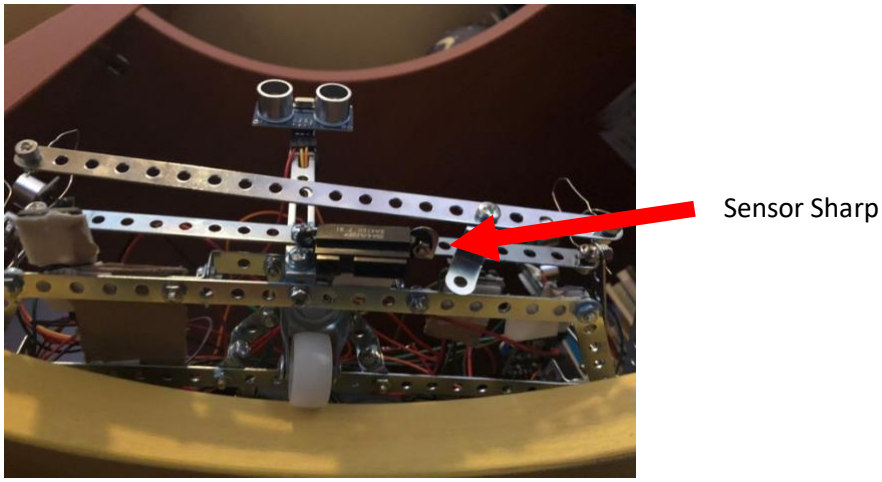


Ilustración 38 Prototipo V2.0 con el Sharp incorporado

Hasta este punto se ha obtenido un prototipo de robot que puede evitar obstáculos de forma autónoma creado con productos *open source*.

Para conseguir un prototipo de robot aspirador funcional, era necesario alimentar los dos motores del módulo de aspiración para que pueda aspirar según se vaya desplazando por las distintas habitaciones.

Se usan un total de 18 baterías como se aprecia en la ilustración 39:

- 6 de ellas para alimentar el L298N y el Arduino Mega que se encargan del movimiento y de evitar obstáculos.
- Otras 12 para alimentar el motor de succión y el motor que hace girar las escobillas que se encargan de la función de aspirado.

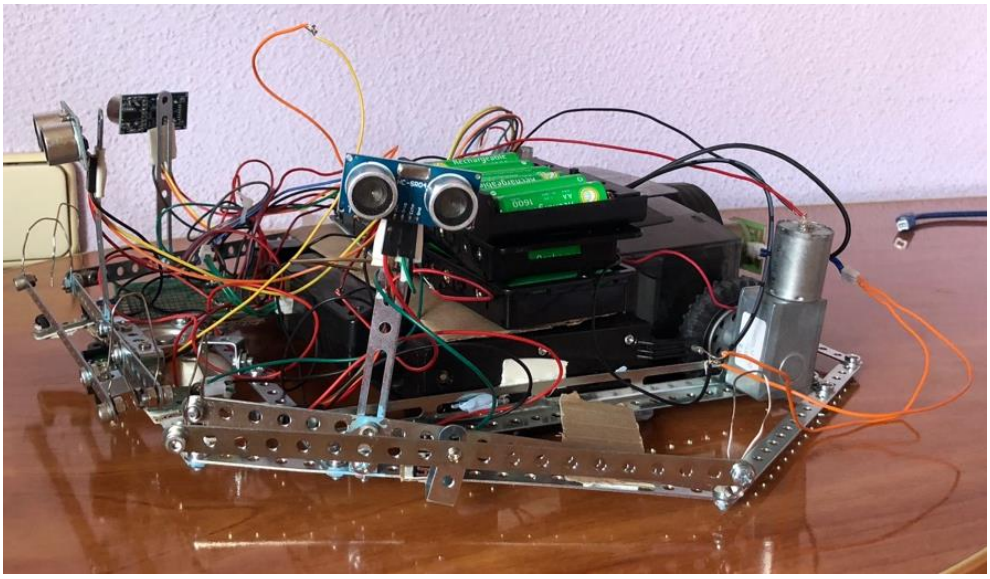
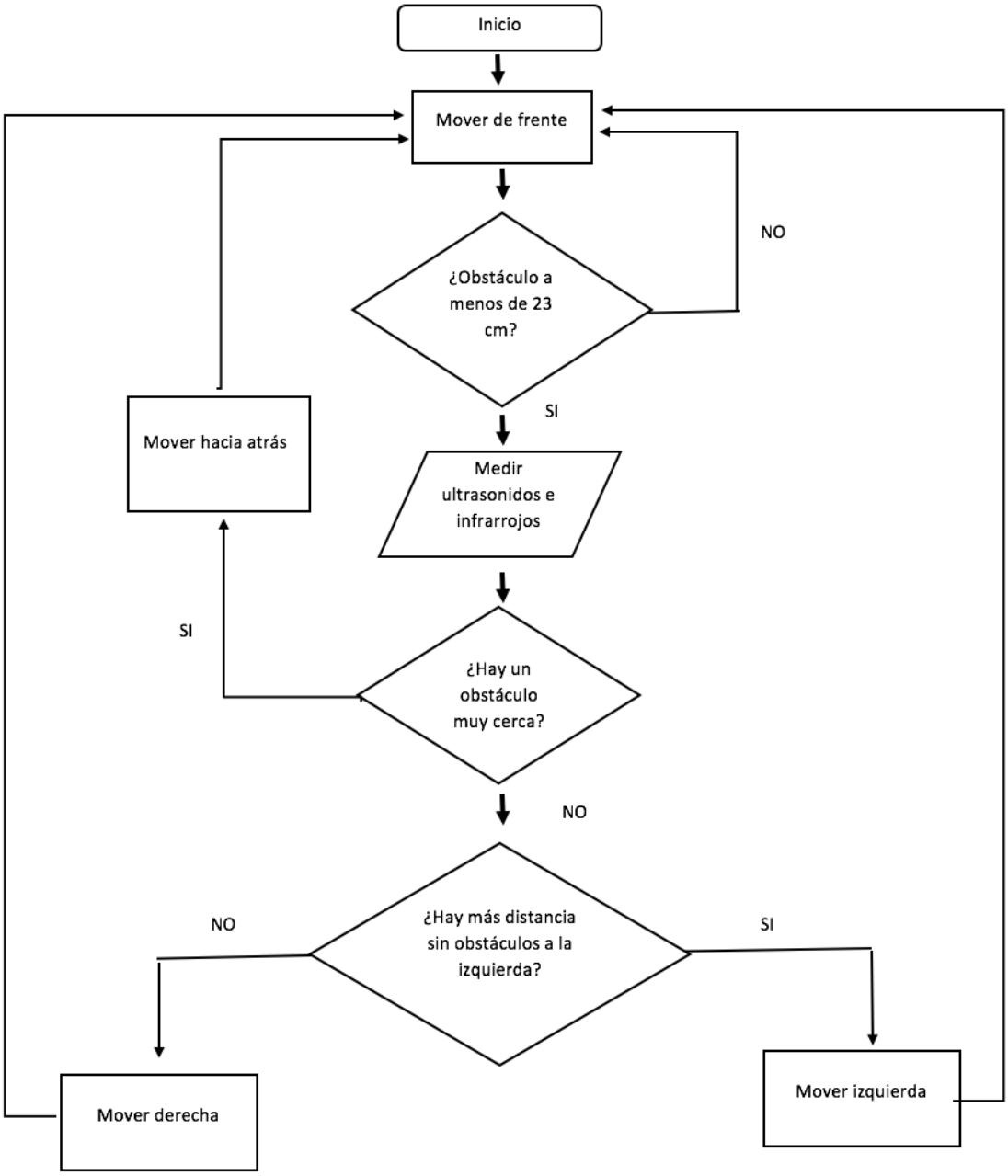


