UNIVERSIDAD POLITECNICA DE VALENCIA

UNIVERSIDAD POLITECNICA DE VALENCIA

Escuela Técnica Superior de Ingeniería del Diseño

PROYECTO FINAL GRADO SISTEMAS INTEGRADOS CON ARDUINO

Alumno: Mohammed El Yakouti

Director del proyecto: Leopoldo Armesto Angel

1. MEMORIA DESCRIPTIVA

1.1 ANTECEDENTES

1.1.1. Primeros coches teledirigidos

En la década de1950, muchos aficionados empezaron a agregar motores para modelos de coche fabricados por plástico, metal y seda, anqué los motores pequeños de nitro-metano estaban disponible en 1940.

La tecnología de radio control se utilizó en la segunda guerra mundial pero no ha sido una apuesta segura hasta la llegada del transistor, después se evoluciono la electrónica con sistemas más pequeños, más simples y con mucho menos consumo.

En la década de1970, vio el surgimiento del coche teledirigido en EE. UU con escala (1/8) que fue elegido para los primeros coches que estaban destinados a evolucionar el mercado controlado por EE. UU. Mediante empresas como" Wencon y Sistemas de Delta". Siete años después se celebró el primer campeonato mundial en California donde participaron europeos de varios países, cuya experiencia les embarco en la aventura de la industria del automóvil con el objetivo del segundo mundial que se celebró dos años después en GINEBRA y lo gano Phill BOOTH que se llevó la corona a reino unido.

Con los años se han ido mejorando tanto motores como baterías que permitieron a los coches con motores eléctricos ofrecer un rendimiento cerca de los modelos térmicos.

1.1.2. Primer robot de la historia y abuelo del coche autónomo

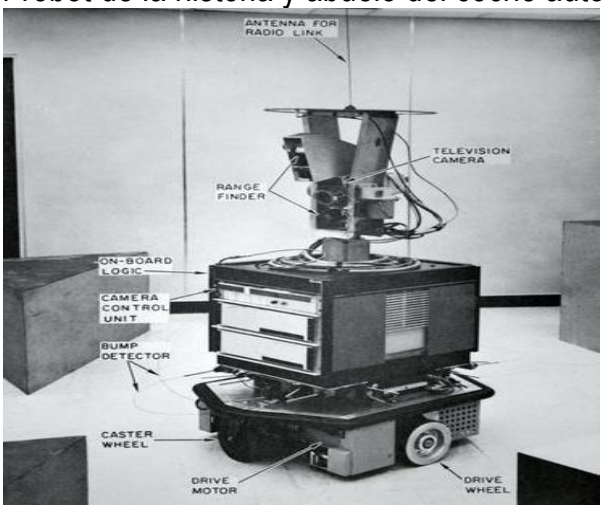


IMAGEN 1 : "SHAKEY" PRIMER ROBOT INTELIGENTE

En la década de 1960, nació el primer robot inteligente cuyo nombre "Shakey", en forma de lavadora, con una altura de dos metros, equipado con una cámara de televisión y otra telemétrica para percibir el entorno y ruedas con piernas para moverse además de antena de radio, era capaz de mover lentamente por una sala planeando su propia ruta evitando obstáculos y navegando autónomamente

Su primer nombre fue un autómata móvil para reconocimiento, ya que un robot será ciencia ficción así que no parecía bien pedir dinero para hacer ciencia ficción.

"Charles Rosen,1960 "fue el impulsador de creación de este autómata móvil inteligente.

Anqué parecía a los robots humanoides de hoy en día, no tenía celebro puesto, pero la electrónica de su cuerpo servía para controlar motores y sensores y para comunicarse con un ordenador para procesar la información que era de tamaño de un armario un "PDP-10" de la compañía DEC. [1]

[1] eldiario.es,Cristina Sanchez,2017

1.2. INTRODUCCION

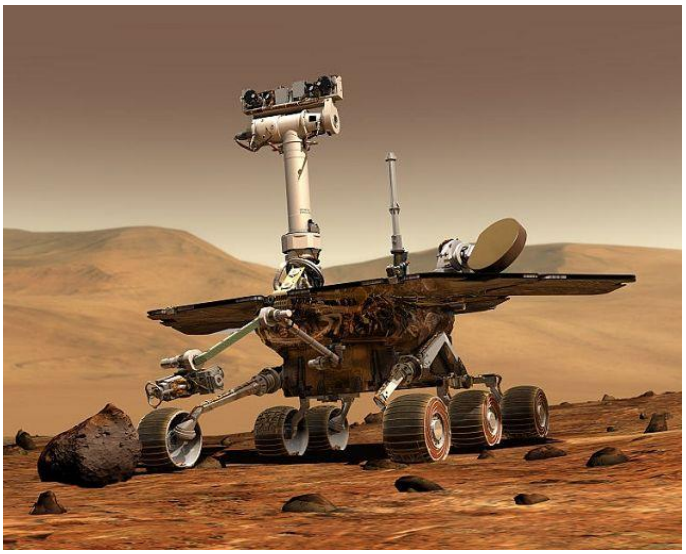


IMAGEN 2: ROBOT NASA

La teoría y aplicación de sistemas multi-sensor en diversas aplicaciones y la gran ayuda observada al utilizarlos en problemas tales como reconocimiento y explorar zonas con difícil acceso como el robot de la NASA en la "Imagen2" o zonas radioactivas, ayuda médica o aplicaciones militares, etc., como vemos se pueden utilizar en un gran variedad de aplicaciones lo que provoca un creciente interés en multitud de investigares en diversas áreas de conocimiento (robótica, estadística, inteligencia artificial ,etc.)
Por estos fines se infiere la necesidad de mejorar la calidad de la información externa recibida en cualquier aplicación, utilizando varios sensores en vez de un sensor por muy preciso que sea, lo que asegura la fiabilidad de los datos obteniendo varios para analizar, así logramos mejorar la interpretación de lo que está siendo observado y ayuda a mejorar el tiempo de respuesta, este será el cometido de un robot multisensorial.

1.3. OBJETIVO GENERAL

El propósito de Nuestro proyecto es el diseño y control de un Robot autómata teledirigido a base de Arduino.
El objetivo es crear un prototipo de un Robot controlado mediante un software creado en "Android".
El prototipo puede ser conducido remotamente en las cuatro direcciones o moverse automáticamente calculando mejores rutas, siguiendo una línea negra o recorrer un laberinto evitando obstáculos y todo controlado atreves de un móvil con sistema "Android", es decir puede pues hacer diversas funciones, se podrá realizar en tiempo real y podemos ver los diferentes eventos que adquiere.

1.3.1. Objetivo específico

- ✓ Diseñar y crear un prototipo de robot manejable a través de Bluetooth y Arduino UNO ya que es un Hardware libre este microcontrolador, cosa que nos ayudara para la implementación
- ✓ controlar y verificar los sensores y eventos que interactúan mediante el Arduino y el Bluetooth.

1.4. FASES DEL PROYECTO

Fase1→ Robot seguidor de línea
Fase2→ Robot evita obstáculos
Fase3→ Control movimiento y velocidad del robot a través de Bluetooth
Fase4→ Control remoto de robot multifunción

1.5. MATERIALES

Antes de empezar a fabricar nuestro robot, debemos de saber cuáles son los diferentes elementos y componentes que los componen.
En este apartado intentaremos describir los componentes utilizados en nuestro proyecto ya que, al haber tanta diversidad de montajes, no se puede explicar el funcionamiento de un componente u otro, ya que nunca se acabaría, pero si podemos explicar las partes genéricas de este tipo de montajes ya que no varían con la multitud de configuraciones que existen.

1.6. HARDWARE

En este apartado damos a conocer todos los elementos de Hardware que componen nuestro proyecto, donde explicamos más adelante en cada fase del proyecto las características y funcionamiento de cada dispositivo utilizado.

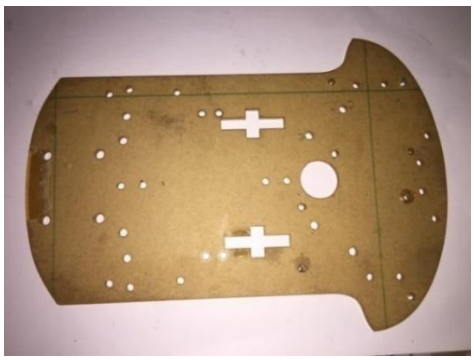


IMAGEN 3: BASE ROBOT

Chasis del robot

Este cuerpo tiene que ser capaz de mover y suportar el resto de componentes del robot, por supuesto que podemos realizar uno a medida ya sea e plástico madera u otros materiales más ligeros que le facilitan el desplazamiento y que lo mantenga en equilibrio y permita cambiar de dirección.

IMAGEN 4: TALADRO

Taladro

Para facilitar colocación de diferentes dispositivos de nuestro robot.

IMAGEN 5: RUEDA LOCA

Rueda loca

Es una rueda sin tracción que puede girar en cualquier sentido libremente y que suele estar situada en la parte inferior de las estructuras, suelen estar en sillas de oficina, carros de compra, etc.

IMAGEN 6: SOPORTE MOTOR

Soportes motores

Soporte de aluminio especialmente diseñada para sujetar motores de corriente continua de 6 a 12V son universales y pueden sujetar el motor

En configuración superior como en configuración inferior.

IMAGEN 7: TORNILLOS Y TUERCAS

Tornillos y tuercas

Tornillo M3 de 6mm y 10mm se venden en bolsas de 25unidades.

Tuercas metálicas de M3 de 3mm se implementan junto a los tornillos para crear estructuras y fijar componentes en el robot.

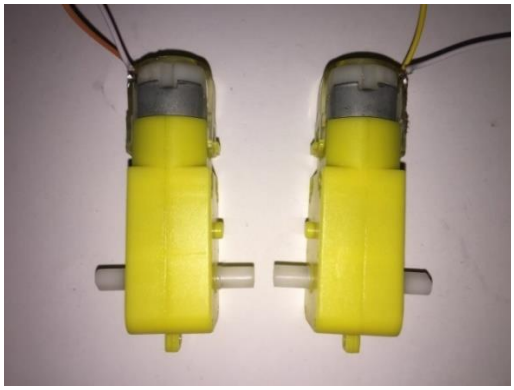


IMAGEN 8: MOTORES DC

Motores DC

Convierte la energía eléctrica en energía mecánica rotacional, es un tipo de motores básicos, es decir es el elemento que va a proporcionar tracción a nuestro robot.

Tamaño del motor:
70mm x 22mm x 18mm
Motor Peso: 50g
Tensión entre 3 V y 12 V (recomendado 6 a 8 voltios).

IMAGEN 9: RUEDAS ROBOT

Ruedas

Para facilitar el movimiento de nuestro robot se acoplan fácilmente a nuestros motores:

Diámetro de la llanta: 65mm

Reductora: 48: 1

Ancho neumático: 25mm

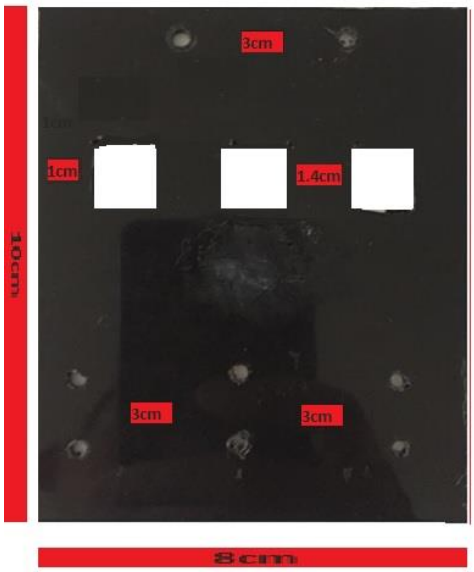


IMAGEN 10: SOPORTE SENSORES TCRT5000

Soporte sensores

Es una placa de plástico construida para sujetar tres sensores con el objetivo de seguir una línea de color cuyas dimensiones están anotadas en la foto.

IMAGEN 11: SEPARADORES

Separadores

Separador metálico hexagonal M3 macho se venden en bosas de 25unidades se usan para fijar circuitos a los robots también se usan en robótica para dar soportes a piezas y componentes de los robots.

IMAGEN 12: SENSOR TCRT5000

Sensor TCRT500

Sensor óptico reflexivo que se compone de un emisor de luz infrarroja y un fototransistor, este último detecta la luz que es reflejada cuando un objetivo pasa enfrente del sensor.

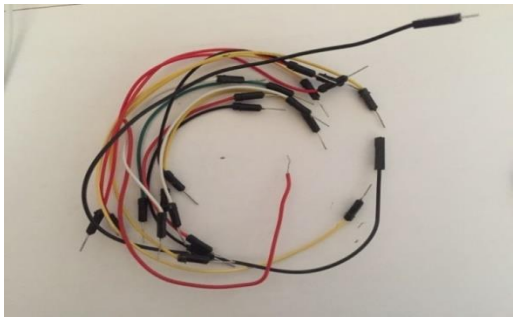


IMAGEN 13: CABLES MACHO-MACHO

Cables macho-machos

Sirven para conexiones y alimentación de sensores.

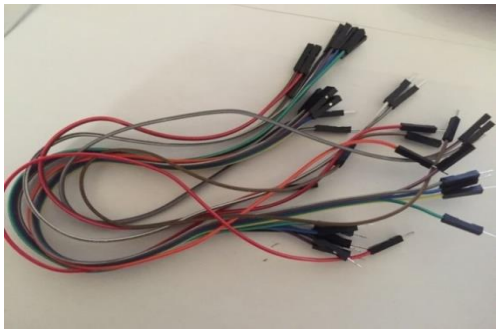


IMAGEN 14: CABLES MACHO-HEMBRA

Cables macho-hembra

Sirven para conectar sensores con Arduino

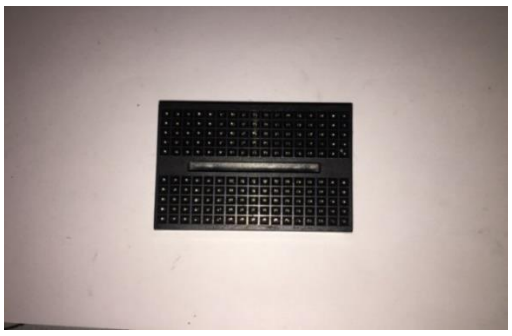


IMAGEN 15: PROTOBOARD-MINI

Protoboard-Mini

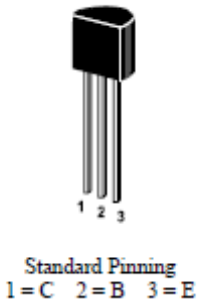
Para conexiones y circuitos adicionales.

Resistencia

Resistencias 330KΩ

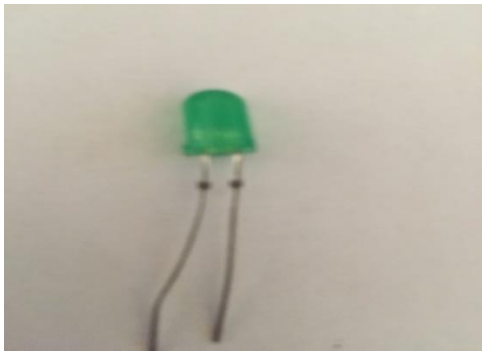
IMAGEN 16: RESISTENCIA

Transistor Bipolar



Transistor BC559B

IMAGEN 17: TRANSISTOR BC559



Led

Led para hacer pruebas y testeo.

IMAGEN 18: LED

Integrado l293D

El circuito integrado L293D se usa para controlar pequeños motores y actuadores de corriente continua, es bastante utilizado en robótica para controlar motores a pasos y de corriente continua.