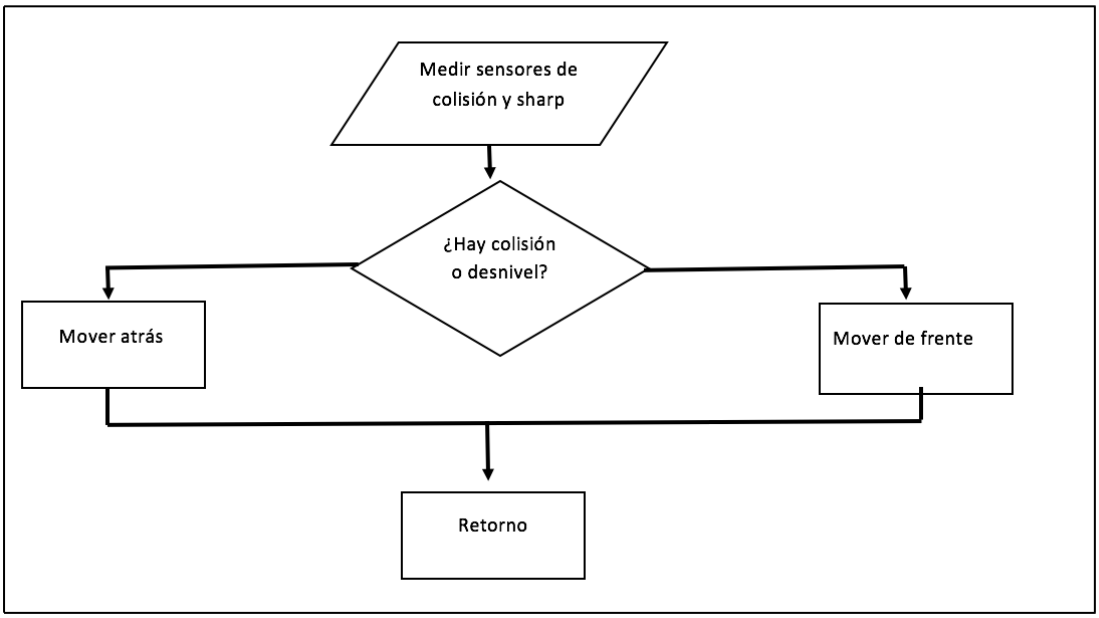
Ilustración 39 Prototipo V2.0 con las baterías conectadas a los motores de succión y de las escobillas

4.3 Diagramas de Flujo

El código final se ha desarrollado por completo con el software gratuito de Arduino y se presentará en el *Capítulo 12: Anexos*.
El diagrama de flujo que se ve representado a continuación:



Rutina de interrupción



5. PRUEBAS EXPERIMENTALES Y DE VALIDACIÓN

En este capítulo se documentarán las pruebas realizadas antes, durante y después del desarrollo del prototipo. También se analizarán cuáles han sido los principales fallos y problemas y se buscarán posibles soluciones por si se quiere retomar este trabajo en un futuro.

Por un lado, se incluyen las pruebas previas al diseño del prototipo que sirvieron para aprender cómo funcionaba cada sensor. Para analizar el prototipo V2.0 y ver si cumple con los requisitos planteados en el *Capítulo 1: Introducción*, se han realizado algunas pruebas basadas en diversos escenarios.

5.1 Pruebas con el material experimental

El trabajo de investigación previo al diseño y construcción del prototipo se basa en una parte teórica y otra práctica. La teórica se ha detallado en el *Capítulo 2: Estado del Arte* y *Capítulo 3: Análisis del Sistema* y para la práctica se llevaron a cabo varios experimentos.

Básicamente, se probaron los distintos sensores con una placa Arduino y se analizó el funcionamiento de cada uno de ellos. Gracias a la función *Serial Plotter* del software, se pudo observar en una gráfica la señal que está recibiendo el Arduino del sensor conectado. En la ilustración 40 se observa un ejemplo de una prueba llevada a cabo con un sensor Sharp y un sensor infrarrojo de proximidad para comparar sus respectivos rangos de funcionamiento.

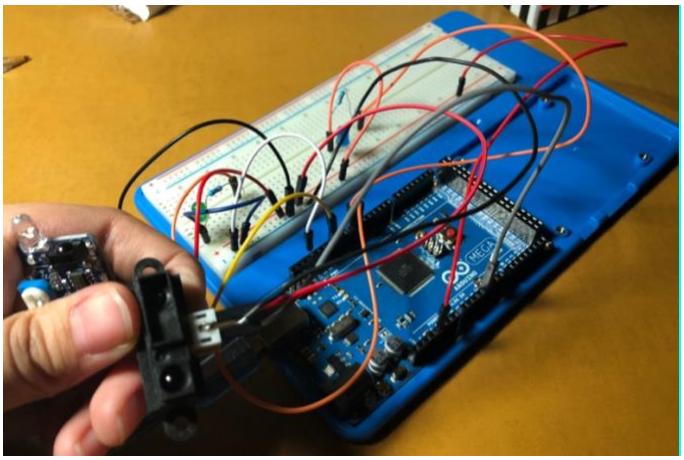


Ilustración 40 Ejemplo de experimento llevado a cabo con un sensor Sharp y un sensor de infrarrojos de proximidad

5.1.1 Sensor de colisión

Si no hay nada que presione el sensor de colisión, la señal digital que envía al Arduino tiene un valor igual a 0. Si, por el contrario, algo hace contacto con él, enviará una señal de valor digital 1 (Ilustración 41).

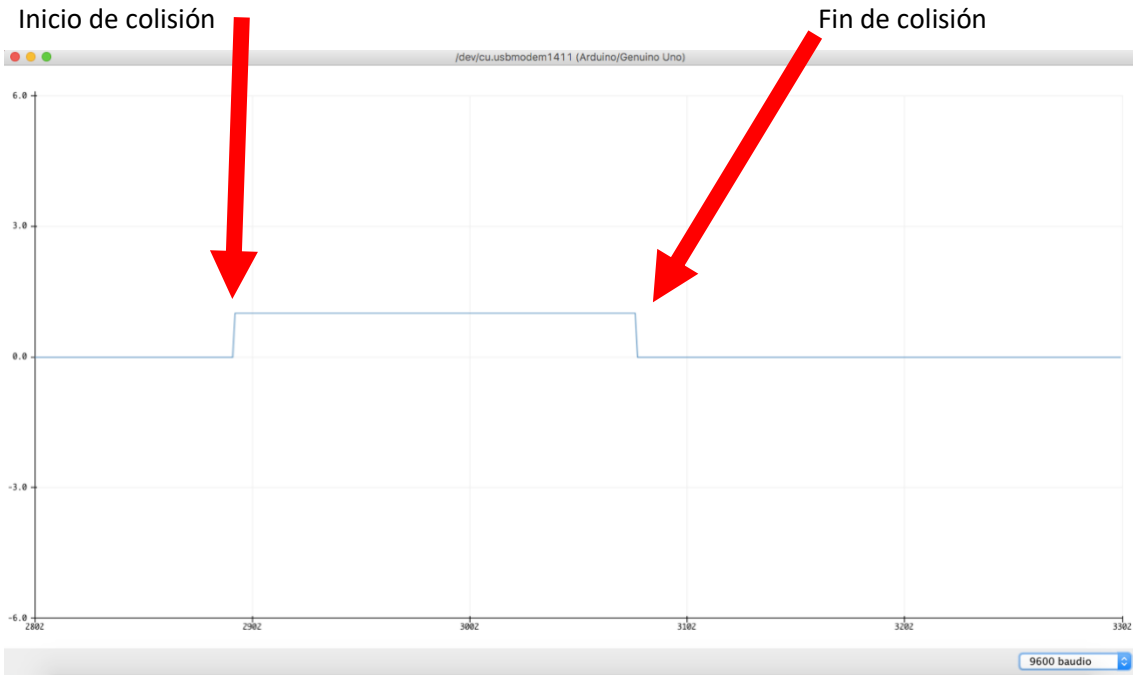


Ilustración 41 Representación con la función Serial Plotter del sensor de colisión

5.1.2 Sensor de infrarrojos de proximidad

Cómo se puede ver en la ilustración 42, la onda mostrada por el *Serial Plotter* del software de Arduino es digital. Tiene valor 1 en el caso de que el sensor no detecte ningún obstáculo en su rango de alcance y de valor 0 en el caso contrario.

Ilustración 42 Representación con la función Serial Plotter del sensor de infrarrojos de proximidad

5.1.3 Sensor infrarrojo de Sharp

Con este sensor, se puede apreciar ruido en la señal enviada al Arduino (Ilustración 43), por lo que también puede dar problemas con la precisión. Cabe destacar que es el único sensor utilizado que genera una señal analógica. Además, teniendo en cuenta que su precio es considerablemente superior a los ultrasonidos, se decide descartarlo como sensores delanteros y laterales para detector obstáculos. Debido a su precisión, se empleará únicamente para detectar escalones y desniveles.

Ilustración 43 Representación con la función Serial Plotter del sensor Sharp

5.1.3 Sensor de ultrasonido HC-SR04

La señal reflejada por el sensor de ultrasonidos no es una señal tan pura como en el caso del sensor de infrarrojos de proximidad a pesar de enviar también una señal digital al Arduino. En este caso, se puede apreciar un ligero ruido en la ilustración 44.

Ilustración 44 Representación con la función Serial Plotter del sensor de ultrasonidos

5.1.4 Resumen de los resultados obtenidos

A continuación, se presenta un resumen de las distancias máximas de detección de obstáculos de los sensores analizados en este capítulo. Los datos se han obtenido de forma experimental, se fijaba el sensor y se colocaba un obstáculo que se iba moviendo mientras se observaba la señal recibida en el Arduino. El valor medido es la distancia máxima a la que el sensor era capaz de detectar el obstáculo.

También se realizará una comparación con la distancia máxima extraída de los *datasheets* de cada componente. Se tomará este dato como valor exacto (oficial) para poder calcular el error relativo con la siguiente fórmula:

$$\varepsilon_{relativo} = \frac{|Valor\ medido - Valor\ oficial|}{Valor\ exacto} x100$$

Tabla 12 Resumen distancias máximas a las que los sensores detectan un obstáculo

Sensor	Valor medido (cm)	Valor oficial (cm)	$\varepsilon_{relativo}$ (%)	Precio (€)
Infrarrojo de proximidad	4	30 [38]	86,67	7,24
Infrarrojo Sharp	40	80 [39]	50	11,90
Ultrasonidos HC-SR04	30	400 [37]	92,5	1,78

En resumen, los tres sensores analizados presentan un comportamiento muy ineficiente al tener un error relativo muy alto. El Sharp es el que tiene un mejor rendimiento, pero hay que tener en cuenta que su precio también es superior. El motivo de dicho error no es del todo claro, podría deberse a problemas de condiciones ambientales, problemas de ajuste, u otras circunstancias. Pero causan inconvenientes a la hora de diseñar los prototipos.

Cabe destacar que son pruebas de distancias máximas de detección de obstáculos, por lo que, si se necesita usar estos sensores para un experimento con un rango menor al valor medido, el funcionamiento de los tres es muy similar y los resultados son positivos. Sin embargo, en el caso de necesitar detectar una distancia mayor, es muy posible que haya que emplear un componente con un rango superior a pesar de lo que indique en el *datasheet*. Esto puede generar muchos problemas a la hora de elaborar prototipos por lo que es necesario tener cuidado con los componentes *open source* ya que su comportamiento y tolerancia sirve para niveles básicos de experimentación.

5.2 Pruebas de funcionamiento autónomo, con obstáculos y desniveles

Estas pruebas sirven para determinar si el robot es capaz de moverse de forma autónoma sin tener dificultades.

Tabla 13 Resultados del prototipo V2.0 ante diversos escenarios

Escenario		**Resultado**
Obstáculo en frente a menos de 23 cm		El robot se para y analiza la distancia a su derecha y a su izquierda.
	Si hay más distancia hasta un obstáculo en el lado derecho que en el izquierdo	El robot gira a la derecha
	Si hay más distancia hasta un obstáculo en el lado izquierdo que en el derecho	El robot gira a la izquierda
Detecta una colisión en uno de los sensores de contacto (frontal o lateral)		El robot para y automáticamente se desplaza hacia atrás y a la izquierda
El robot alcanza una esquina		Realiza maniobras de escape, y consigue salir, aunque le puede llevar un rato.
Colisión		Si hay una colisión en uno de los parachoques el robot se desplaza hacia atrás y hacia la izquierda, salvando la situación. Problema: Si el golpe no coincide con el parachoques el robot no sabe que ha chocado y los motores siguen funcionando
Superar un cable		Remonta cables de hasta 2 mm de grosor
Remontar una superficie elevada de perfil abrupto perpendicular al avance		El robot remonta obstáculos de hasta 3,77 mm de grosor
Subir obstáculos de perfil abrupto orientado en 45° respecto al avance		El robot es capaz de remontarlos hasta 3 mm de grosor
Subir a una alfombra		No, dado que el grosor de la mayoría de las alfombras supera los 3,77 mm
Avanzar por una alfombra		El robot no tiene la potencia necesaria para avanzar en la alfombra y se queda atascado
Patrones alternativos blanco, negro		El robot no los percibe como escalones y no retrocede. Para muchos modelos de robot supone un problema dado que los confunden con un escalón.

Como conclusiones podemos extraer el robot es capaz de actuar de forma autónoma de una manera rudimentaria. Es decir, puede detectar casi todos los obstáculos y actuar acorde a ellos dependiendo de la situación y distancia, pero no lo hace de una forma eficiente al necesitar en muchos casos varios movimientos para salir de escenarios más complejos. La solución requeriría de algoritmos de escape muy elaborados.

Otro problema son los ángulos muertos, ya que el campo de "visión" de los sensores ultrasonidos no cubre todo el lado del chasis del robot, y en ocasiones no detectan obstáculos pequeños al no entrar en su rango de visión. Una posible solución es incrementar el número de sensores por cada lado para que pueda detectar todos los obstáculos, por pequeños que sean.

Además, de cara a futuros trabajos, sería interesante usar ruedas con mejor agarre, motores más potentes, o incluso combinarlos con bandas de tracción como los del robot aspirador comercial de la marca Dyson, y que permitan que suba alfombras y cables.

No obstante, es capaz de detectar escalones independientemente de lo reflectante que sea el suelo (negro o blanco) gracias al Sharp que usa como *Cliff sensor*. Tampoco confunde los patrones que alternan franjas blancas y negras con escalones. Estos casos suponen unos de los fallos más comunes de robots aspiradores comerciales. Por un lado, suelen interpretar un escalón muy reflectante como suelo nivelado (mientras que uno poco reflectante se detecta fácilmente como desnivel). Y, por otro lado, sucede con frecuencia que una alfombra con patrones blanco-negros se confunda con un escalón en la transición del blanco al negro, y decida retroceder aunque no haya más que una alfombra. En ambos casos el prototipo se comporta de forma correcta.

5.3 Pruebas de aspirado

Debido principalmente a que el módulo de aspiración se ha reutilizado de un robot de segunda mano de una marca poco conocida, los resultados de aspirado no han sido del todo satisfactorios.

Cómo prueba para determinar la calidad de la limpieza (explicada en el *Capítulo 2: Estado del Arte, subcapítulo, 2.1.1 Criterios de evaluación de robots aspiradores*) se dispusieron trozos de papel colocados delante del robot aspirador. Debido a una serie de imprevistos (motor de succión endeble, caja de aspirado con abertura muy grande, mal sellado del combo de aspirado y falta una escobilla) la capacidad de aspirado del prototipo no era demasiado alta. Para obtener una tasa de aspirado mayor (en torno al 40% de papeles aspirados) era necesario bajar el combo de aspiración para que estuviese más cerca del suelo por lo que el grosor de cables y de obstáculos que podía remontar disminuía. Por el contrario, si se elevaba el combo de aspiración para que pudiese sortear obstáculos del grosor mencionado en la tabla 13, la tasa de aspirado bajaba a un 10%. De tal forma, fue necesario buscar un balance entre estos dos diseños.

No obstante, es un problema que puede tener una solución sencilla de cara a futuros trabajos ya que únicamente habría que añadir otra escobilla y emplear un motor de succión más potente. No se ha podido hacer en este proyecto por falta de tiempo.

6. GESTIÓN DEL PROYECTO

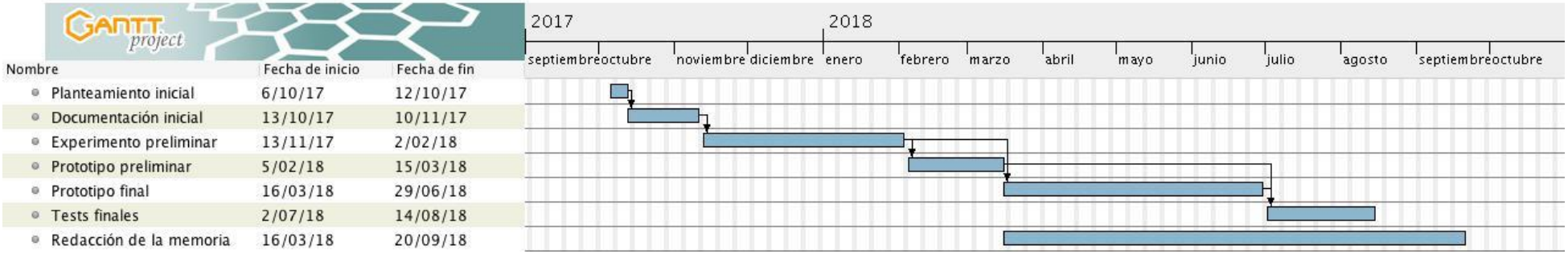
En este capítulo se analizan los recursos usados para la elaboración del proyecto. Para ello se subdivide en los dos principales recursos empleados: tiempo y dinero. Para analizar el primero, se estudiará el tiempo invertido en cada una de las principales tareas llevadas a cabo. Por otro lado, el aspecto económico se estudiará en el apartado de presupuesto donde se desglosará cada coste.