IMAGEN 19: INTEGRADO L298D

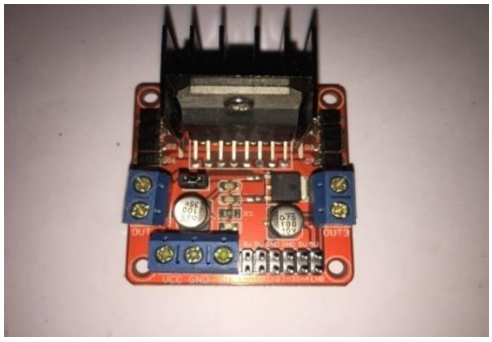


IMAGEN 20: DRIVER DE POTENCIA L298N

Módulo L298N

H-bridge nos permite controlar la velocidad y la dirección de dos motores de corriente continua gracias a su dos puentes-H, el nombre H viene dado por lo cuatro transistores que nos permiten invertir el sentido de la corriente así podemos controlar el giro del motor en ambas direcciones.

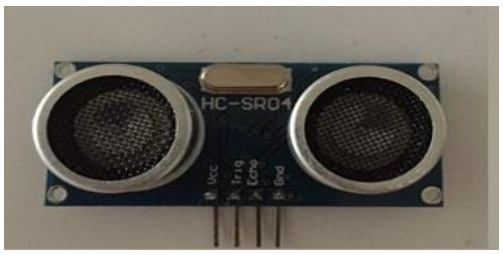


IMAGEN 21: SENSOR ULTRASÓNICO HC-SR04

Sensor ultrasónico HC-SR04

Es un módulo que incorpora dos transsductores de ultrasónico que se utilizan de manera conjunta para determinar la distancia exacta entre el sensor y un objeto colocado enfrente.

IMAGEN 22: SOPORTE SENSOR HC-SR04

Soporte sensores HC-SR04

Soporte para los sensores ultrasónico HC-SR04 que es fácil de fijar en el chasis del robot

IMAGEN 23: MODULO BLUETOOTH HC-06

Modulo Bluetooth HC-06

Es un módulo Bluetooth que puede actuar como esclavo, lo que nos ayudaría de forma fácil a comunicarnos con nuestro robot de forma inalámbrica.

Pila 9Voltios

Proporcionan energía fiable de manera prolongada para alimentar nuestra placa Arduino.

IMAGEN 24: PILA 9V

Conector Jack-macho

Sirve para alimentar la placa Arduino atreves de la pila de 9V.

IMAGEN 25: CONECTOR JACK-MACHO

Porta-pilas

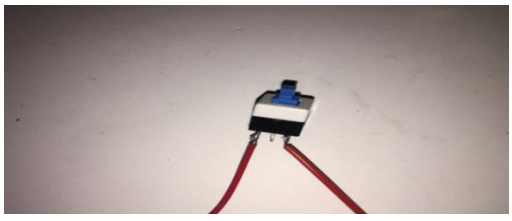
Es una porta pilas de cuatro pilas de 1.5Voltios equivale a una fuente de alimentación de 6Voltios suficiente para alimentar nuestros 2 motores.

IMAGEN 26: PORTA PILAS

Pilas de 1.5Voltios

Para alimentar el puente-H que a su vez alimenta los motores y da tracción a nuestro robot.

IMAGEN 27: PILAS 1.5V AA



Interruptor

Para poder apagar nuestro puente-H

IMAGEN 28: INTERRUPTOR

IMAGEN 29: ARDUINO UNO

Placa Arduino UNO

Placa electrónica con el microcontrolador de la marca Atmel: ATmega328, cuenta con 14pines digitales que pueden ser entrada o salidas de las cuales 6 se pueden utilizar como salidas PWM (modulación por ancho de pulsos) y 6 analógicas aparte toda la circuitería de soporte que incluye regulador de tensión un puerto –USB conectado a un módulo adaptador que permite programar el micro Atmel desde cualquier ordenador.

1.6.1. Un sistema electrónico

Es el conjunto de dispositivos o sensores dentro del campo de la ingeniería que se encargan de la aplicación de circuitería cuya función es procesamiento y control y almacenar información.

Los sensores obtienen información que puede consistir en imagen o datos en voz, números o música, receptor de radio, del mundo físico externo y la transforman en una señal eléctrica que puede ser manipulada por la circuitería interna de control.

Existen sensores de todo tipo:

Temperatura, Proximidad, Movimiento, Humedad, etc.

Los circuitos internos de un sistema electrónico son los encargados de procesar la señal eléctrica convenientemente.

La manipulación de la señal eléctrica dependerá tanto del diseño de los diferentes componentes hardware del sistema, como del conjunto lógico de instrucciones que dicho hardware sea capaz de ejecutar de forma autónoma.

Los actuadores transforman las señales eléctricas de salida de la unidad de control en magnitudes mecánicas, es decir que trasforma la señal eléctrica acabada de procesar por los circuitos internos en energía que actúa directamente sobre el mundo físico externo.

Por ultimo La fuente de alimentación que proporciona la energía necesaria para que se pueda realizar todo el proceso descrito de obtención:

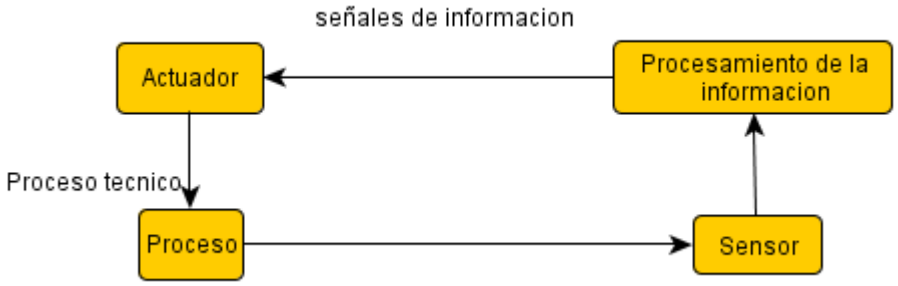


ILUSTRACIÓN 1: PROCESAMIENTO DE LA SEÑAL

Los microcontroladores son los nuevos conquistadores de nuestro moderno mundo. Están por todas partes, presentes en nuestro trabajo, casa y en nuestra vida. Los puedes encontrar controlando el funcionamiento de los ratones y teclados de las computadoras, microondas, televisores de nuestro hogar y también nuestros teléfonos.
Pero la invasión masiva de estos diminutos computadores da comienzo en el siglo XXI, que controlan la mayor parte de los aparatos que usamos los humanos en todos los ámbitos.
Un microcontrolador es un circuito integrado programable de alta escala de integración que es capaz de ejecutar las ordenes grabadas en su memoria además incorpora todos los dispositivos que configuran un controlador, así será el pequeño celebro de nuestros robots, que contiene todos los componentes fundamentales para procesar la información de los sensores y dar respuesta adecuada a cada orden recibida para actuar según lo programado.
A parte de las ventajas que tiene, sus prestaciones son un poco limitadas, además suele controlar una sola tarea, solo un programa podemos grabar en su memoria que controla el funcionamiento de una tarea determinada, las líneas de entradas y salidas del microcontrolador se conectan a los sensores y actuadores del dispositivo a controlar, debido a su pequeño tamaño, suele ir integrado en el propio dispositivo al que controla.
Los componentes de un microcontrolador son:

- ✓ Procesador o CPU (Unidad Central de Proceso) que se ocupa de procesar la información parcial y envía ordenes, también se compone de una unidad aritmética y un bus de datos para ejecutar el programa incorporado en el microcontrolador.
- ✓ Memoria RAM para contener los datos temporales requeridos para los cálculos.
- ✓ Memoria para el programa tipo ROM/EPROM/EEPROM/Flash, se utiliza para almacenar programas que controlan la aplicación a la que está dedicado el microcontrolador.
- ✓ Líneas de Entradas/Salidas para comunicarse con el exterior.
- ✓ Módulos para el control de periféricos.
- ✓ Temporizadores que generar unas señales de medida, con una alta precisión temporal.
- ✓ Puertos serie/paralelo.
- ✓ Convertidores Analógico/Digital.
- ✓ Convertidores Digital/Analógico.
- ✓ Generador de impulsos de reloj para sincronizar el funcionamiento de todo el sistema.

1.7. ARDUINO

1.7.1. Antecedentes

Arduino se creó como una herramienta para crear objetos interactivos sin la intervención de un especialista poniendo en las manos de cualquiera la capacidad de experimentar con la electrónica de una forma sencilla y práctica.

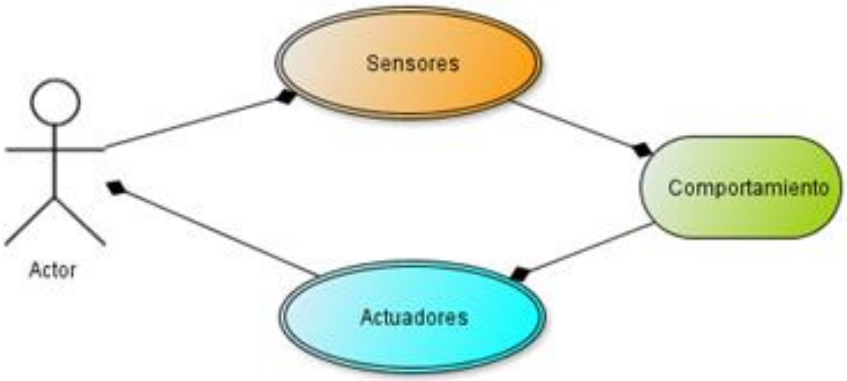


ILUSTRACIÓN 2: PROCESO INTERACTIVO

1.7.2. El origen de Arduino

Arduino, empezó como un proyecto de algunos estudiantes y su profesor "Massimo Banzi", quienes querían crear sus propias tarjetas de entrenamiento para el aprendizaje de la electrónica.

Debido a que las tarjetas que se encontraban en el mercado, en eso tiempos eran de un precio muy elevado.

La filosofía en la creación de estas tarjetas fue la de ser de hardware abierto y sobre todo de un precio asequible lo que facilita el aprendizaje de los estudiantes, así que se aseguraron de que todas las herramientas informáticas que permitían compilar los programas y crear la interfaz gráfica de la aplicación sean parte del dominio público.

El microcontrolador que cumplía con esos requisitos fue el AVR, fabricado por ATMEL que Fue un gran éxito, debido a que aquellos tiempos grandes partes de las herramientas informáticas que existían para este microcontrolador no eran muy usadas ni tampoco conocidas.[2]

[2] Noarduino,Wodpress, Abril26,2012

Actualmente, existen diversas herramientas informáticas para los microcontroladores en especial los del fabricante Atmel, Los programadores para estos microcontroladores tienen un mínimo precio y sobre todo, si se usa un BOOTLOADER para cargar el programa en el microcontrolador se evita usar el programador, además la forma de enviar la información desde el PC a la memoria del microcontrolador es fácil y rápida, similar "A descargar o pasar información a un celular por medio del puerto USB".

También existen contradicciones con los objetivos principales de la Plataforma Arduino.

Referente a un precio asequible para el estudiante se puede comprobar que el precio es muy pero muy superior a un microcontrolador, pero si consideramos que junto a Arduino va algún escudo (Shields), superara el precio del ARDUINO UNO, cosa que no tiene nada de ser asequible para los estudiantes.

Según la filosofía de ARDUINO, se recomienda comprar las placas originales y no la fabricadas en China ya que dichas placas se fabrican siguiendo normas ambientales. Pero revisando los proveedores que distribuyen las tarjetas ARDUINO, podemos darnos cuenta de que estas empresas envían a construir sus PCB en china. Lo cual queda demostrado que la verdadera filosofía del proyecto ARDUINO, es la de un negocio.

Por último, si analizamos el microcontrolador de 8 bits que usa ARDUINO, lo que implica condicionar nuestros diseños a una plataforma obsoleta. Debido a que los actuales avances tecnológicos nos exigen trabajar con microcontroladores de 32 o más bits, que nos permiten desarrollar proyectos actuales y novedosos.[3]

1.7.3. Introducción a Arduino

ILUSTRACIÓN 3: LOGO ARDUINO

Una plataforma de electrónica abierta no mucho más grande que una tarjeta de visita que tiene un microcontrolador programable, múltiples entradas/salidas analógicas y digitales, sirve para la creación de prototipos basada en software y hardware flexible y fácil de usar, basado en una sencilla placa de circuito impreso que contiene un microcontrolador de la marca "ATMEL" con varias entradas y salidas tanto digitales como analógicas , en un entorno basado en el lenguaje de programación Processing.

Con este dispositivo se logra conectar el mundo analógico con el digital controlando diversos sensores y actuadores físicos, como sensores, motores, servos, etc.

Se creó para aficionados, diseñadores, y cualquiera interesado en crear entornos u objetos interactivos.

[3] https://hipertextual.com/2011/02/open-hardware, Geraldine Juárez 11/02/11

1.7.4. Características placa Arduino

Se tratan de placas open hardware, es decir que se puede utilizar, modificar y distribuir dichos diseños sin tener que acudir al fabricante además de descargar el Software gratis.

1.7.5. Arduino hardware libre

También llamado "Open-Source" comparte muchos de los principios y metodologías del software libre, es decir puede tener acceso al diseño de esquemas del fabricante por ser estudiados, modificados, mejorados y compartidos con la comunidad Arduino.
Los principios de los productos físicos que se consideran como open hardware son los siguientes:

- ✓ Publicar la documentación completa del hardware, incluyendo los archivos de los diseños originales, que debe permitir su modificación y mejora.
- ✓ Ofrecer el software para el visionado, además los archivos de diseño y de la documentación, para que se pueda escribir el código open-source fácilmente.
- ✓ Ofrecer una licencia que permita producir derivados y modificaciones, por último, fabricación y venta de productos creados.
- ✓ La licencia no debe discriminar a ningún grupo o persona.
- ✓ No se puede limitar su uso únicamente para negocios o prohibir que sea utilizado Para investigación nuclear.
- ✓ El licenciamiento de la obra no puede depender de un producto en particular.
- ✓ La licencia no debe restringir otro hardware o software, es decir que no puede insistir en que otros componentes de hardware o software externos a los dispositivos sean también open-source.

Siguiendo este tipo de licencias, donde toda la información en el dominio público, todas las placas pueden ser construidas por cualquier usuario o bien comprarlas ya montadas.
Los ficheros de diseño de referencia (CAD), al estar disponibles bajo una licencia abierta, pueden ser libremente modificados y adaptados a las necesidades particulares y también el software tiene que estar disponible de forma gratuita.

1.7.6. Arduino software libre

Según la "free software fundatión", que es una organización encargada de fomentar el uso y desarrollo de software libre a nivel mundial un software para ser considerado libre debe respetar la libertar de los usuarios, eso significa:

- ✓ Libertad de ejecutar el programa con cualquier sistema operativo y por cualquier propósito sin tener que comunicarse con el de desarrollador.
- ✓ Libertad de estudiar cómo funciona interinamente un programa copiarlo modificarlo y adaptarlo a las necesidades particulares.
- ✓ La libertad de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie.

Resumiendo, el software libre es aquel software que da a los usuarios la libertad de poder copiarlo, modificarlo, mejorarlo y compartirlo sin tener que pedir ni pagar permisos al fabricante original ni a ninguna otra entidad.