6.1 Planificación

En este apartado se muestra de una manera global el orden de las tareas realizadas para llevar a cabo el proyecto y las fechas en las que se ha ido desarrollando. A continuación, se exponen cuáles han sido las tareas principales:

- Planteamiento inicial del proyecto: esta tarea se basa en estudiar la viabilidad de diseñar y construir un prototipo funcional de un robot aspirador basándose en productos *open source*.
- Documentación inicial: tras estudiar la viabilidad del proyecto, se realiza una búsqueda de información para analizar las alternativas de productos *open source* en el mercado y cuáles se pueden incluir en el prototipo.
- Experimentos preliminares: esta tarea incluye todas las pruebas con los distintos componentes tecnológicos para entender su funcionamiento.
- Prototipo preliminar: incluye el diseño, construcción, depuración y pruebas de control de los prototipos V1.0, V1.1, V1.2 y V1.3.
- Prototipo final: incluye el diseño, construcción, depuración y análisis del prototipo V2.0.
- Tests finales: esta tarea se basa en la obtención de los resultados del proyecto explicados en el *Capítulo 5: Pruebas experimentales y de validación*.
- Redacción de la memoria: es la única tarea que puede solaparse con el resto. Puesto que se pueden ir redactando algunas partes de la memoria mientras se realizan pruebas para optimizar el tiempo.

En la ilustración 45 se muestra el diagrama de Gantt correspondiente a la planificación previa. No se ha incorporado un diagrama de recursos ya que la principal restricción ha sido el tiempo.



Nombre	Fecha de inicio	Fecha de fin
Planteamiento inicial	6/10/17	12/10/17
Documentación inicial	13/10/17	10/11/17
Experimento preliminar	13/11/17	2/02/18
Prototipo preliminar	5/02/18	15/03/18
Prototipo final	16/03/18	29/06/18
Tests finales	2/07/18	14/08/18
Redacción de la memoria	16/03/18	20/09/18

Ilustración 45 Diagrama de Gantt del proyecto

En la tabla 14 se expone un resumen de las fechas y de la duración de cada una de las tareas.

Tabla 14 Planificación de tareas

Tarea	Fecha inicio	Fecha fin	Duración (días)
Planteamiento inicial	06/10/2017	12/10/2017	6
Documentación inicial	13/10/2017	12/11/2017	30
Experimentos preliminares	13/11/2017	02/02/2018	81
Prototipo preliminar	03/02/2017	15/03/2018	40
Prototipo final	16/03/2018	31/06/2018	107
Tests finales	02/07/2018	14/08/2018	43
Redacción de la memoria	16/03/2018	20/09/2018	188

De forma que el proyecto ha durado un total de 11 meses con un promedio de 4 horas dedicadas al día.

6.2 Presupuesto

En esta parte se detallará el presupuesto empleado para realizar este TFG, desglosando cada coste en función de su naturaleza.

6.2.1 Coste del personal

Los costes de personal detallados a continuación se han calculado usando el salario medio trabajando una jornada de 4 horas al día de un ingeniero junior con 1 año de experiencia en España:

Tabla 15 Coste del personal

Perfil	Meses	Coste mensual (€)	Coste total (€)
Ingeniero Junior	11	700	7.700

6.2.2 Coste de herramientas

Los costes de las herramientas usadas se detallan a continuación.

Tabla 16 Coste de herramientas

Componente	Coste (€)	Tiempo de vida (meses)	Coste mensual (€)	Meses utilizados	Coste amortizado (€)
Portátil Mac Book Air	1100	72	15,27	11	167,97
Multímetro digital	8	60	0,13	5	0,65
Soldador	20	60	0,33	6	1,98
				TOTAL	**170,6**

6.2.3 Coste de material experimental

Previamente al diseño de los prototipos se realizaron una serie de pruebas con distintos ejemplos de motores y sensores para observar mejor las diferencias y elegir los que se consideraron más adecuados para el prototipo.

Además, en esta sección se incluyen una serie de kits usados de distintas formas:

- *Elegoo Car Kit V2.0*: se usa un prototipo comercial para tener una idea general del funcionamiento y de los componentes de un coche que evita obstáculos basado en Arduino.
- *Elegoo Starter Kit y Elegoo Sensor Kit V2*: se usan estos kits debido a su gran versatilidad al disponer de muchos componentes para probar cuáles son los adecuados para el prototipo. Además, las resistencias y LEDs usados provienen de estos kits.

El resto de los componentes presupuestados no se usaron finalmente en los prototipos.

Tabla 17 Coste del material experimental

Componente	Unidades	Precio de venta unitario (€)	Precio total (€)
Elegoo Car Kit V2.0	1	69,99	69,99
Elegoo Sensor Kit V2	1	39,99	39,99
Elegoo Starter Kit	1	55,99	55,99
Sunfounder holder+wires	1	11,99	11,99
Elegoo breadboard 830 pts	1	9,99	9,99
US-015	1	8,07	8,07
US SF-SR02	1	5,99	5,99
Protoshields + Breadboards	1	1,60	1,60
32 mini PCB	1	11,99	11,99
Sharp GP2Y0A21YK	1	14,98	14,98
Sharp gp2y0a02yk0F	1	14,82	14,82
Motor 12V 50 rpm	2	11,50	23,00
Motor 12V 40 rpm	2	11,54	23,08
		TOTAL	**291,48**

6.2.4 Costes de prototipos

En este apartado se presupuesta el coste de cada prototipo de forma individual sin tener en cuenta componentes reutilizados. Es decir, en el caso de querer fabricar de cero uno de los siguientes prototipos, esta estimación serviría para dar una idea del precio.

6.2.4.1 Costes de prototipos preliminares

Tabla 18 Coste del prototipo V1.0

Componente	Unidades	Precio de venta unitario (€)	Precio total (€)
Kit chasis car 2WD	1	11,59	11,59
Arduino Uno	1	20,72	20,72
L298N Sunfounder	1	8,55	8,55
Motor DC + rueda	2	6,33	12,66
Baterías LiPo 18650 3,7V	2	8,40	16,80
Portapilas 2x18625	1	0,95	0,95
		TOTAL	**71,27**

Tabla 19 Coste del prototipo V1.1

Componente	Unidades	Precio de venta unitario (€)	Precio total (€)
Chasis Car TBS2652	1	24,99	24,99
Arduino Uno Elegoo	1	7,99	7,99
Arduino Nano Elegoo	1	3,66	3,66
L298N Sunfounder	1	8,55	8,55
Sensores de colisión bte 16-15	3	1,40	4,20
Motor DC + rueda	4	6,33	25,32
Baterías LiPo 18650 3,7V	2	8,40	16,80
Portapilas 2x18625	1	0,95	0,95
		TOTAL	**92,46**

Tabla 20 Coste del prototipo V1.2

Componente	Unidades	Precio de venta unitario (€)	Precio total (€)
Chasis Car TBS2652	1	24,99	24,99
L298N Sunfounder	1	8,55	8,55
Sensores de colisión bte 16-15	3	1,40	4,20
Motor DC + rueda	2	6,33	12,66
Arduino Mega Elegoo	1	11,99	11,99
Baterías LiPo 18650 3,7V	2	8,40	16,80
Portapilas 2x18625	1	0,95	0,95
US HC-SR04	3	1,78	5,34
		TOTAL	**85,48**

Tabla 21 Coste del prototipo V1.3

Componente	Unidades	Precio de venta unitario (€)	Precio total (€)
Chasis Car TBS2652	1	24,99	24,99
L298N Sunfounder	1	8,55	8,55
Sensores de colisión bte 16-15	3	1,40	4,20
Motor 15 rpm + rueda	2	6,33	12,66
Arduino Mega Elegoo	1	11,99	11,99
Baterías NiMh AAA	6	1,50	9,00
Portapilas 6x AAA	1	2,57	2,57
US HC-SR04	3	1,78	5,34
Sharp GP2Y0A41SK0F	1	11,90	11,90
SunFounder Obstacle Avoidance Sensor	2	7,24	14,48
		TOTAL	**105,68**

En este apartado para el cálculo del presupuesto final, se considerará solo el coste del prototipo V1.3. Este es el prototipo preliminar que más elementos tiene, y que han sido reutilizados en parte de los anteriores que se llevaron a cabo como pruebas para desarrollarlo.

6.2.4.2 Costes de prototipo final

Tabla 22 Coste del prototipo V2.0

Componente	Unidades	Precio de venta unitario (€)	Precio total (€)
Motor 12 V 100 rpm	2	7,34	14,67
L298N Sunfounder	1	8,55	8,55
Sensores de colisión bte 16-15	3	1,40	4,20
Arduino Mega Elegoo	1	11,99	11,99
Pilas	18	1,50	27
Portapilas	3	2,57	7,71
US HC-SR04	3	1,78	5,34
Sharp GP2Y0A41SK0F	1	11,90	11,90
SunFounder Obstacle Avoidance Sensor	2	7,24	14,48
Pack de ruedas (Eitech)	1	12,61	12,61
Pack de segmentos (Eitech)	3	13,12	39,36
Pack de piezas planas (Eitech)	1	16,68	16,68
Pack de piezas angulares	1	5,99	5,99
Q7 reutilizado	1	18	18
PCB	1	2,88	2,88
		TOTAL	**201,36**

6.2.5 Costes adicionales

En este apartado se detalla el coste de los elementos auxiliares empleados en los prototipos. Es decir, tanto cables como elementos de fijación y pegamentos (Tabla 23).

Tabla 23 Otros costes

Componente	Unidades	Precio de venta unitario (€)	Precio total (€)
Elegoo cables dupont	2	7,09	14,18
Cable unipolar 0,5 mm	25 m	4,99	4,99
Cables jst para Sharp IR	4	3,79	15,16
Bridas cortas	1	1,00	1,00
Bridas medianas	1	1,50	1,50
Espaciadores M3 latón	1	13,07	13,07
Eitech tuercas y tornillos	1	13,67	13,67
Tornillos M3 300 piezas	1	9,99	9,99
Tornillos M2 M3 M4 1080 piezas	1	20,69	20,69
Tornillos M4 270 piezas	1	10,49	10,49
Pegamentos variados	1	10,00	10,00
		TOTAL	**114,74**

6.2.6 Coste total

Tras presentar los distintos costes se adjunta una tabla resumen de los costes del proyecto (Tabla 23).

Tabla 24 Coste total

Descripción	Costes (€)
Personal	7.700,00
Herramientas	170,60
Material experimental	291,48
Prototipo V1.3	105,68
Prototipo V2.0	201,36
Costes adicionales	114,74
SUBTOTAL	8.583,86
IVA (21%)	1.802,61
COSTE TOTAL DEL PROYECTO	**10.386,47**

Es importante aclarar que éste es el coste total del proyecto. En el caso de querer comercializar el prototipo, el coste corresponde aproximadamente al coste del prototipo final V2.0 añadiéndole los costes adicionales y los costes de amortización.

Además, si se quiere vender un gran número de ellos habría que analizar los costes del Arduino. En ese caso es más probable que compense diseñar el Hardware de un controlador específico. De esta forma se aprovecharían todos los recursos del microcontrolador, al contrario que en el caso del Arduino que con el que se dejan muchos pines y recursos de la placa sin usar, aumentando el coste.

7. ENTORNO SOCIO-ECONÓMICO

En este apartado, se realizará un análisis de los 10 robots aspiradores autónomos más vendidos según la web Amazon España (Tabla 25).

Tabla 25 Robots aspiradores autónomos más vendidos en Amazon España en el 2017
Fuente: https://www.amazon.es/gp/bestsellers/kitchen/2165706031

Posición	Marca	Modelo	Precio (€)
1	Cecotec	Conga excellence 990	219,00
2	Cecotec	Conga excellence	199,00
3	Yokkao	Yokkao	211,00
4	iRobot	Roomba 605	199,00
5	iRobot	Roomba 615	285,00
6	ILife	V3s Pro	139,00
7	Cecotec	Conga Slim 890	166,00
8	Cecotec	Conga Slim	159,00
9	iRobot	Roomba 960	694,90
10	Housmile	Housmile	99,99

Con esta tabla, se puede ver la tendencia del mercado español respecto a los robots aspiradores. Por un lado, la marca española Cecotec ocupa cuatro puestos de esta lista, incluyendo el primero y el segundo, por lo que se puede deducir que esta marca ha triunfado en el mercado español con este tipo de productos.

Por otro lado, los consumidores españoles están dispuestos a pagar precios relativamente bajos para un robot aspirador, aunque tengan menos prestaciones y sean menos eficientes que los modelos más caros que existen actualmente. De hecho, en esta lista hay dos modelos de marcas menos conocidas como son Housemile y Yokkao que tienen precios bastante bajos y es probable que por eso estén en esta lista de los más vendidos.

Hay que remarcar que los modelos más caros son los de iRobot, que sigue siendo la marca líder a nivel mundial en este mercado, y en esta lista tiene tres modelos entre los más vendidos por Amazon en España.

Por lo tanto, se pueden extraer dos grandes conclusiones. La primera es que los consumidores españoles no se quieren gastar mucho dinero en este tipo de aparatos y están dispuestos a aceptar robots con peores prestaciones y menor precio. La segunda es que los consumidores que están dispuestos a gastarse más dinero en un robot aspirador, lo hacen en una gran medida en los distintos modelos de Roomba, seguramente por la potente imagen de marca que tiene iRobot.

Extrapolando estas conclusiones al prototipo desarrollado con productos *open source*, éste tendrá éxito en España si se consiguiese vender a un precio muy bajo. Es posible que para lograr llegar a economías de escala y reducir lo máximo posible los costes, haya que sustituir los productos *open source* por diseños ad-hoc basados en microcontroladores y componentes discretos que tengan mejores tolerancias, sean más baratos y fáciles de encontrar.

8. MARCO REGULADOR

En el *Capítulo 1: Introducción* se ha expuesto que el movimiento *open source* es muy reciente, por lo que actualmente no existen unas pautas bien definidas sobre la protección legal de las distintas creaciones de cada autor.

Por un lado, según la legislación europea [40] se incluye el software libre como obra literaria por lo que está regulado por la propiedad intelectual. Es un tema complejo, puesto que, aunque un usuario puede proteger su código con una patente, esto va en contra del movimiento *open source* que se basa en compartir información libremente. Por otro lado, el hardware libre es un tema aún más confuso ya que es más reciente y aún se están negociando cuáles son los siguientes pasos a seguir [41]. El caso de Arduino es un claro ejemplo de la vinculación del software libre y del hardware libre, ya que permite que muchos usuarios puedan crear dispositivos con sus propios códigos.

Para el desarrollo del código de este proyecto, no ha sido necesario el uso de ninguna librería externa a las que ya lleva incorporado el IDE de Arduino. Por lo que no será necesario mencionar a su autor en este proyecto.

Respecto al mercado de robots aspiradores, su progresiva sofisticación mediante técnicas de localización y generación de mapas, empleando cámaras, GPS y conexión a internet, plantea nuevos retos que afectan a la privacidad de los usuarios. Es el caso de iRobot que ha tenido algunas polémicas con sus últimos modelos de la serie 900 que permiten hacer mapas de las viviendas. Antes de empezar a usar estos modelos, hay que aceptar obligatoriamente como término y condición que la empresa pueda vender esos datos a terceros [42]. Algunas empresas como Amazon o Apple están interesadas en comprar la información de cara a tener datos que analizar para implementar en un futuro un hogar inteligente.

Debido al profundo rechazo que esta idea generó y a la pérdida de clientes que valoran su privacidad y que no están dispuestos a que haya empresas que vendan sus datos a terceros, el CEO de la compañía Coling Angles declaró que usan los datos de los clientes únicamente para la elaboración de estadísticas y que en ningún caso los venderán a otras empresas [43].

Cabe destacar, que esto no aplica en el caso de este proyecto, puesto que en ningún momento se obtienen planos de viviendas.

Además, puesto que no se está planteando comercializar el prototipo, no se realizará un estudio más exhaustivo de la legislación europea actual para el comercio de aparatos electrónicos.

Por último, mencionar que se han usado dos programas *open source* gratuitos para este trabajo: Fritzing y Ganttproject. El primero permite crear y diseñar circuitos electrónicos de forma sencilla y el segundo sirve para la elaboración de diagramas de Gantt.