1.7.7. Placa Arduino UNO

ILUSTRACIÓN 4: PINES TARJETA ARDUINO UNO

Como demuestra la "Ilustración 4", la placa Arduino puede tomar y procesar información del exterior o de su entorno a través sus pines de entrada que pueden ir conectados a toda una gama de sensores y actuadores, es una plataforma de Hardware libre que consiste en un placa con un microcontrolador del fabricante ATmel "AVR" y varios puertos de (entrada/salida) tanto digitales como analógicos así como salidas PWM y dos pines de comunicaciones ,para controlar y actuar sobre actuadores ,los Microcontroladores más usados de la familia ATmel son:

- ✓ ATmega1280 para mayor capacidad
- ✓ ATmega328 y ATmega168 para las placas básicas
- ✓ ATmega8 para las placas más antiguas

Todos esos microcontroladores incluyen un cargador de arranque (boot loader) para facilitar el trabajo con ellos.
El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino basado en **Wiring**.
Los proyectos hechos con Arduino pueden ejecutarse sin
El entorno de desarrollo Arduino basado en **Processing.**
necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software.[4]

[4] http://www.cortoc.com/2011/12/introduccion-arduino.html, Julio Roberto Letrán cardona

1.7.7.1. Processing

Un lenguaje de programación de código abierto, creado para diseñadores audiovisuales que desean crear asombrosos proyectos multimedia, interacciones e animaciones.
Fue desarrollado principalmente para servir como software de dibujo y enseñar los fundamentos de la programación en un contexto visual.
la web oficial donde se puede descargar el IDE y ver tutoriales y ejemplos.[5]

1.7.7.2. Wiring

Se sabe que en Processing es donde surge el lenguaje Wiring para microcontroladores, es una plataforma que nos permite la programación de entradas/salidas de un código abierto para explorar fácilmente las artes de la electrónica, los medios materiales, la enseñanza y el aprendizaje de la programación informática y creación de prototipos con electrónica.
A diferencia con Processing, en el IDE de Wiring encontramos un compilador GCC para C/C++ simplificado debido que no tenemos todas las características de este lenguaje.
Al utilizar Arduino, nos aparecerán las estructuras de nuestros skecths muy similares, ya que tienen casi las mismas funciones principales:
Setup (): función donde se inicializa las variables
draw(Wiring): función que funciona igual que la función Loop () (Arduino) o, mejor dicho, Arduino funciona igual que esta función siendo un bucle infinito.
La web oficial donde se puede descargar el IDE y ver tutoriales y ejemplos.[6]

1.7.8. Porque Arduino?

Hoy en día existen muchos microcontroladores y plataformas con microcontroladores disponibles para la computación física como, por ejemplo: Parallax Basic Stamp, BX-24 de Netmedia, et., además ofrecen funcionalidades similares. Todas estas herramientas organizan el complicado trabajo de programar un microcontrolador en paquetes fáciles de usar.

La simplicidad de Arduino en el proceso de trabajar con microcontroladores ofrece muchas ventajas respecto a otros sistemas para los estudiantes:

I. **Multi-Plataforma:** programar una placa Arduino requiere conectarla a un ordenador y utilizar el IDE de Arduino, cosa que facilita el software para la programación que funciona en los sistemas operativos Windows, Macintosh OSX y Linux, lo que le da ventaja contra de La mayoría de los entornos para microcontroladores que están limitados a Windows.

II. **Software ampliable y de código abierto:** Eso significa que el software Arduino está publicado bajo una licencia libre y preparada para recuperar el esquema original y modificarlo o ampliarlo por programadores experimentados.

[5] http://Processing.org.co/

[6] http://wiring.org.co/

III. **Comunidad:** Una comunidad es muy importante en este tipo de proyectos y gracias a la segunda ventaja que dio lugar a una rápida difusión de las placas Arduino por todo el mundo, lo que creo muchos foros y documentos en línea con el fin de utilizar la placa Arduino para ir solucionados problemas que van sugiriendo.

IV. **Simplicidad:** El entorno de programación de Arduino es fácil de usar para principiantes ya que fue diseñado para principiantes en la electrónica y además suficientemente flexible para usuarios avanzados.

V. **Hardware ampliable y de Código abierto:** Arduino está basado en los microcontroladores ATMEGA168, ATMEGA328 yATMEGA1280. Los planos de los módulos están publicados bajo licencia "Creative Commons", por lo que diseñadores de circuitos con experiencia pueden hacer su propia versión del módulo, ampliándolo u optimizándolo. Incluso usuarios relativamente inexpertos pueden construir la versión para placa de desarrollo para entender cómo funciona y ahorrar algo de dinero.[7]

VI. **Precio:** El coste de Las placas Arduino son más asequibles comparadas con otras plataformas con microcontroladores, la versión más popular que es el modelo UNO cuestan alrededor de 20-25 euros, también se venden clones auténticos a los originales con una diferencia de 10 euros, pero la principal diferencia es la calidad de la tarjeta.

1.7.9. Las diferentes placas Arduino

Con el fin de elegir una placa Arduino que cumpla con nuestros objetivos, debemos responder una serie de preguntas que agilizarán bastante la elección de una u otra placa:

- ✓ Obviamente lo primero es el tamaño que tendrá nuestro proyecto que estamos creando y cuales más importante el tamaño o las prestaciones.
- ✓ Dispongo de un programador de micros o necesitamos alimentar y programar con la propia placa.
- ✓ Arduino va a interactuar sólo con su propia circuitería o se comunicará con más dispositivos o placas.
- ✓ Finalmente, el precio.

Dicho esto, vamos a intentar seleccionar nuestra placa Arduino entre las diferentes placas que analizamos a continuación.

[7] http://arduinodhtics.weebly.com/iquestqueacute-es.html

1.7.10. TABLA COMPARATIVA DE PLACAS ARDUINO

Como se observa en la “tabla 1” siguiente, las características en color rojo son la clave a la hora de elegir una placa Arduino para nuestro proyecto.
Si queremos que nuestro proyecto saldrá barato la Arduino uno es ideal, en el caso de minimizar el sistema el Arduino Nano o Mini serán lo correcto, pero si queremos conectar muchos y diversos sensores y dispositivos en serie la Arduino Mega será perfecto, por último, si buscamos algo parecido a un ordenador a bordo el Yun es la opción más ideal.

TABLA COMPARATIVA DE PLACAS ARDUINO

Arduino	UNO R3	UNO R3 Ethernet	Leonardo	Esplora	Yun	Mega 2560	Mega ADK	Mini	Nano	DUE	Zero PRO
Características											
Microcontrolador	ATmega328P	ATmega328P	ATmega32u4	ATmega32u4	ATmega32u4	ATmega2560	ATmega2560	ATmega328P	ATmega328P	AT91SAM3X8E	ATSAMD21G18
Reloj	16 Mhz	16 Mhz	16 Mhz	16 Mhz	16 Mhz	16 Mhz	16 Mhz	16 Mhz	16 Mhz	84Mhz	48Mhz
Tensión alimentación	5V	5V	5V	5V	5V	5V	5V	5V	5V	3.3V	3.3V
Tensión Referencia	7-12V	7-12V	7-12V	7-12V	7-12V	7-12V	7-12V	7-9V	7-9V	5V	5V
Entradas/Salidas Digital	14/6	14/4	20/7	NO Tiene	20/7	54/15	54/15	14/6	14/6	54/12	14/12
Entradas/Salidas (PWM) Analógicas	6/0	6/0	12/0	No Tiene	12/0	16/0	16/0	8/0	8/0	12/2(DAC)	6/1(DAC)
Memoria Flash	32Ko	32Ko	32Ko	32Ko	32Ko	256Ko	256Ko	32Ko	32Ko	512Ko	256Ko
Memoria EEPROM	1Ko	1Ko	1Ko	1Ko	1Ko	4Ko	4Ko	1Ko	1Ko	No Tiene	16Ko
USB	USB-B	USB –B	Micro-USB	Micro-USB	Micro-USB	USB-B	USB-B & USB-A (android)	NO TIENE	Mini-USB	2 Ports Micro-USB	2 Ports Micro-USB
Puerto UART	1	1	1	No Tiene	1	4	4	No Tiene	1	4	2
Tarjeta SD	No Tiene	Tiene	No Tiene	No Tiene	Tiene	No Tiene	No Tiene	No Tiene	No Tiene	No Tiene	No Tiene
Ethernet	No Tiene	Tiene	No Tiene	No Tiene	Tiene	No Tiene	No Tiene	No Tiene	No Tiene	No Tiene	No Tiene
Wifi	No Tiene	No Tiene	No Tiene	No Tiene	Tiene	No Tiene	No Tiene	No Tiene	No Tiene	No Tiene	No Tiene
Tamaño	68x3mm	68x53mm	68x53mm	165x60mm	68x53mm	101x53mm	101x53mm	30x18mm	45x18mm	101x53mm	68x53mm
Precio	20,50€	26,99€	20,50€	39,99€	70€	35€	53,10€	14,50€	20,49€	39€	42,90€

TABLA 1: COMPARACIÓN DE PLACAS ARDUINO

1.7.11. Tarjetas Arduino destacadas

Arduino Mini:

ILUSTRACIÓN 5: TARJETA ARDUINO MINI

Es la placa más pequeña oficialmente de Arduino, ya que simplificada al máximum ni siquiera tiene puerto USB, mide tan sólo 30x18mm y permite ahorrar espacio en los proyectos que lo requieran.
Necesitas soldadora para soldar los pines.

Arduino Nano

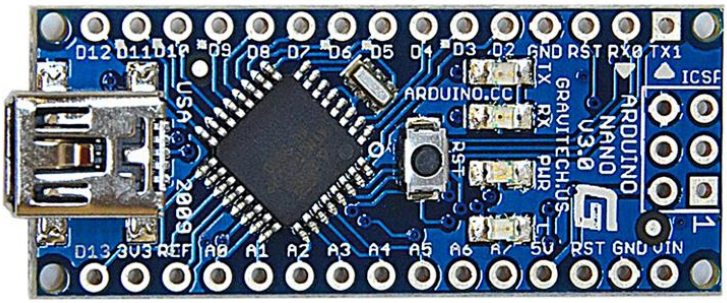


ILUSTRACIÓN 6: TARJETA ARDUINO NANO

El tamaño es una de sus ventajas en una restricción de espacio, además puede ser pinchado directamente sobre una Protoboard haciendo muy cómodo el prototipo al igual que el Arduino mini, también necesita soldadora.

Arduino UNO

ILUSTRACIÓN 7: TARJETA ARDUINO UNO

Esta tarjeta por excelencia es la más utilizada y recomendable para proyectos tecnológicos de robótica, ya que nos permite realizar proyectos con comodidad y solo sus números de entradas /salidas podría limitar el marco de proyectos convencionales, además su fuente de alimentación externa facilita la alimentación de la placa con una batería y un pequeño conector.

Arduino Mega

ILUSTRACIÓN 8: TARJETA ARDUINO MEGA

Es una versión mejor equipada de la Arduino Uno es con mucha diferencia el más potente y el que más puertos de Entradas/Salidas tiene, apto para trabajos algo más complejos, aunque tengamos que sacrificar un poco el espacio, cuenta con el microcontrolador Atmega1280 con más memoria para el programa, más RAM y más pines que el resto de los modelos, se puede alimentar a través del puerto USB o con una fuente de alimentación como batería.

Arduino Yun

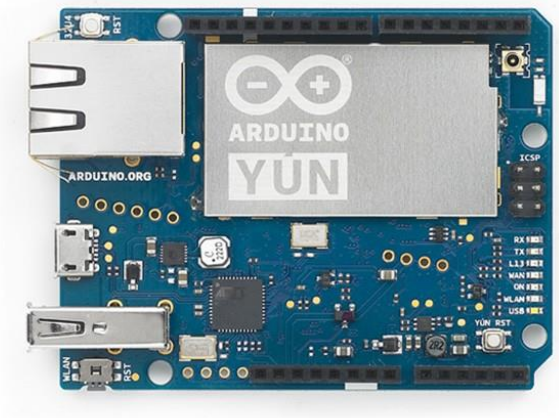


ILUSTRACIÓN 9: TARJETA ARDUINO YUN

Se considera la primera placa con WIFI integrado que combina con la potencia de Linux y la facilidad de uso de Arduino, tiene puerto para microSD y la red Ethernet/Wifi, ambos comunicados mediante un puente.

Se trata de una placa similar a Arduino UNO, pero es más complejo teniendo conexión Ethernet, Wifi, USB y microSD sin necesidad de agregar o comprar Shields aparte. Contiene 20 pines digitales, 7 pueden ser usados en modo PWM y 12 como analógicos.

1.8. SOFTWARE

1.8.1. IDE ARDUINO

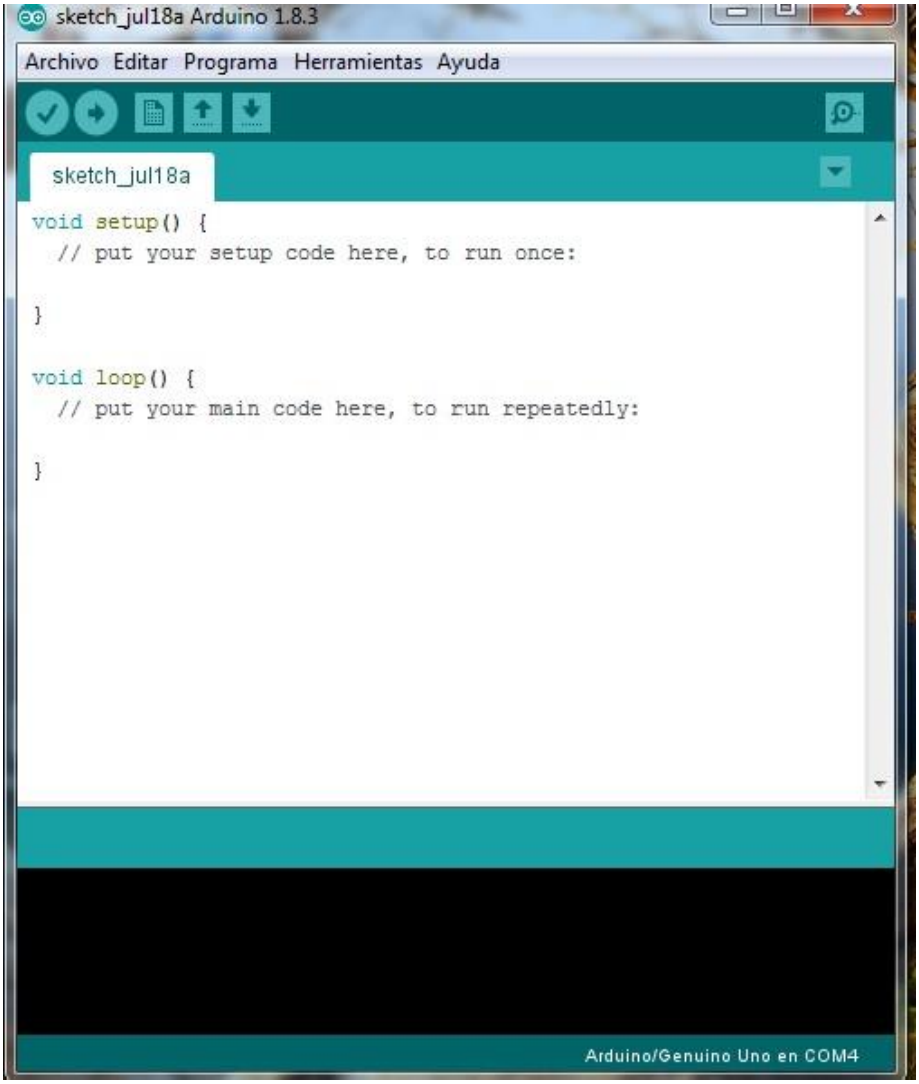


ILUSTRACIÓN 10: SKETCH IDE ARDUINO

El software de Arduino (IDE) es el que nos permite escribir el programa y cargarlo en nuestra placa Arduino, lo cual hay dos formas de usarlo:

- Si deseamos trabajar sin conexión, debemos descargar e instalar la versión más reciente de Arduino desde la página Arduino.[8]