9. CONCLUSIONES Y FUTUROS TRABAJOS

En este capítulo se exponen las conclusiones obtenidas a lo largo del desarrollo del proyecto y un breve resumen de posibles mejoras de cara a futuros trabajos.

9.1 Conclusiones

El objetivo principal de este proyecto era obtener un prototipo de robot aspirador funcional, elaborado únicamente con productos *open source*. El resultado final ha sido satisfactorio, obteniéndose las siguientes conclusiones al respecto:

- El prototipo se ha desarrollado por completo con productos *open source* y de fácil acceso para todo el mundo.
- Se ha obtenido un amplio conocimiento sobre los diversos sensores, componentes y motores compatibles con Arduino y cuál se debe usar en cada circunstancia.
- Se ha construido un prototipo totalmente funcional de un robot capaz de evitar obstáculos de forma autónoma.
- Se ha obtenido un prototipo capaz de aspirar mientras se va desplazando.
- En base a que fuese lo más ecológico posible, se han usado únicamente baterías recargables.

No obstante, se han encontrado algunos problemas que han impedido que se cumplan los siguientes requisitos:

- No se ha logrado obtener un prototipo con un aspirado óptimo debido a problemas externos. El módulo de aspiración fue reciclado de un robot aspirador comercial y presentó una serie de problemas (motor de succión endeble, sellado incorrecto en las juntas del combo de aspiración, rendija de aspiración muy grande, y falta una de las dos escobillas).
- Debido principalmente a la falta de tiempo, no se ha podido incorporar un módulo Wifi que permita su control estando fuera de casa. Además, se intentó conectar el prototipo a un mando de infrarrojos, pero no fue posible debido a problemas de software con las librerías y el IDE de Arduino.

Tras la investigación llevada a cabo a lo largo de este proyecto, conviene añadir algunas consideraciones finales sobre los productos *open source*:

- Facilitan el diseño de dispositivos electrónicos y permiten elaborar prototipos básicos de forma rápida y económica.

- Existe una comunidad enorme que facilita información y ayuda en caso de tener problemas. Además de proporcionar muchos códigos ya creados que se pueden usar de forma libre.

No obstante, existen algunos aspectos mejorables.

- Todo el mundo puede subir información, pero no hay nadie que controle si es correcta. En el caso de no tener muchos conocimientos se pueden asumir como verdaderas cosas que no lo son.

- Numerosas empresas que fabrican componentes para este sector, copian y venden productos diseñados por otros a precios más bajos sin cuidar algunos aspectos:

    - Problemas de tolerancia en algunos productos. Por ejemplo, ocurre con frecuencia que para un par de motores idénticos uno de ellos gire a una velocidad mayor que el otro en igualdad de condiciones.

    - Calidad a veces deficiente. Los sensores no funcionan de forma óptima siempre, sino que hay mucho ruido en la señal de salida que dificulta su lectura. Por ejemplo, los sensores de infrarrojos de proximidad que en su ficha técnica indica un rango de funcionamiento de 3 a 30 cm y en realidad se ha comprobado que sólo funcionan a menos de 4 cm.

    - Falta de información técnica, es deficiente o errónea. A la hora de buscar motores para los prototipos, en muchos casos ni los vendedores ni los fabricantes ofrecen información técnica de los modelos.

Todas ellas han causado algunas complicaciones en el desarrollo de los prototipos de este proyecto. Por tanto, se puede extraer como conclusión que los productos *open source* permiten crear prototipos muy básicos que sirvan como primera aproximación del problema. Pero, es probable que en el caso de necesitar productos más robustos con una buena tolerancia y un funcionamiento óptimo sea necesario buscar otra solución.

No obstante, como conclusiones generales del proyecto, el resultado ha sido bastante satisfactorio, ya que se ha logrado diseñar y construir un robot que evita obstáculos partiendo de cero, usando los diversos conocimientos adquiridos a lo largo del grado.

9.2 Futuros trabajos

Si bien se ha dado el prototipo por cerrado tras haber obtenido los resultados necesarios para este proyecto, es importante destacar el enorme potencial que hay de cara a retomar el trabajo en un futuro:

- Incorporar otros motores de aspiración más potentes para que pueda aspirar de una forma más eficiente. Además, realizar un diseño con un recipiente de residuos que sea fácilmente extraíble para la comodidad del usuario.

- Incorporar un módulo wifi para controlarlo a pesar de estar fuera de casa.
- Se podría estudiar la posibilidad de optimizar el código para mejorar la eficiencia del prototipo e incluso plantear la opción de incluir un GPS o una cámara para detectar los obstáculos y saber qué camino escoger de una forma más eficiente.
- Sustituir las baterías recargables por una única batería más potente para que sea más sencillo de cargar.
- Realizar un diseño con una impresora 3D para otorgarle una flexibilidad total a la elaboración del chasis del prototipo. De esta forma se podría reducir mucho el peso del prototipo lo que optimizaría su consumo de batería. Además, cabe destacar que también se podría realizar una carcasa externa de cara a mejorar la estética del robot aspirador.

10. GLOSARIO

Open source: código abierto.

SLAM (*Simultaneous Location and Mapping*): localización simultánea y mapeado.

MIT (*Massachusetts Institute of Technology*): Instituto de Massachusetts de Tecnología.

I+D: Investigación y Desarrollo.

IR: infrarrojo.

US: ultrasonido.

LED (*Light Emitting Diode*): diodo emisor de luz.

App: aplicación diseñada para ser usada en móviles, tablets y otros dispositivos.

SRAM (Static Random Access Memory): memoria estática de acceso aleatorio.

EEPROM (Electrically Erasable Programmable Read-Only Memory).

RAM (Random Access Memory): memoria de acceso aleatorio.

SD (Secure Digital): tarjeta de memoria para dispositivos portátiles.

GPIO (General Purpouse Input/Output): pines que sirven tanto de *inputs* como de *outputs*.

IoT (*Internet of Things*): Internet de las Cosas.

STEM (*Science Technology Engineering and Maths*): Acrónimo que designa las disciplinas de Ciencias, Tecnología, Ingeniería y Matemáticas.

IDE (*Integrated Development Environment*): entorno de desarrollo integrado.

PWM (*Pulse Width Modulation*): modulación por ancho de pulsos.

I2C (*Inter Integrated Circuit*): circuito inter-integrado que establece un protocolo de comunicación en serie.

DC (*Direct Current*): corriente continua.

PSD (*Position Sensitive Sensor*): sensor de posición sensible.

TFG: Trabajo de Fin de Grado.

GPS (*Global Positioning System*): Sistema de Posicionamiento Global.

CEO (*Chief Executive Officer*): director ejecutivo de una organización o institución.

11. BIBLIOGRAFÍA

[1] "Open Source Initiative", Septiembre 2012. [En línea]. Available: https://opensource.org/history. [Último acceso: 26 Marzo 2018].

[2] "The Open Source Way", [En línea]. Available: https://opensource.com/open-source-way. [Último acceso: 26 Marzo 2018].

[3] C.-H. Kuo, "Pneumatic Sensor: A complete coverage improvement approach for robotic cleaners", IEEE, vol. 60, nº 4, p. 19, 4 Abril 2011.

[4] S. Riisgard y M. Blas, "SLAM for Dummies: A tutorial approach to simultaneous location and mapping", 2005. [En línea]. Available: https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf. [Último acceso: 27 Marzo 2018].

[5] "iRobot,History"» 2017. [En línea]. Available: http://www.irobot.com/About-iRobot/Company-Information/History.aspx . [Último acceso: 27 Marzo 2018].

[6] "¿Qué Roomba comprar?" [En línea]. Available: http://aspirame.es/que-roomba-comprar. [Último acceso: 27 Marzo 2018].

[7] T. E. Kurt, «Reading the Roomba sensors», de "Hacking Roomba", Indianapolis, Wiley Publishing Inc, 2007.

[8] "iRobot introduces Roomba (TM) Intelligent FloorVac-the first automatic floor cleaner in the U.S.", The Industrial Robot, 2003. Available: http://media.irobot.com/2002-09-18-iRobot-Introduces-Roomba-Intelligent-FloorVac-The-First-Automatic-Floor-Cleaner-In-The-U-S. [Último acceso: 18 Septiembre 2018].

[9] E. Ackerman y E. Guizzo, "iRobot Brings Visual Mapping and Navigation to the Roomba 980", IEEE Spectrum, 15 Septiembre 2015. [En línea]. Available: https://spectrum.ieee.org/automaton/robotics/home-robots/irobot-brings-visual-mapping-and-navigation-to-the-roomba-980 . [Último acceso: 27 Mazro 2018].

[10] "Cecotec, Serie Conga", [En línea]. Available: https://cecotec.es/conga/robots-aspiradores . [Último acceso: 27 Marzo 2018].

[11] "Comparativa robots aspiradores ILife", 2 Noviembre 2017. [En línea]. Available: https://www.robotaspiradorya.com/ilife/. [Último acceso: 27 Marzo 2018].

[12] "LG aspirador Hombot" [En línea]. Available: https://www.elcorteingles.es/lg/electrodomesticos/info/aspirador-hombot/. [Último acceso: 27 Marzo 2018].

[13] "Dyson 360 Eye" [En línea]. Available: https://www.dyson.es/aspiradoras/robot/dyson-360-eye.aspx. [Último acceso: 29 Marzo 2018].

[14] "My life, my neato". [En línea]. Available: https://www.neatorobotics.com/es/. [Último acceso: 29 Marzo 2018].

[15] “Cecotec Conga Excellence 990” [En línea]. Available: https://www.amazon.es/aspirador-Excellence-Programable-limpieza-Silencioso/dp/B01MUGXRT9. [Último acceso: 22 Abril 2018].

[16] “ILife A6” [En línea]. Available: https://www.amazon.es/ILIFE-Aspirador-silencioso-limpieza-Alfombras/dp/B06WD32HYR. [Último acceso: 22 Abril 2018].

[17] “LG VSR9640PS - Hombot Turbo Serie 12” [En línea]. Available: https://www.amazon.es/LG-VSR9640PS-aspirador-vigilancia-metalizado/dp/B01M9FZCO5. [Último acceso: 22 Abril 2018].

[18] “Botvac D7 connected” [En línea]. Available: https://www.neatorobotics.com/es/robot-vacuum/botvac-connected-series/botvac-d7-connected/. [Último acceso: 22 Abril 2018].

[19] “iRobot Roomba 980” [En línea]. Available: https://www.amazon.es/iRobot-Roomba-980-aspiradora-robotizada/dp/B017WD5KZM. [Último acceso: 22 Abril 2018].

[20] “Dyson 360 Eye” [En línea]. Available: https://www.amazon.es/Dyson-360-Eye-Nickel-Blue/dp/B017SQEHBG/. [Último acceso: 22 Abril 2018].

[21] “Arduino vs Raspberry Pi” 28 Marzo 2016. [En línea]. Available: https://aprendiendoarduino.wordpress.com/2016/03/28/arduino-vs-raspberry-pi/. [Último acceso: 2 Junio 2018].

[22] “Arduino UNO datasheet”. [En línea]. Available: https://www.farnell.com/datasheets/1682209.pdf.

[23] “Raspberry Pi 3 Model B+ Datasheet”. [En línea]. Available: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

[24] “Arduino UNO A000066 - Placa con microcontrolador basada en el ATmega328”. [En línea].Available: https://www.amazon.es/iRobot-Roomba-980-aspiradora-robotizada/dp/B017WD5KZM. [Último acceso: 17 Septiembre 2018].

[25] “Raspberry Pi 3 Modelo B - Placa base (1.2 GHz Quad-core ARM Cortex-A53, 1GB RAM, USB 2.0)” [En línea]. Available: https://www.amazon.es/Raspberry-Pi-Modelo-Quad-core-Cortex-A53/dp/B01CD5VC92. [Último acceso: 17 Septiembre 2018].

[26] “Arduino Nano datasheet”. [En línea]. Available: http://www.farnell.com/datasheets/1682238.pdf. [Último acceso: 29 Marzo 2018].

[27] “Arduino Mega datasheet”. [En línea]. Available: https://www.robotshop.com/media/files/pdf/arduinomega2560datasheet.pdf. [Último acceso: 29 Marzo 2019].

[28] T. Hirzel, “Tutorial de PWM” [En línea]. Available: https://www.arduino.cc/en/Tutorial/PWM. [Último acceso: 29 Marzo 2018].

[29] L. Llamas, “El bus I2C en Arduino” 16 Mayo 2016. [En línea]. Available: https://www.luisllamas.es/arduino-i2c/. [Último acceso: 29 Marzo 2018].

[30] L. Llamas, “Qué son y cómo usar interrupciones en Arduino” 28 Abril 2016. [En línea]. Available: https://www.luisllamas.es/que-son-y-como-usar-interrupciones-en-arduino/. [Último acceso: 21 Julio 2018].

[31] “Adquisición de datos con Arduino I: Tiempo de muestreo y Resolución” 15 Febrero 2015. [En línea]. Available: http://booleanbite.com/web/adquisicion-de-datos-con-arduino-i-tiempo-de-muestreo-y-resolucion/. [Último acceso: 8 Agosto 2018].

[32] J. Boxall, «Chapter 12: Motors and movement,» de “Arduino Workshop”, San Francisco, No Starch Press, 2013.

[33] “Arduino Motor Guide” [En línea]. Available: https://www.circuito.io/blog/arduino-motor-guide/. [Último acceso: 31 Marzo 2018].

[34] «El módulo controlador de motores L298N» [En línea]. Available: https://www.prometec.net/l298n/. [Último acceso: 29 Marzo 2018].

[35] «L298N datahseet». [En línea]. Available: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf. [Último acceso: 29 Marzo 2018].

[36] “Sharp, distance measuring sensor unit” [En línea]. Available: https://www.pololu.com/file/0J713/GP2Y0A41SK0F.pdf. [Último acceso: 8 Junio 2018].

[37] “Ultrasonic Ranging Module HC-SR04 datasheet”. [En línea]. Available: https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf. [Último acceso: 17 Septiembre 2019].

[38] “Obstacle Avoidance Sensor Module”. [En línea]. Available: https://www.sunfounder.com/obstacle-avoidance-sensor-module.html. [Último acceso: 18 Septiembre 2019].

[39] “Sharp GP2Y0A21YK datasheet”. [En línea]. Available: https://www.sparkfun.com/datasheets/Components/GP2Y0A21YK.pdf. [Último acceso: 17 Septiembre 2019].

[40] “Directiva 91/250/CEE del Consejo, de 14 de mayo de 1991, sobre la protección jurídica de programas de ordenador” [En línea]. Available: http://booleanbite.com/web/adquisicion-de-datos-con-arduino-i-tiempo-de-muestreo-y-resolucion/. [Último acceso: 17 Septiembre 2018].

[41] F. Andrades “¿Qué es el Hardware libre?” 6 Junio 2013. [En línea]. Available: https://www.eldiario.es/turing/Hardware-Libre_0_139986451.html. [Último acceso: 22 Junio 2018].

[42] “Tienes un 'espía' en el hogar: las Roomba ahora venderán los planos de tu casa” Available: https://www.eldiario.es/turing/Hardware-Libre_0_139986451.html. [Último acceso: 22 Junio 2018].

[43] D. Sarabia “Roomba dice ahora que no va a vender los planos de tu casa a Google o Amazon” 31 Julio 2017. [En línea]. Available: https://www.eldiario.es/tecnologia/Resulta-ahora-aspiradores-Roomba-venderan_0_670883052.html. [Último acceso: 22 Julio 2018].