- Si tenemos una conexión de internet fiable, podemos usar el IDE online ya que siempre tendremos la última versión más actualizada del IDE, además de tener disponible nuestros códigos en la nube y poder acceder a su contenido a través de cualquier dispositivo y todo eso sin la necesidad de instalar actualizaciones nuevas. A continuación, Link para acceder al IDE online:[9]

[8] https://www.arduino.cc/en/Main/Software

[9] https://create.arduino.cc/

Descarga e instalación

Para descargar el IDE en nuestro sistema operático solo tendremos que seguir unos sencillos pasos:

I. Descargar el programa gratuito Arduino IDE haciendo clic en el enlace anterior o dirigir a la web después Software.

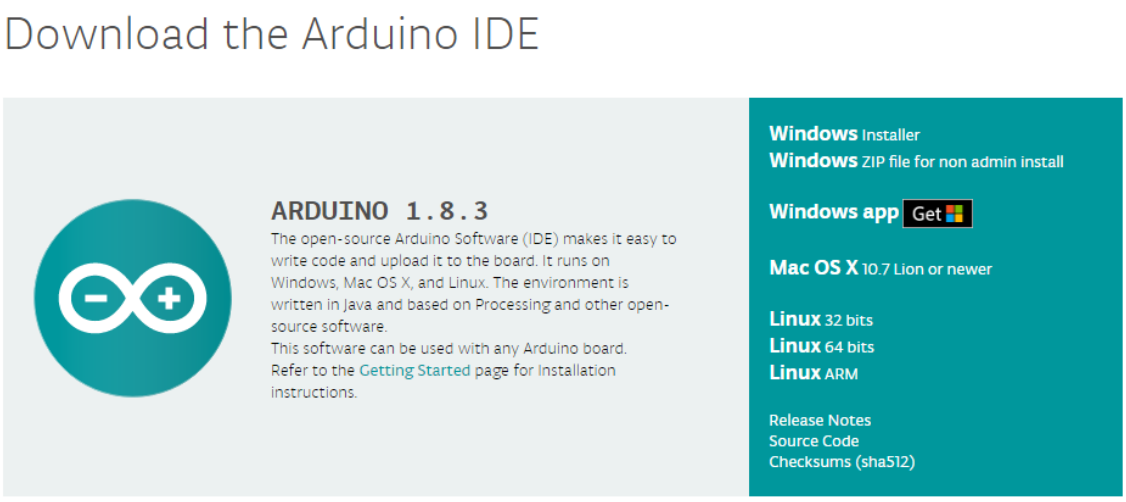


ILUSTRACIÓN 11 DESCARGAR SOFTWARE IDE ARDUINO

II. A continuación, podemos elegir entre Arduino web, trabajar sin necesidad de descargar u optar por descargar Arduino IDE solo tenemos que elegir nuestro sistema operativo y seguir las instrucciones.

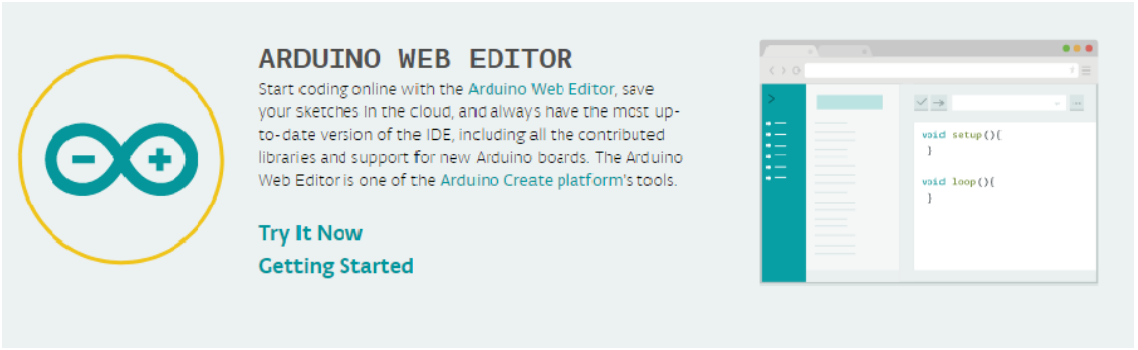


ILUSTRACIÓN 12: LINK WEB IDE ARDUINO

1.9. FRITZING

Es una iniciativa de hardware de código abierto que hace que la electrónica sea accesible como material creativo para todos, que nos permita hacer diagramas, circuitos electrónicos y montajes, también sirve para hacer circuitos impresos PCB.
También nos permite documentar nuestros prototipos y compartirlos con otros.

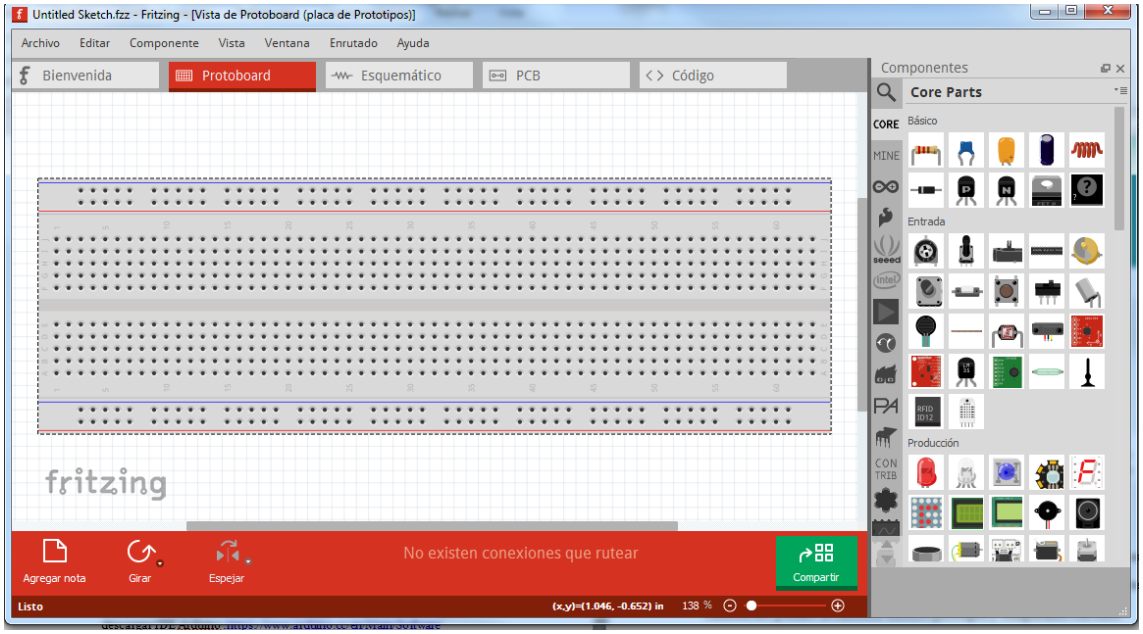


ILUSTRACIÓN 13: FRITZING

Descarga e instalación

Para descargar Fritzing, se puede descargarlo desde este mismo enlace[10] ,seleccionamos el sistema operativo corresponde a nuestra computadora y le damos a Download, seguimos unos pasos sencillos y listo.

.

[10] www.fritzing.org/download/

2. FASE-UNO ROBOT SEGUIDORE DE LÍNEA

En este apartado vamos a explicar en qué consiste un seguidor de línea y cómo funcionan los diferentes módulos que lo componen.

2.1. OBJETIVO GENERAL

EL Propósito de esta fase del proyecto es introducirnos al mundo de robótica construyendo un robot seguidor de línea, también más adelante usaremos un dispositivo Android para configurar los parámetros de control para mejorar la respuesta de nuestro robot.
Esta parte del proyecto que es la primera de tres partes más complejas, donde nuestro objetivo principal es explorar el potencial de los robots a base de Arduino y darles más funcionalidades.

2.2. OBJETIVOS ESPECÍFICOS

- ✓ Diseñar y construir un robot seguidor de línea con posibilidad de acoplarle otros módulos para facilitar el buen funcionamiento y darle más funcionalidad.
- ✓ Adaptar y colocar sensores necesarios para detectar líneas negras con fondo blanco.
- ✓ Analizar ventajas y desventajas sobre robots seguido de línea y adquirir más conocimientos sobre robótica.

2.3. FUNDAMENTOS Y ARQUITECTURA

Se sabe que todos los robots rastreadores pueden variar desde lo más básicos a más complejos, dependiendo de la complejidad del recorrido.
Se basan principalmente en los sensores los cuales los más básicos seguidores de línea suelen utilizar dos sensores. En nuestro caso utilizamos tres sensores para mejorar la velocidad de nuestro robot y corregir mejor su recorrido.
La estructura de nuestro hardware es como muestra en la figura siguiente.

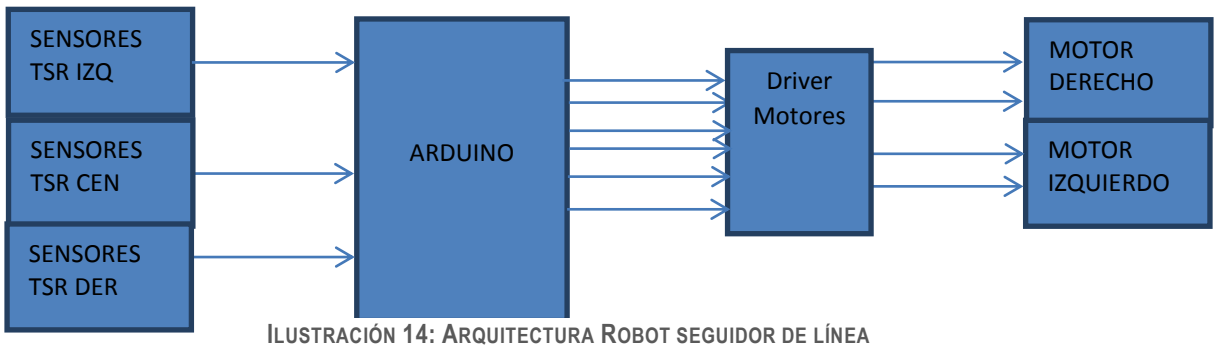


ILUSTRACIÓN 14: ARQUITECTURA ROBOT SEGUIDOR DE LÍNEA

2.4. MATERIALES

El kit que compre por internet es básico y no estaba completo, pero tiene los elementos imprescindibles para el desarrollo de nuestro Robot, así que estudiaremos todos y cada elemento con sus características y sus utilidades

2.4.1. Chasis

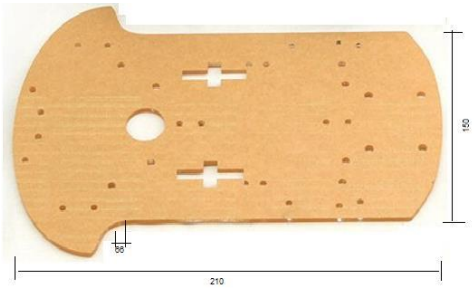


IMAGEN 30: DISEÑO BASE ROBOT

Como se ve en la "Imagen 30", el chasis de nuestro robot Compite en cuanto a aplicaciones con otros plásticos como el policarbonato (PC) o el poliestireno (PS), pero el acrílico se destaca frente a otros modelos en cuanto su resistencia a la intemperie, transparencia y resistencia al rayado.
La estructura mecánica es muy fácil de montar ya que viene recortada con unas medidas específicas de manera que todos los componentes a montar como driver para motores y diversos sensores quedan perfectamente colocados en el chasis.

A continuación, le colocamos la rueda loca y los soportes de motores como se ve en la "Imagen 31"

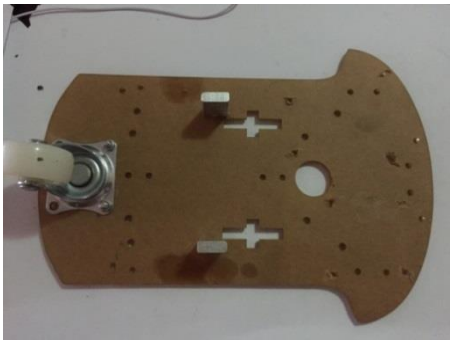


IMAGEN 31: BASE ROBOT CON RUEDA LOCA Y SOPORTE MOTORES

Así tendremos la base del robot lista a falta de montar motores, ruedas y las demás circuiterías.

2.4.2. Placa Arduino UNO

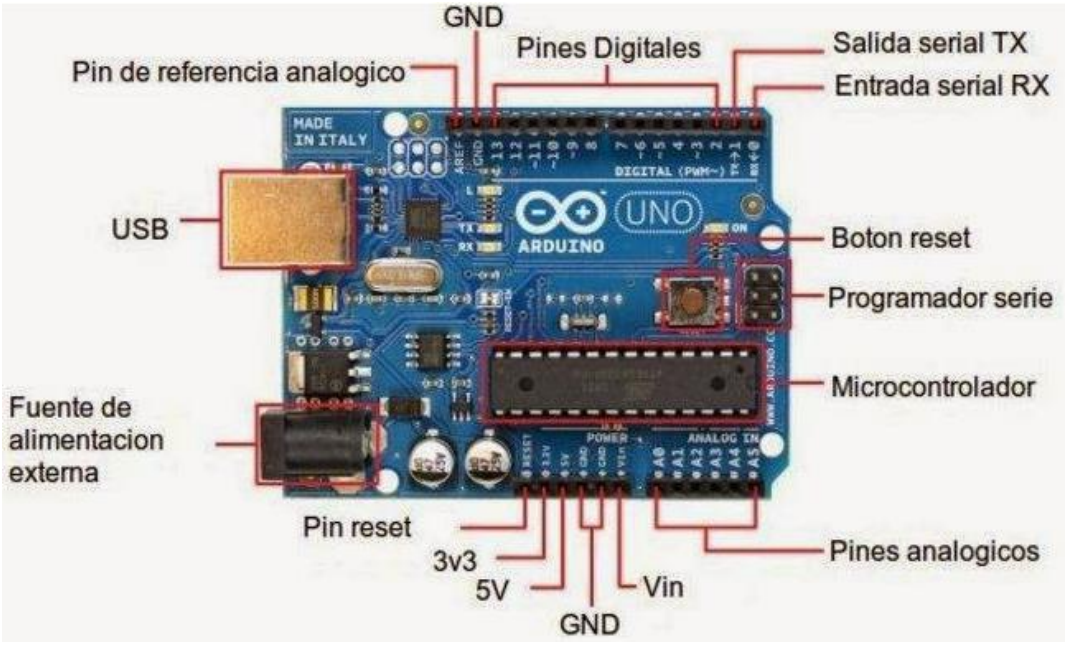


ILUSTRACIÓN 15: CARACTERÍSTICAS ARDUINO UNO

La placa Arduino Uno es de la más práctica de Arduino, actualmente existen dos tipos de placas a causa de la separación de los creadores de Arduino Italia que solo se diferencian por el nombre ,cuyos nombres :Arduino unica USA y Genuino en el resto de los paises.
Tambien existen muchas copias o clones que son de baja calidad que se pueden diferenciar facilmente como se ve en la figura abajo.

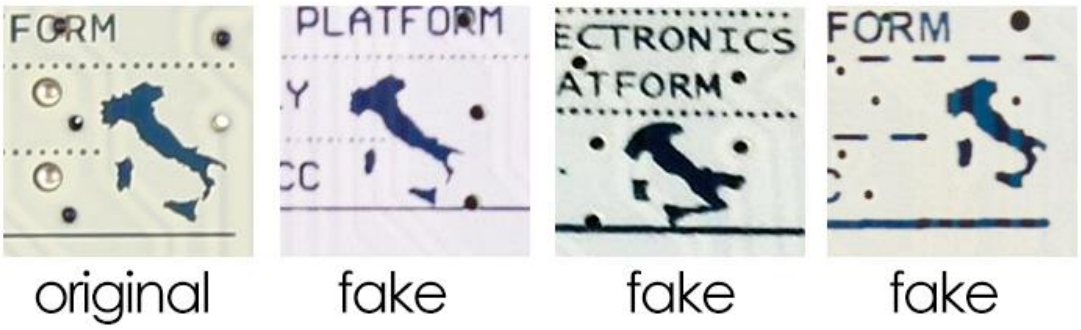


ILUSTRACIÓN 16: DIFERENCIA ENTRE ARDUINO ORIGINAL Y CLONE

La primera imagen a la izquiera es la Original como lo pone, el resto de placas son clones como su propio nombre indica en ingles **Fake**.

Como ya especifacamos las diferentes caracteristicas de la placa arduino en “la tabla 1” de comparacion,,a continuacion veremos en detalles las diferentes especifiaciones de la placa.

2.4.2.1. Alimentación

Hay tres formas de alimentar a la placa Arduino :

I. Alimentacion mediante puerto USB sin ordenador, usando un cargador de movil ya que suelen porcionar 5V DC ,lo suficiente para el consumo de la placa también entregan al menos una corriente DC de 1A , pero esta limitada a casi la mitad por la presencia de un fusible electrónico que corta la energia si supera los 500mA, cuya función es proteger el puerto USB del ordenador.

II. La segunda opción es la más adecuada para nuestro proyecto. Mediante el jack hembra. Hará falta un conector macho de 2.1mm de diamentro ,tiene dos conectores + y – como muestra la “imagene 32” a continuación.

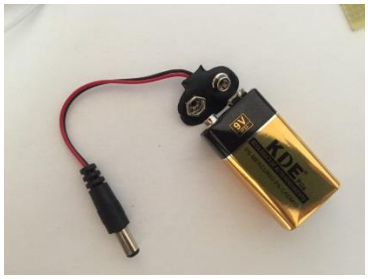


IMAGEN 32: ALIMENTACIÓN ARDUINO UNO

III. La tercera solución es alimentar nuestra placa mediante el pine **Vin** (tensión de entrada) y conectar la masa a cualquier **GND** de la placa, esta forma es menos segura ya que un pequeño fallo puede dañar seriamente nuestra placa.

2.4.2.2. Entradas y salidas digitales

La placa Arduino UNO dispone de 14 pines digitales de 0 a 13.Cada pin puede ser entrada o salida depende de las funcciones :

- pinMode();
- digitalWrite();
- digitalRead();

Cada salida o pine puede porporcionar una corriente de 40mA anque se recomienda el uso de la mitad 20mA, con un consumo total de 200mA que no debe ser superado por todos los pines .
Todos los pines digitales tienen una resistencia interna pull-up de 20 a 50 homnios que estan desactivas por defecto, pueden ser activadas, declarando la entrada de tal manera
pineMode(pin_nmr , IMPUT_PULLUP);

2.4.2.3. Entradas/salidas analogicas

La placa Arduino UNO dispone de 6 pines analogicos A0 a A5 ,miden un voltaje variable de 0V a 5V.
Estos pines estan conectados con convertidores analogico-digital ,es decir que el valor de 0 Voltios analogico equivale a B0000000000 y 5Voltios B1111111111 , en otras palabras todo valor analogico es expresado con un valor entre (0y 1023)→Resolucion de 10bits y suma 1 en Binario cada 4.883mVoltios.

2.4.2.4. Salidas PWM

Las salidas PWM de Arduino UNO son los pines que tienen el symbolo "~"
Pines 3, 5, 6, 9,10y11.
En nuestra placa Arduino la frecuencia del PWM esta predeterminada :

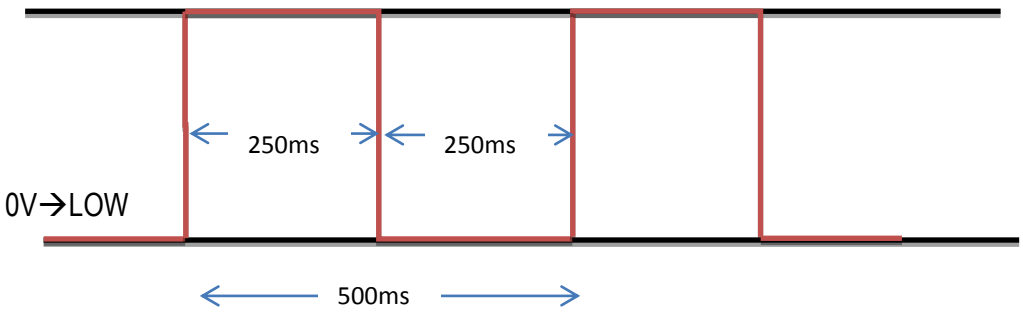
- ✓ pines 3, 9, 10 ,11= 490Hz
- ✓ pines 5,6= 980Hz
- ✓ se puede cambiarla pero no es algo de facil acceso.

2.4.2.5. Señal PWM

Salidas de pulso modulado es una señal que se va cambiando de nivel ,es decir alternativamente ALTO y BAJO en el mismo ciclo ,por ejemplo si conectamos un led se encendera y se apagara tan rapido que parecesa un led encendido permanente pero se notara menos brillo en led.

Ejemplo : un periodo de 1000ms asi que el led se iluminara durante 500ms y se apagara durante otros 500ms para poder visualizarle y tener la impresión de un brillo del 50%.
Tenemos $Frecuencia = \frac{1}{\text{Periodo}} \rightarrow F = \frac{1}{T} = \frac{1}{250x10^{-3}} = 4Hz$
Duty cycle varia de (0 a 100)%
0% → indica un señal baja todo el periodo.
50% →este caso significa que la mitad del periodo de la señal esta en estado Alto ,es decir utilizamos la mitad de potencia.
100% → señal alta en su totalidad de potencia todo el periodo.

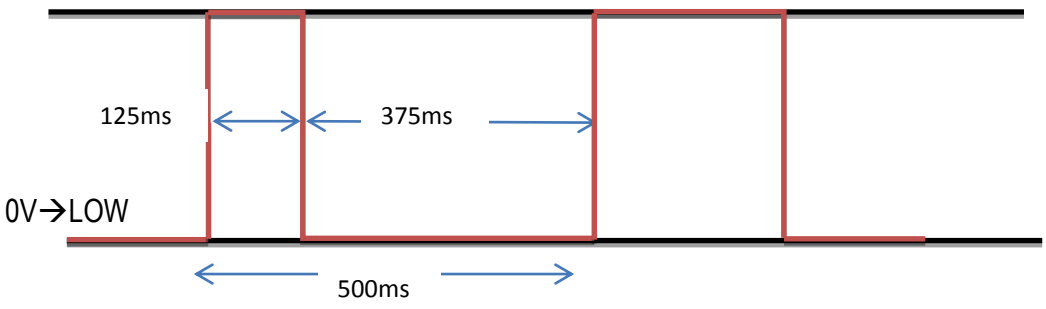
3.3V a 5V → HIGH



PWM 50%
Frecuencia =4Hz y el 50% dutty Cycle

Acontinuacion con un 25%duty el led se encendera 125ms y se apagara 375ms,asi que tendremos la impresion de un brillo del 25%.

3.3V a 5V → HIGH



PWM 25%
Frecuencia =4Hz y el 25% dutty Cycle

La señal PWM se utiliza tambien como tecnica para controlar circuitos analogicos ,ya que podemos determinar la tension suministrada a dicho circuito.

Ejemplo: un voltaje maximun de 5Voltios con 25%duty →se obtiene 1.25V.
80%duty →se obtiene 4V .

Como se puede ver ,con la tecnica de modulacion por ancho de pulsos o PWM podemos controlar el tiempo en que la señal esta en estado alto asi controlamos la potencia que le aplicamos dicha señal ,que a su ves suministra a motores, leds y diversos dispositivos electronicos.

2.4.2.6. Puerto Serie

los pines 0(Rx) y 1(Tx) : pines de recepcion y transmicion estos pines estan conectados a ATmega 16U2 que maneja el puerto serie USB asi que se recomienda desconectarlos cuando se realiza una descarga del programa.

Pin 13: este pin es especial ya que tiene conectado un led incorporado a la placa Arduino ,por lo cual se puede sacar aprocho.

2.4.2.7. Pines Power

GND :Se trata de la masa.
3V3 :Este pin ofrece 3V3 para una corriente máximo de 50 mA. Se puede utilizar para alimentar un dispositivo de este voltaje.
AREF :tensión de referencia para analógico-pin para su uso con la función analogReference ().
Reset :este pin srive para reiniciar la placa conectandolo a una masa o poniendolo a LOW . Es ampliamente utilizado en los escudos que permita un reinicio rapido.
IOREF: Este pin proporciona la tensión de referencia a la que funciona la tarjeta. En nuestro caso será 5V.

2.4.2.8. Esquema electrico Arduino

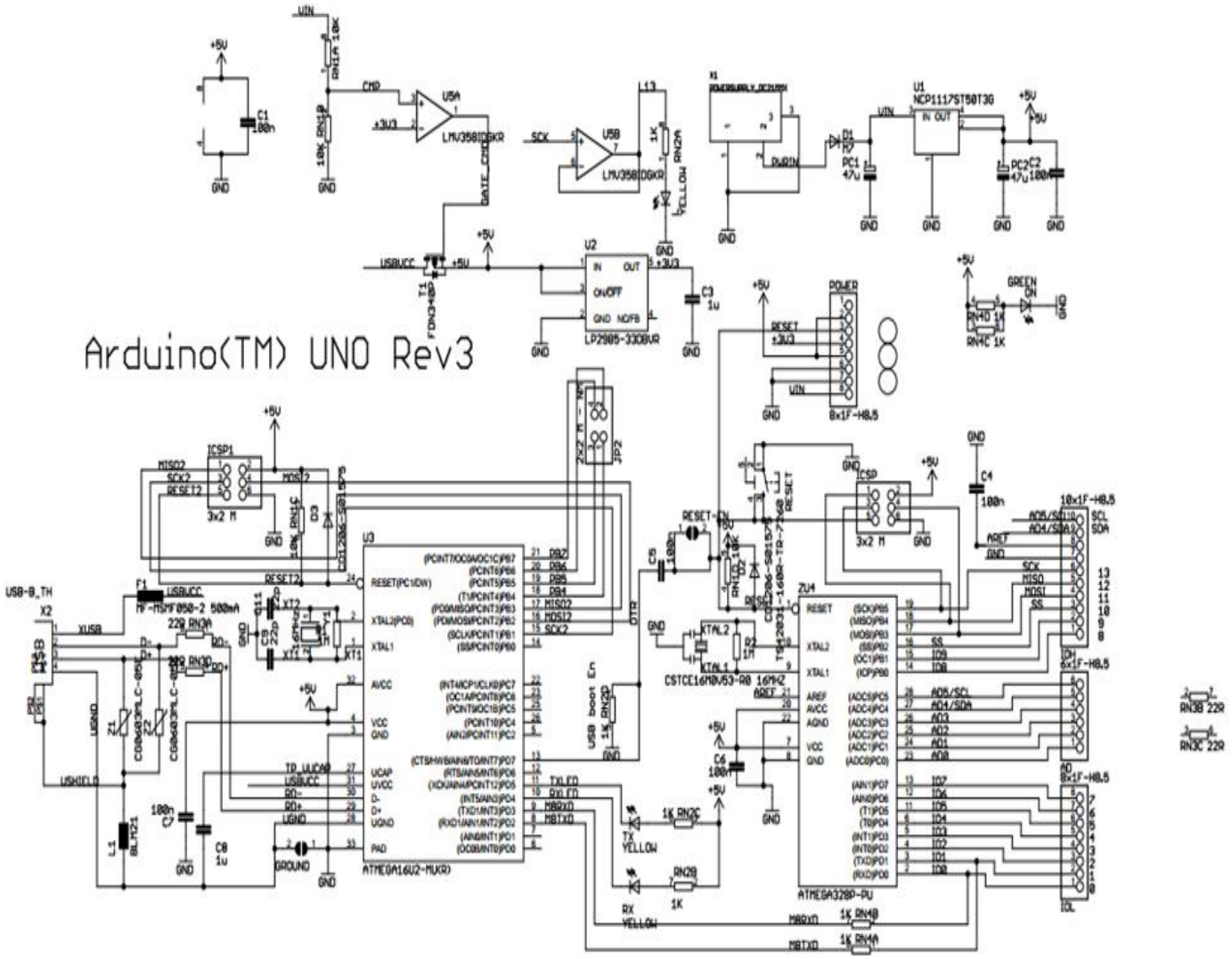


ILUSTRACIÓN 17: ESQUEMA ARDUINO UNO

2.4.2.9. Diagrama completo Pines placa Arduino UNO

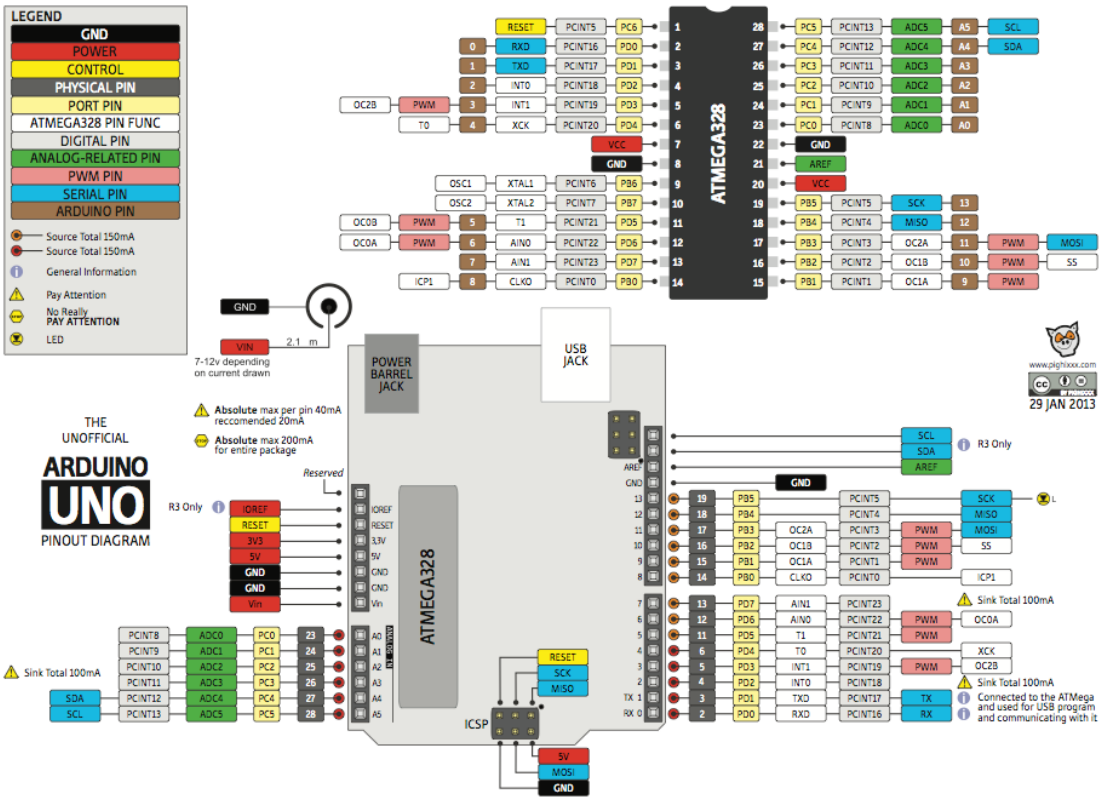


ILUSTRACIÓN 18: PLACA ARDUINO UNO

2.4.3. Motores DC

IMAGEN 32: MOTOR DC

Un motor de corriente continua o motor (DC), es un convertidor electromecánico para la conversión de energía bidireccional.
Los motores de corriente continua son más simples y se utilizan en electrodomésticos, ventanas automóviles, etc. Pero también se pueden utilizar como generadores de energía eléctrica que pueda convertir energía mecánica rotacional en energía eléctrica como el ejemplo de la dinamo en bicicletas.