12. ANEXOS

12.1 Código

Aquí se incluye el código final que gobierna el prototipo V2.0

```
int EchoR =18;
int TrigR =36;
int EchoC = 14;
int TrigC =15;
int EchoL = 17;
int TrigL =16;
int in1 = 7;
int in2 = 8;
int in3 = 9;
int in4 = 10;
int ENI = 5;
int END= 6;
int ABS = 200;
int collisionPinL=3;
int collisionPinC=2;
int collisionPinR=19;
int IRPinL=42;
int IRPinR=44;
int rightDistance = 0,leftDistance = 0,centerDistance = 0 ;

void mForward(){ //Funcion para que el robot se desplace hacia delante
    analogWrite(ENI,ABS);
    analogWrite(END,ABS);
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    digitalWrite(in3,HIGH);
    digitalWrite(in4,LOW);
    Serial.println("go forward!");
}

void mBackward(){//Funcion para que el robot se desplace hacia atras
    analogWrite(ENI,ABS);
    analogWrite(END,ABS);
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    digitalWrite(in3,LOW);
    digitalWrite(in4,HIGH);
    Serial.println("go backward!");
}
```

```
void mChoque(){//Funcion para que el robot retroceda en caso de choque
    analogWrite(ENI,ABS);
    analogWrite(END,ABS);
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    digitalWrite(in3,LOW);
    digitalWrite(in4,LOW);
    Serial.println("Choque!");
}

void mLeft(){//Funcion para que el robot se desplace hacia la izquierda
    analogWrite(ENI,ABS);
    analogWrite(END,ABS);
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    digitalWrite(in3,HIGH);
    digitalWrite(in4,LOW);
    Serial.println("go left!");
}

void mRight(){//Funcion para que el robot se desplace hacia la derecha
    analogWrite(ENI,ABS);
    analogWrite(END,ABS);
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    digitalWrite(in3,LOW);
    digitalWrite(in4,HIGH);
    Serial.println("go right!");
}

void mStop()//Funcion para que el robot pare
{
  digitalWrite(ENI,LOW);
  digitalWrite(END,LOW);
  Serial.println("Stop!");
}
```

```
int distanceTest(int trigPin,int echoPin)
//Función para medir la distancia con ultrasonidos
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(20);
  digitalWrite(trigPin, LOW);

  float Fdistance = pulseIn(echoPin, HIGH);
  int distance= Fdistance/58;
  return distance;
}

void golpe() {
// Choque o desnivel
  mChoque();
  delay(500);
  Serial.println("     Interrupción     ");
}

void setup()
{
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);
  pinMode(ENI,OUTPUT);
  pinMode(END,OUTPUT);
  pinMode(EchoL, INPUT);
  pinMode(TrigL, OUTPUT);
  pinMode(EchoC, INPUT);
  pinMode(TrigC, OUTPUT);
  pinMode(EchoR, INPUT);
  pinMode(TrigR, OUTPUT);
  pinMode(TrigR, OUTPUT);
  pinMode(IRPinR,INPUT);
  pinMode(IRPinL,INPUT);
  Serial.begin(9600);
  pinMode(collisionPinL,INPUT_PULLUP);
  pinMode(collisionPinC,INPUT_PULLUP);
  pinMode(collisionPinR,INPUT_PULLUP);
  attachInterrupt(collisionPinL,golpe, RISING); // Pin de interrupción
  attachInterrupt(collisionPinR,golpe, RISING);
  attachInterrupt(collisionPinC,golpe, RISING);
}
```

```
void loop()
{
// Se leen los sensores de colision y el Sharp
   int obstacleL=digitalRead(collisionPinL);
   int obstacleC=digitalRead(collisionPinC);
   int obstacleR=digitalRead(collisionPinR);
   float volts=analogRead(1)*0.0048828125;
   int height = 13*pow(volts, -1);
   Serial.println(" HEIGHT           ");
   Serial.println(height);

   if ((obstacleL==0)&&(obstacleC==0)&&(obstacleR==0)&&(height<=6)){
   // Si los sensores de colision y el Sharp no detectan nada el robot
lee la distancia del US central
       centerDistance=distanceTest(TrigC,EchoC);
       Serial.println(" Center distance          ");
       Serial.println(centerDistance);

       if (centerDistance<=20){
       // Si la distancia es menor de 20, el robot lee la distancia de
los US izquierda y derecha y de los dos IR de proximidad
            mStop();
            delay(500);
            leftDistance=distanceTest(TrigL,EchoL);
            rightDistance=distanceTest(TrigR,EchoR);
            Serial.println(" Right distance           ");
            Serial.println(rightDistance);
            Serial.println(" Left distance          ");
            Serial.println(leftDistance);
            int obstacleIRR=digitalRead(IRPinR);
            int obstacleIRL=digitalRead(IRPinL);
            Serial.println(" obstacle IR Right          ");
            Serial.println(obstacleIRR);
            Serial.println(" obstacle IR Left          ");
            Serial.println(obstacleIRL);

            if ((obstacleIRR==LOW)||(obstacleIRL==LOW)) {
            //Si uno de los IR está detectando un obstáculo, el robot
retrocede
               mBackward();
               delay(360*4);
```

```
            }else if ((leftDistance>rightDistance)){
            // Si la distancia hasta un obstáculo es mayor a la
izquierda se va a la izquierda
               mLeft();
               delay(360);

            }else if((rightDistance>leftDistance)){
            // Si la distancia hasta un obstáculo es mayor a la
      derecha se va a la derecha
              mRight();
              delay(360);
            }

        } else {
        // Si el sensor de US central no detecta nada a menos de 20,
el robot avanza de frente
          mForward();
       }

   } else{
    // Si uno de los sensores de colision o el Sharp detectan algo
     golpe();
   }
}
```

12.2 Esquema Electrónico del Prototipo Final

El siguiente esquema corresponde al circuito electrónico de los prototipos V1.3 y V2.0

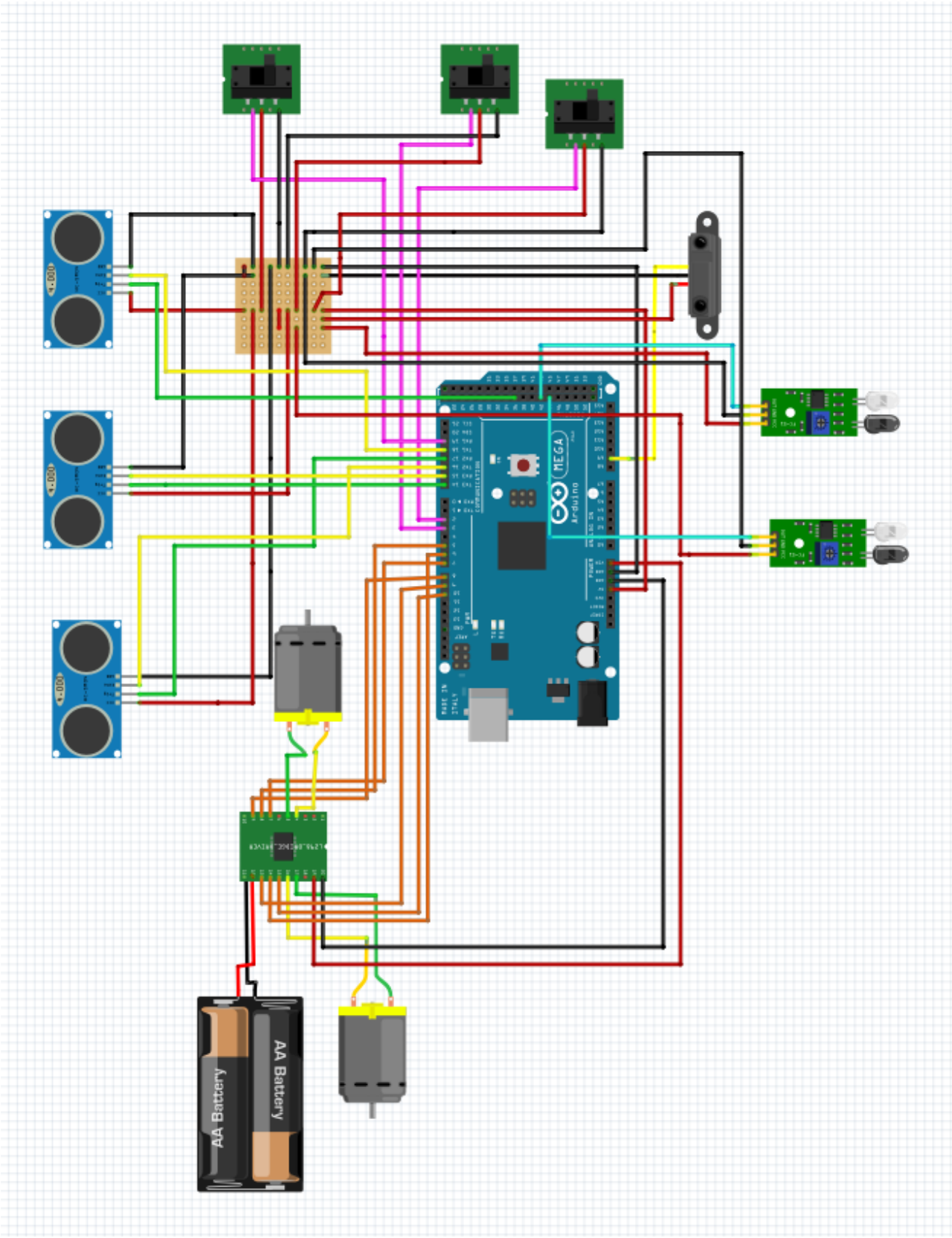


Ilustración 46 Esquema ampliado de los prototipos V1.3 y V2.0