2.4.3.1. Características motor

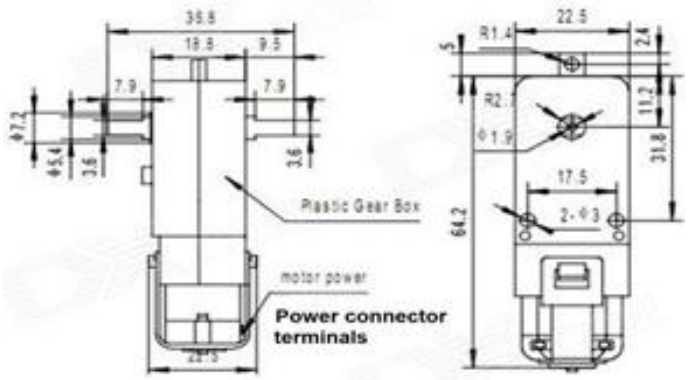


ILUSTRACIÓN 19: DISEÑO MOTOR

- ✓ Tamaño del motor: 70mm x 22mm x 18mm
- ✓ Peso Motor 50g.
- ✓ Tensión entre 3 V y 12 V (recomendado 6 a 8 voltios).

Los motores DC son capaces de convertir energía eléctrica en energía mecánica rotacional gracias a la acción de un campo magnético, la polaridad opuesta entre dos campos magnéticos dentro del motor hace que gire.

A continuación, veamos que se compone un motor de corriente continua.

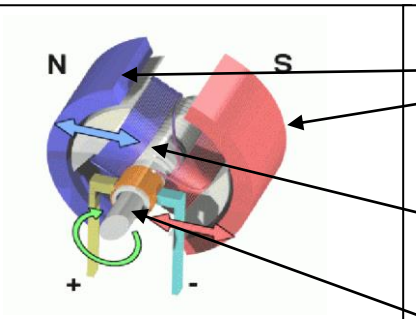


ILUSTRACIÓN 20: COMPONENTES MOTOR DC

Estator: parte no giratoria, contiene los polos del motor que pueden ser devanado de hilo de cobre sobre un núcleo de hierro o imanes permanentes como se ve en la figura.

Rotor: parte giratoria, forma cilíndrica generalmente y también llamado armadura que se somete a la acción inductora.

Colector: alterna la corriente entre una bobina y otra, es decir conecta la fuente de energía

2.4.3.2. Principio de funcionamiento

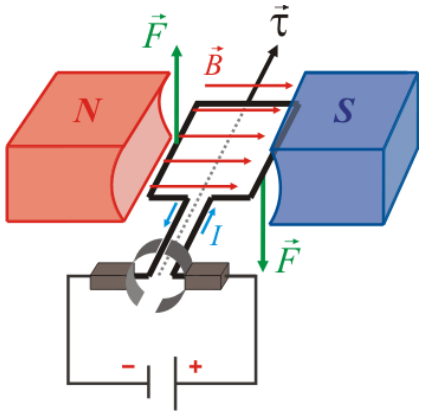


ILUSTRACIÓN 21: CONVERSIÓN ENERGÍA ELÉCTRICA EN MECÁNICA ROTACIONAL

En la "ilustración 21" se observa la forma de convertir la corriente eléctrica en mecánica rotacional, si inducimos corriente atreves de la armadura ubicada entre los polos norte y sur del imane se observa que el campo generado por la armadura interactúa con el campo del imane generando torsión está en un campo magnético este último que hará transformar la energía eléctrica en mecánica rotacional y la espira empezara a girar.

F=B x L x I

- F: fuerza sobre el conductor (newton)
- B: inducción de campo magnético (teslas)
- L: longitud del conductor(metros)
- I: intensidad que recorre al conductor(amperios)

2.4.3.4. Alimentación básica motor DC

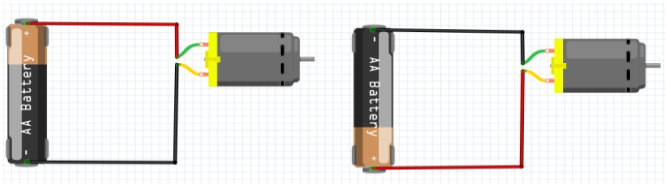


FIGURA 1: ALIMENTACIÓN MOTOR DC

En la figura anterior se observa la forma más sencilla de conectar un motor de corriente continua a una batería de 9V. Al conectar un amperímetro veremos que el motor se acerca a los 105 mA (miliamperios). Para un motor más grande vamos a necesita varios cientos de miliamperios.

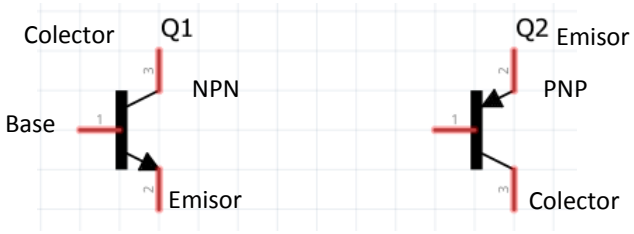
En este caso nuestro motor no consume más de 105mA en vacío, así que para una batería simple es demasiado corriente y no durara mucho, aparte que el motor siempre gira en su máximo velocidad.

La velocidad del motor no es ajustable además había que desconectar la batería para detener el motor y volver a conectarla para iniciarlo.

Resumiendo, esta configuración no es la mejor opción para darle movilidad a nuestro robot.

2.4.4. Transistor bipolar BC559B

El transistor bipolar BJT (bipolar junción transistor) es un transistor de tres terminales emisor, colector y base pueden ser de tipo PNP o NPN, se diferencia en el sentido de la corriente del emisor, si la flecha apunta hacia dentro del transistor es un PNP si la flecha apunta hacia fuera es NPN como se muestra la figura a continuación:



2.4.4.1. Características transistor BC559B

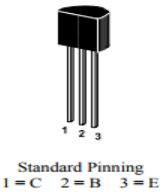


ILUSTRACIÓN 22: TRANSISTOR BC559B.

Especificaciones máximas: "Obtenida del Datasheet"

- IC: corriente colector 100mA
- Tensión colector-base 30V
- Tensión colector-emisor 25V
- Tensión emisor-base 5V
- Temperatura máxima 150·C
- Potencia máxima 500mW

2.4.5. Resistencias

La resistencia eléctrica es un componente electrónico con dos pines, se utilizan para proteger otros componentes y disminuir la intensidad de la corriente eléctrica, cuanta más alta la resistencia más disminuye la corriente que le atraviesa, su valor suele ser marcado con la letra R.

Símbolo:

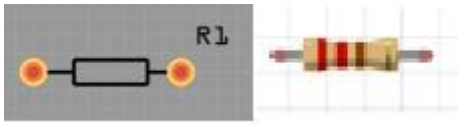
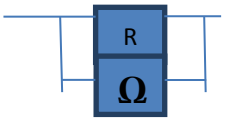


IMAGEN 34: SÍMBOLO RESISTENCIA

2.4.5.1. Valor de resistencia

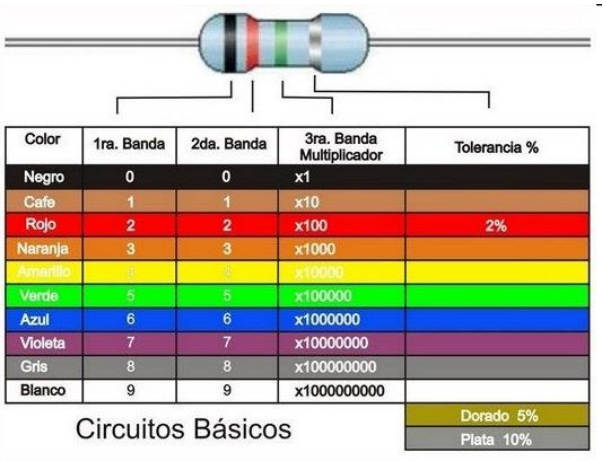
a) Mediante un ohmímetro en paralelo a la resistencia



b) Ley de ohm V(voltios)=R(ohmios) x I(Amperios)

R=V/I (Ω)

c) Código colores



Color	1ra. Banda	2da. Banda	3ra. Banda Multiplicador	Tolerancia %
Negro	0	0	x1	
Cafe	1	1	x10	
Rojo	2	2	x100	2%
Naranja	3	3	x1000	
Amarillo	4	4	x10000	
Verde	5	5	x100000	
Azul	6	6	x1000000	
Violeta	7	7	x10000000	
Gris	8	8	x100000000	
Blanco	9	9	x1000000000	
				Dorado 5%
				Plata 10%

Circuitos Básicos

ILUSTRACIÓN 23: CÓDIGO COLORES RESISTENCIAS[11]

2.4.6. Diodo Led

Light Emitting Diode o Diodo Emisor de luz, son componentes electrónicos que permiten el paso de la corriente en un solo sentido, una vez pasa la corriente atreves del led se dice que esta polarizado directamente así emite luz.

[11] http://www.areatecnologia.com/electricidad/resistencia-electrica.html

2.4.6.1. Características diodos led

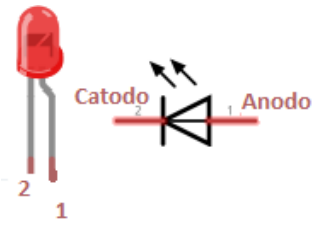


IMAGEN 35: SÍMBOLO DIODO LED

Los diodos leds tienen dos patas de conexión, una más larga que la otra como se ve en la "imagen35" anterior, para que emita luz se debe conectar la pata larga al positivo (pata numero 1) y la pata corta (pata nmr2) al negativo, en caso contrario la corriente no pasara por el cátodo del diodo y no emitirá luz.

Los leds suelen trabajar a tensiones de 2V a 3.3V si queremos conectarlos a tensiones más altas es recomendable que sea a través de una resistencia en serie para que disipe parte de la tensión.

2.4.7. Puente H (H-Bridge)

Un puente en H, es un circuito electrónico que permite a un motor de corriente continua girar en ambos sentidos, avanzar y retroceder también pararlo rápidamente.

Para darle movilidad a nuestro le robot y poder moverlo libremente adelante, Atrás o pararlo necesitaremos una etapa de potencia que debe ser compatible con niveles TTLS (0 o 5 V) ya que utilizamos circuitos lógicos digitales.

2.4.7.1. Funciones requeridas

Movimiento	Voltaje terminal A-Motor	Voltaje terminal B-Motor
Giro a la derecha	+5V	0V
Giro a la izquierda	0V	+5V
Motor parado	0V	0
	+5V	+5V

TABLA 2: FUNCIONES PUENTE H

En la "tabla 2" se ven los tres movimientos requeridos con cuatros posiciones posibles.

Su nombre "H" viene dado de la posición de los transistores que veamos continuación también llamados interruptores de estado sólido en forma de H.

2.4.7.2. Funcionamiento puente en H con interruptores

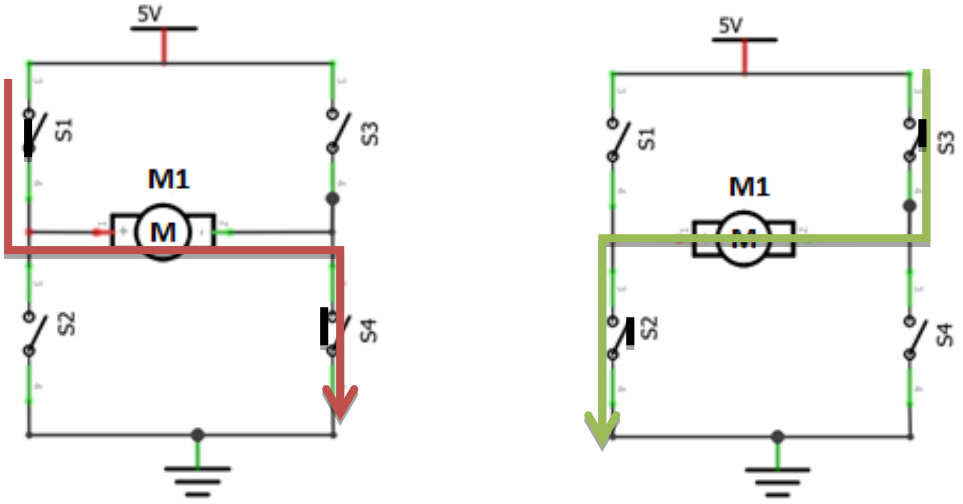


FIGURA 2: GIRO MOTOR EN AMBOS SENTIDOS A TRAVÉS DE INTERRUPTORES

Como se ve en la "figura 2" para que el motor gire en un sentido se tiene que cerrar los interruptores S1 y S4 y para que gire en el otro sentido se deben de cerrar los interruptores S3 y S2 y para pararlo se deben de abrir S1 y S3 o S2 y S4.

Nota: jamás debe de cerrarse los interruptores S1 y S2 ni S3 y S4 ya que causaría un cortocircuito en la fuente de alimentación.

2.4.7.3. Funcionamiento puente en H con transistores

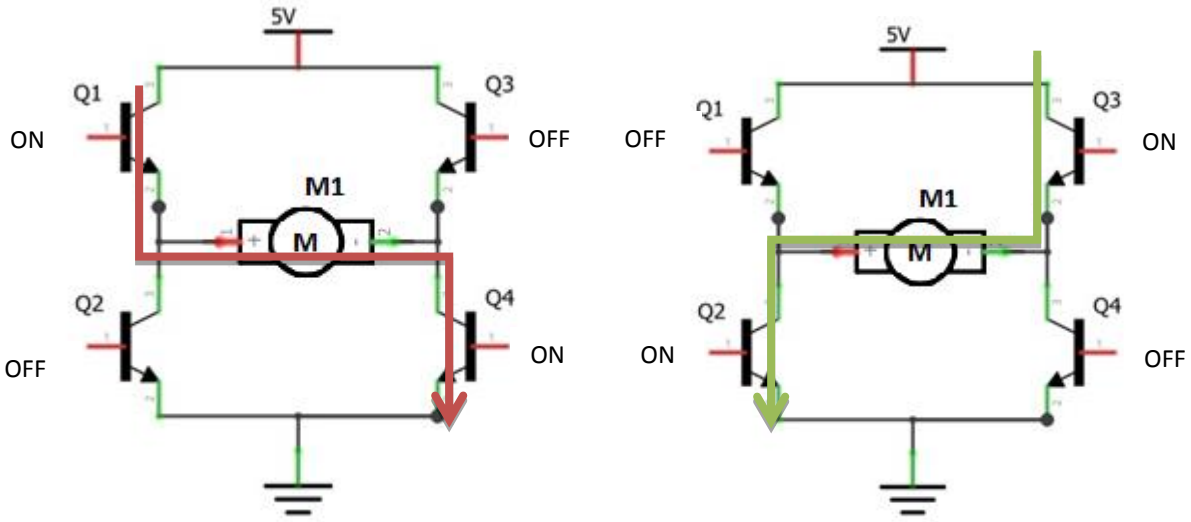


FIGURA 3: GIRO MOTOR EN AMBOS SENTIDOS ATREVES TRANSISTORES

Los transistores Q1, Q2, Q3 y Q4 tienen la misma función que los interruptores S1, S2, S3 y S4 consecutivamente, solo trabajan en dos modos:

- ***Corte:*** un transistor en corte es equivalente a un interruptor abierto, no deja corriente por sus terminales.
- ***Saturación***: un transistor saturado es igual que un interruptor cerrado deja pasar Toda la corriente que circula.

Como se ve en los dos circuitos de la "Figura 3", un puente H se construye con 4 transistores o 4 interruptores, cada interruptor puede estar en dos estados abierto o cerrado, lo mismo pasa con los transistores estarán ON u OFF.
Cuando los transistores Q1yQ4 están ON→ Q2yQ3 estarán OFF se aplica una tensión positiva al motor y el motor gira en sentido horario como muestra la "figura 3" anterior a la izquierda.
Cuando los transistores Q3yQ2 están ON→Q1yQ4 estarán OFF se aplica una tensión negativa al motor lo que hace invertir su giro en sentido antihorario.
Con esta técnica logramos controlar el sentido del motor en ambas direcciones.
La ventaja de utilizar transistores en vez de interruptores es que tienen más tiempo de vida y altas frecuencias de conmutación que los interruptores, además lo podemos controlar mediante señales lógicas y circuitos de potencia.
Así que usar interruptores mecánicos en circuitos de potencia es una cosa impensable.

2.4.8. Integrado L293D

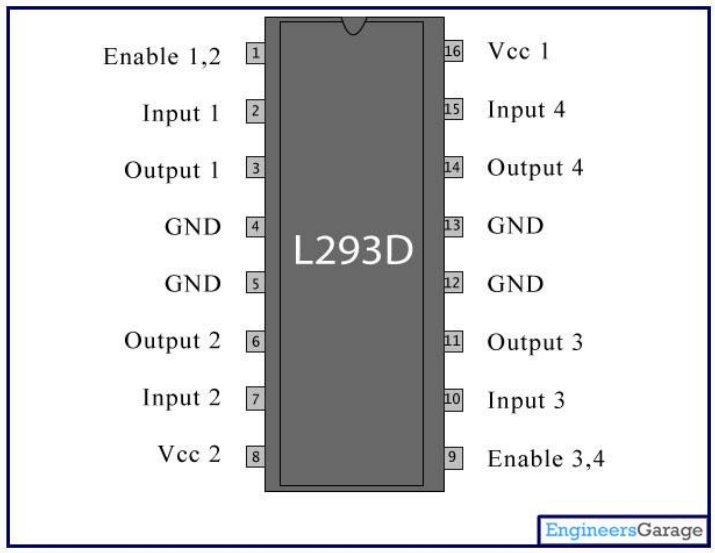


ILUSTRACIÓN 24: INTEGRADO L298D[12]

2.4.8.1. Definición

Es un integrado construido para controlar motores de corriente continua, el L293D es un sistema que facilita el control del sentido de giro de un motor DC usando dos puentes en H como se ve en la "figura 4" a continuación

[12] https://www.engineersgarage.com/electronic-components/l293d-motor-driver-ic

2.4.8.2. Tabla de pines integrado L293D

Pin	Nombre	Función
1	Enable 1, 2	Pin para habilitar motor 1 HIGH→ activo
2	Entrada 1	Entrada 1→ Motor 1
3	Salida 1	Salida 1 →Motor 1
4	GND	Tierra 0V
5	GND	Tierra 0V
6	Salida2	Salida 2 → Motor 1
7	Entrada2	Entrada 2→Motor 1
8	Vcc 2	Tensión de alimentación Motores 9 a 12V
9	Enable 3, 4	Pin para habilitar Motor 2 HIGH →Activo
10	Entrada 3	Entrada 1 →Motor 2
11	Salida 3	Salida 1 →Motor 2
12	GND	Tierra 0V
13	GND	Tierra 0V
14	Salida 4	Salida 2 →Motor 2
15	Entrada 4	Entrada 2→Motor 2
16	Vcc1	Tensión de alimentación 5V (suporta hasta 36V)

TABLA 3: FUNCIONES, PINES INTEGRADO L293D

2.4.8.3. Aplicación L293D

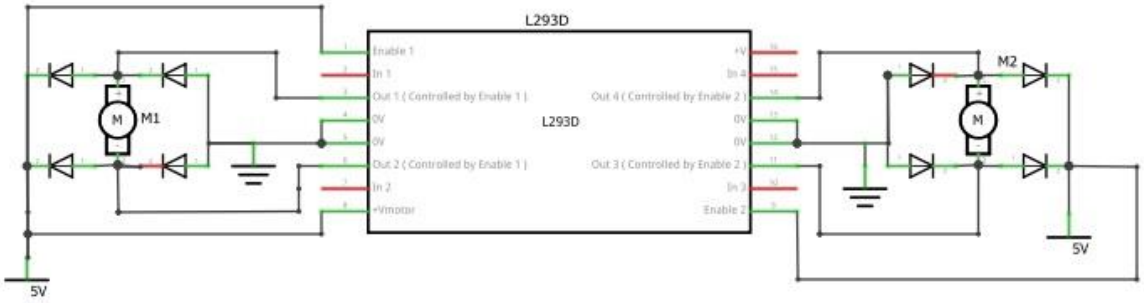


FIGURA 4: CIRCUITO CONTROL MOTORES L293D

Como se ve en la "figura 4", este integrado nos permite controlar dos motores en simultáneo.
Los pines 3 y 6 conectados a los terminales del motor1 y los pines 11 y 14 conectado a los pines del motor2 como se ilustra en la figura anterior, sin olvidar conectar los pines 1 y 9 a Vcc para habilitar los dos motores
El pin 16 es para alimentar el mismo integrado que debería estar conectado a la tensión que alimentara los motores que puede varia de 5V a 36V.
Así que es importante saber que esté integrado se alimenta con dos niveles de tensión diferentes, la primera corresponde a la propia alimentación del integrado Vss<7V y la segunda con la que alimentamos los motores VC<36V.

2.4.9. Módulo L298N

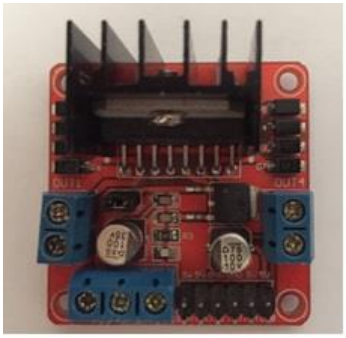


IMAGEN 36: DRIVER DE POTENCIA L298N

Este módulo de potencia basado en el chip L298N que es el hermano mayor del L293D dispone de dos canales de puente H que nos permiten controlar dos motores DC o un motor paso a paso de hasta 2 Amperios, mucho más de lo que ofrece su hermano pequeño L293 que a corrientes altas, más de 600mA se calentara mucho.

Además, este módulo cuenta con disipador de calor además de todos los componentes necesarios para el control de los motores sin necesidad de ningún elemento adicional.

2.4.9.1. Esquema electrónico

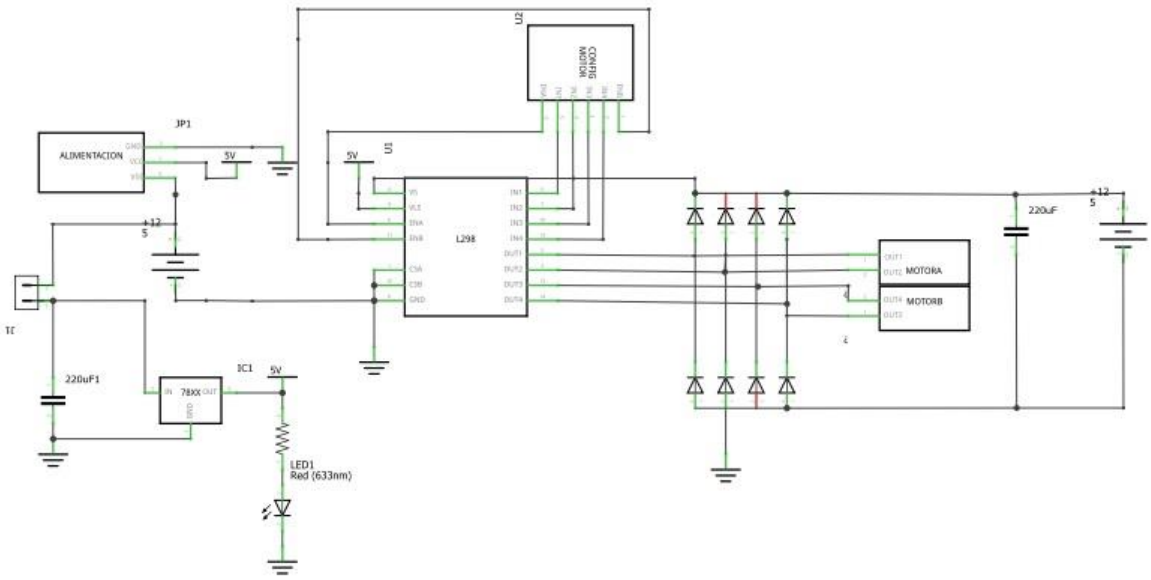


IMAGEN 37: CIRCUITO INTERNO L298N

Como se ve en la figura, el módulo está compuesto por un driver L298N, diodos de protección que evitan pasar corriente y un regulador de voltaje LM78M05 que nos proporciona una salida estable de 5V que alimenta a su vez el integrado l298N.

Un conector de pines que le puse el nombre (CONFIG MOTOR) para ingresar las señales TTL para controlar los motores y habilitar a cada salida del módulo.

- ✓ Conector1 La salida A→OUT1 Y OUT2,
- ✓ Conector 2 La salida B→OUT3 Y OUT4

Finalmente, el jumper con los pines de habilitación ENA Y ENB para habilitar las salidas A y B.

Al final optamos por el módulo comercial porque sale más barato que comprar componentes y placa además de imprimirla y soldar todos los componentes.

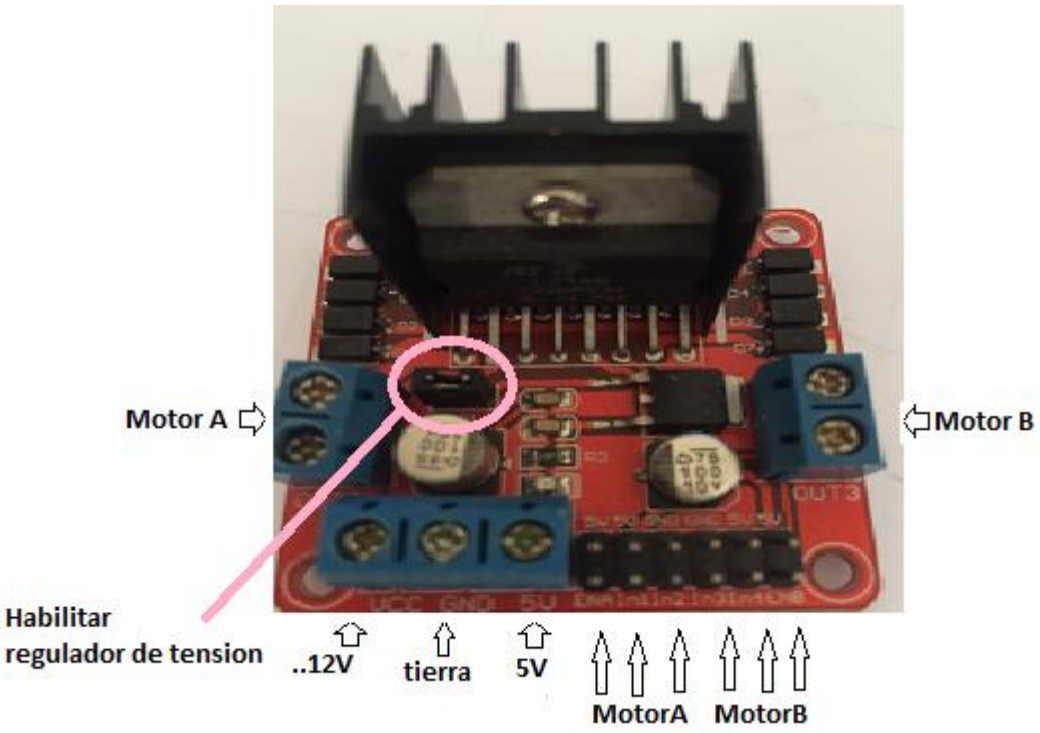


ILUSTRACIÓN 25: PINES L298N

Para alimentar nuestro modulo, solo utilizamos una sola fuente de alimentación de 6 a 12 V, que es la misma tensión que se suministrara a los motores, pero el regulador de tensión nos proporciona además una tensión de 5V estable y una corriente máxima de 500mA lo que nos proporciona una fuente de alimentación adicional para alimentar sensores u otros dispositivos.

En el caso de utilizar dos fuentes, por ejemplo: una batería de 12V para que trabajen los motores y otra fuente de 5V conectada a la entrada de 5V, se debe de desconectar el jumper lo que deshabilitara el regulador de tensión LM7805.

2.4.10. Sensores TCRT500

IMAGEN 38: SENSOR TCRT500

El sensor TCRT500 es un sensor óptico refractivo infrarrojo que está construido en una base compacta y posee un led emisor y un fototransistor como receptor, ideal para detectar de obstáculos, proximidad, color, etc.

Se puede conectar directamente al microcontrolador de la placa Arduino e ideal para nuestro seguidor de línea.

2.4.10.1. Características TCRT500

- ✓ Alimentación y voltaje de trabajo: 5V (DC)
- ✓ Tamaño: 10mm x 40mm
- ✓ Distancia de detección: Desde 1mm a 8mm, ajustable mediante un potenciómetro de calibración de sensibilidad y un led que detecta su estado.
- ✓ Tiene salida digita y un filtro que bloquea la luz del día

Pines:

- VCC (+5V)
- OUT (Salida)
- GND (0V)

Si detecta un objeto color negro → Vout > 4.5V (DC).
Si no detecta nada línea blanca → Vout = 0V.

2.4.10.2. Funcionamiento TCRT500

El módulo TCRT5000 es un sensor compuesto por un diodo led emisor de luz infrarroja en constante funcionamiento, y un fototransistor que tiene un filtro de luz natural que permite el paso de la luz emitida por el led.
Por lo tanto, dependiendo de la efectividad de la superficie el fototransistor recibe un valor mayor o menor de la señal reflejada.

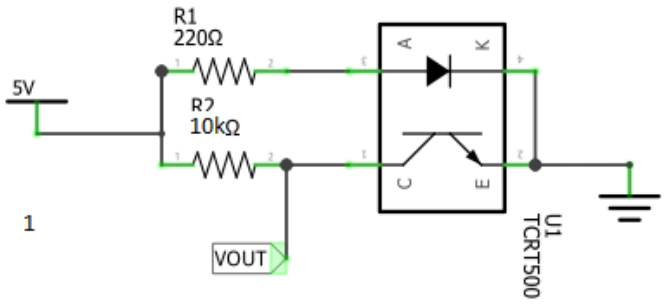


FIGURA 5 : CIRCUITO DE APLICACIÓN TCRT500

Una resistencia de 220Ω entre la alimentación +5V y el diodo led emisor de infrarrojo.
Una resistencia de 10KΩ en serie con fototransistor receptor de infrarrojo.

2.4.11. Soporte sensores TCRT500

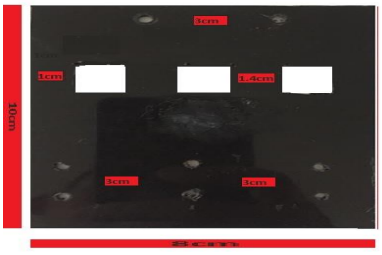


IMAGEN 39: SOPORTE TCRT500

Una placa de plástico recortada para poder conectarla al chasis del robot y al mismo tiempo sirve como soporte para los tres sensores TCRT5000.

2.4.12. Tornillos

IMAGEN 40: TORNILLOS

Como se ve en la "Imagen 4", vamos a utilizar tornillos 3M de 3mm de diámetro.
4 tornillos de 3x30mm para los motores y el resto de tornillos son de 3x10mm y 3x5mm.

2.4.13. Separadores

IMAGEN 41: SEPARADORES

Separador metálico hexagonal M3 macho se venden en bosas de 25unidades, vamos a utilizarlos para fijar los módulos TCRT5000 al soporte de sensores que a su vez montado en el chasis del robot.

2.4.14. Protoboard-Mini

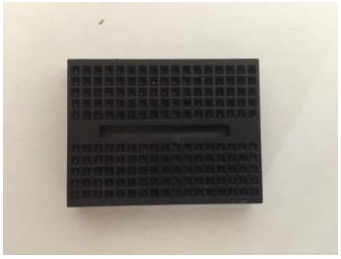


IMAGEN 42: PROTOBOARD-MINI

Tablero con orificios se encuentran conectados eléctricamente entre sí, normalmente siguiendo patrones de líneas, donde se pueden insertar cables y componentes electrónicos para el armado y prototipo de circuitos electrónicos.
Está construido por dos materiales:

i. Aislante, suele ser generalmente un plástico.
ii. Conductor, conecta los orificios entre sí.

Su principal utilidad es la creación y comprobación de circuitos electrónicos antes de llegar a la impresión mecánica del circuito en sistemas de producción comercial.
Ideal para construir pequeños circuitos electrónicos adicionales de nuestro proyecto cuya dimensión: 4,4 cm x 3,5 cm x 0,9 cm y no pesa casi nada.

2.4.15. Cable Macho-Macho

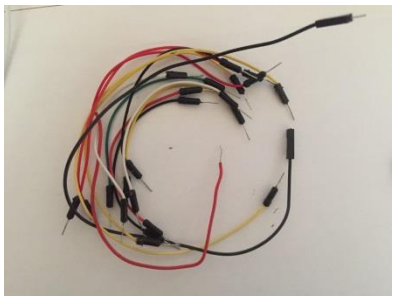


IMAGEN 43: CABLE MACHO-MACHO

Cables para circuitos adicionales, juntar masas y alimentación.

2.4.16. Cable Hembra-Macho

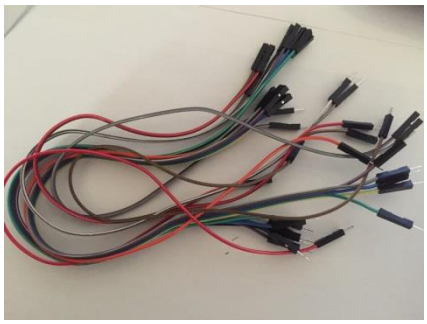


IMAGEN 44: CABLE MACHO-HEMBRA

Cables para conector los sensores a la placa Arduino.

2.4.17. Conector Jack-Macho + Batería 9V

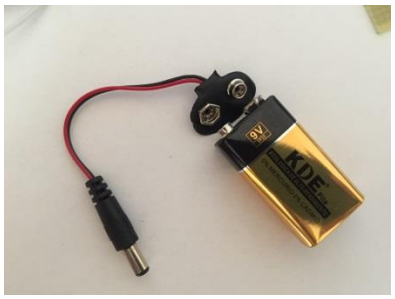


IMAGEN 45: BATERÍA 9V +CONECTOR JACK

Conector Jack- Macho y batería para alimentar la placa Arduino mediante el Jack-Hembra.

2.4.18. Soporte pilas

IMAGEN 46: PORTA PILAS

Una porta pilas de cuatro pilas de 1.5V los que nos da una tensión total de 6V suficiente para alimentar nuestro módulo de potencia L298N que a su vez alimenta otros sensores.

2.4.19. Cuatro pilas 1.5V AA

IMAGEN 47: PILAS 1.5V A

Baterías o fuentes de energía más populares del mundo.

2.5. CONFIGURACIÓN DEL HARDWARE

Tras analizar las diferentes placas de la plataforma Arduino, nuestra elección ha sido la placa Arduino UNO ya que cumplí nuestros objetivos esenciales, pines suficientes comunicación TX, RX y se puede alimentarse fácilmente con una batería externa.
A continuación, vamos a proceder a comunicar nuestra placa Arduino UNO con ordenador atreves el puerto USB, para comprobar su buen funcionamiento antes de proceder con las fases del proyecto.
.

2.5.1. Comprobar placa Arduino UNO.

Mediante el programa Fritzing hacemos un pequeño circuito con dos leds los cual podemos apagar y encender con el fin de comprobar la comunicación puerto serie y el funcionamiento de la placa, si no disponemos de leds podemos utilizar el mini-led de la placa Arduino UNO que va conectado al pin 13 con el mismo fin.

2.5.2. Materiales de la prueba

- ✓ Placa Arduino
- ✓ Protoboard
- ✓ 1 Resistencia de 330khom
- ✓ Led interno placa Arduino
- ✓ Led

2.5.3. Esquema Arduino Led

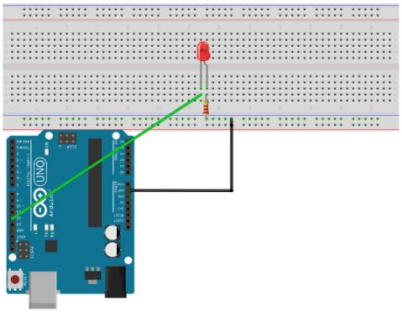


FIGURA 6: COMPROBACIÓN TARJETA ARDUINO

2.6. Configuración del software

Primero le damos el IDE Arduino para abrir un nuevo sketch

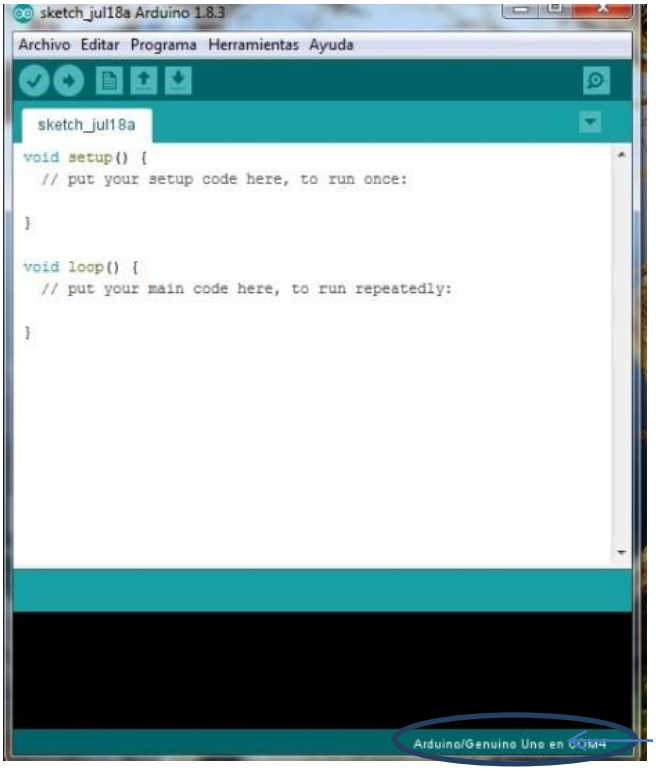


ILUSTRACIÓN 26: INTERFACE IDE ARDUINO

Una vez abierto un nuevo sketch, en el menú desplegable:

Herramientas → se selecciona la tarjeta **Arduino Genuino UNO**
Herramientas → puerto de comunicación se suele estar escrito abajo del Sketch como se ve en la "Ilustración 26" en este caso el **COM4**.

2.6.1. Tabla código resumido

Estructura	Variables	Funciones
• setup() →inicialización • loop() →bucle **Estructuras de control** • if →comparador si-entonces • if...else →comparador si...sino • for →bucle con contador • switch case →comparador múltiple • while →bucle por comparación booleana) • do... while →boucle por comparación booleana • break →salida de bloque de código • continue →continuación en bloque de código • return →devuelve valor a programa **Sintaxis** • ; • {} →llaves • // →comentarios en una línea • /* */ →comentarios en múltiples líneas **Operadores Aritméticos** • = →asignación) • + →suma →resta • →multiplicación • / →división • % →resto **Operadores Comparativos** • == →igual a • != →distinto de • < →menor que • > →mayor que • <= →menor o igual que • >= →mayor o igual que **Operadores Booleanos** • && →(y) • || →(o) • ! →(negación) **Operadores de Composición** • ++ →incrementa • -- →decrementa • += →composición suma • -= →composición resta • *= →composición multiplicación • /= →composición división	**Constantes** • HIGH | LOW • INPUT | OUTPUT • true | false • Constantes Numéricas **Tipos de Datos** • boolean →booleano • char →carácter • byte • int →entero • unsigned int →entero sin signo • long →entero 32b • unsigned long →entero 32b sin signo • float →en coma flotante • double en coma flotante de 32b • string →cadena de caracteres • array →cadena • void →vacío **Conversión** • char() • byte() • int() • long() • float()	**E/S Digitales** • pinMode() • digitalWrite() • digitalRead() **E/S Analógicas** • analogRead() • analogWrite() - **PWM (modulación por ancho de pulso** **Tiempo** • millis() • micros() • delay() • delayMicroseconds() **Matemáticas** • min() →mínimo • max() →máximo • abs() →valor absoluto • constrain() limita • map() →cambia valor de rango • pow() →eleva a un número • sq() →eleva al cuadrado • sqrt() →raíz cuadrada **Trigonometría** • sin() →seno • cos() →coseno • tan() →tangente **Números Aleatorios** • randomSeed() • random() **Comunicación** • Serial

2.6.2. Estructura código Arduino

- Declaración de las variables: tipo constante, una variable que no cambia durante la ejecución del programa, su comando es **const**, además se debe especificar qué tipo de dato es la variable.
- Definir si la variable es entrada o de salida, para eso se utiliza el comando **void setup (),** abrimos un corchete **({)** definimos se las variables si son entradas o salidas mediante el comando **pinMode(led, OUTPUT)** en este caso declaramos el pin del led como salida y al finalizar el comando se cierra corchete (**}**).
- Una vez definidas las variables, se procede a realizar la estructura del código a través del comando **voidloop ()**, de igual manera se abre corchete ({) y se cierra luego de terminar su cumplimiento (}).
- En el **voidloop ()** se establece las instrucciones que ejecutará Arduino una vez compilado.
- Una vez terminamos de desarrollar el programa, se debe compilar
  Programa → Verificar, se verificar si existen errores dentro del código. En caso contrario se puede cargar el código en la tarjeta Arduino para que esta lo ejecute.

2.6.3. Codigo1_ensayo placa Arduino

```
// Testear placa Arduino con un led
const int Led = 13 ; // Pin 13 conectado al led interno de Arduino
const int Led1 = 11;//Pin 11 Conectado al led1 Verde
void setup()
{
    pinMode(Led,  OUTPUT) ;// declaramos pin3 como salida
    pinMode(Led1,  OUTPUT) ;// declaramos pin3 como salida
}

void loop()
  {
      digitalWrite(Led, HIGH);// pin13→nivel Alto Led Amarillo Arduino ON
     digitalWrite(Led1, LOW);// pin11→nivel Bajo Led Verde OFF
      delay(2000);// 2s de espera led ON
      digitalWrite (Led, LOW);// pin13→nivel Bajo Led Amarillo OFF
     digitalWrite(Led1, HIGH);// pin11→nivel Alto Led Verde OFF
      delay(1000);// 1s de espera
  }
```

2.6.4. Complicar y cargar código IDE Arduino

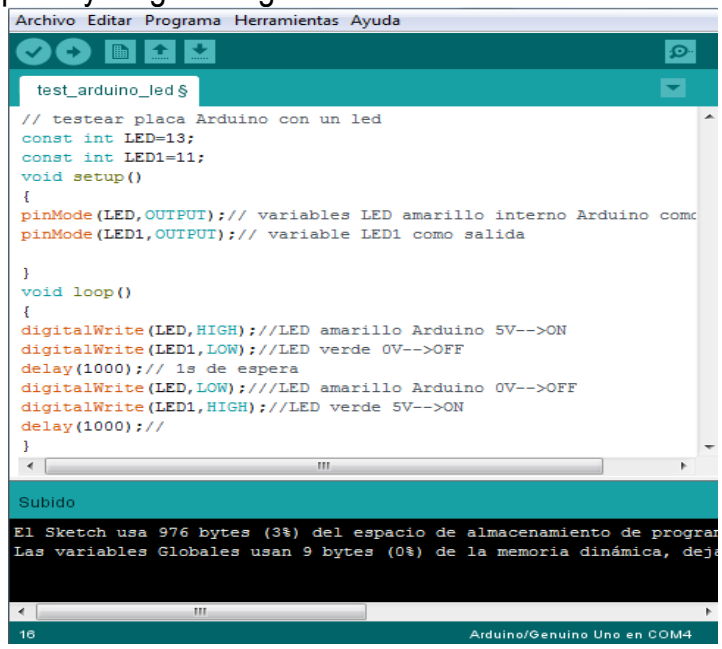


ILUSTRACIÓN 27: CÓDIGO LED ARDUINO UNO

Una vez tenemos el código se puede complicar y cargar de dos formas:

Una vez introducido el código en Sketch del IDE Arduino, le damos a compilar  y después si no tiene ningún error lo cargamos  .

2.6.5. Compilar y carga código con Fritzing

La otra opción es mediante el programa Fritsing como se demuestra a continuación:

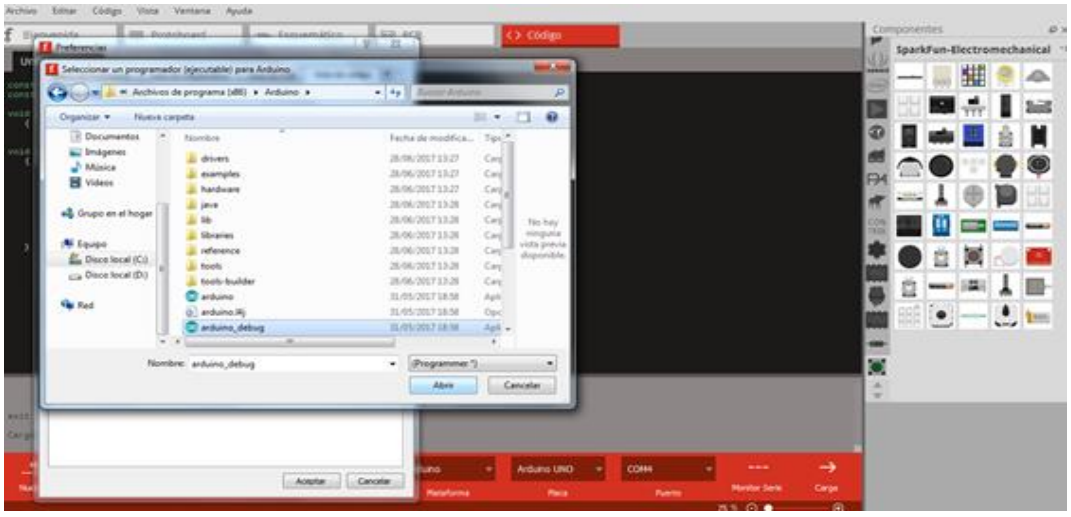


ILUSTRACIÓN 28: COMPILAR CÓDIGO ARDUINO EN FRITZING

Una vez tenemos nuestro circuito implementado en el Fritsing procederemos a meter el código, seleccionamos la tarjeta Arduino UNO y el puerto serial COM4.

Después iremos al menú del programa **editar→preferencias**, le damos a las flechas hasta llegar a la **vista del código→ubicación** abrimos carpeta Arduino y cargamos **arduino-debug.exe** como se muestra en la "ilustración 28".

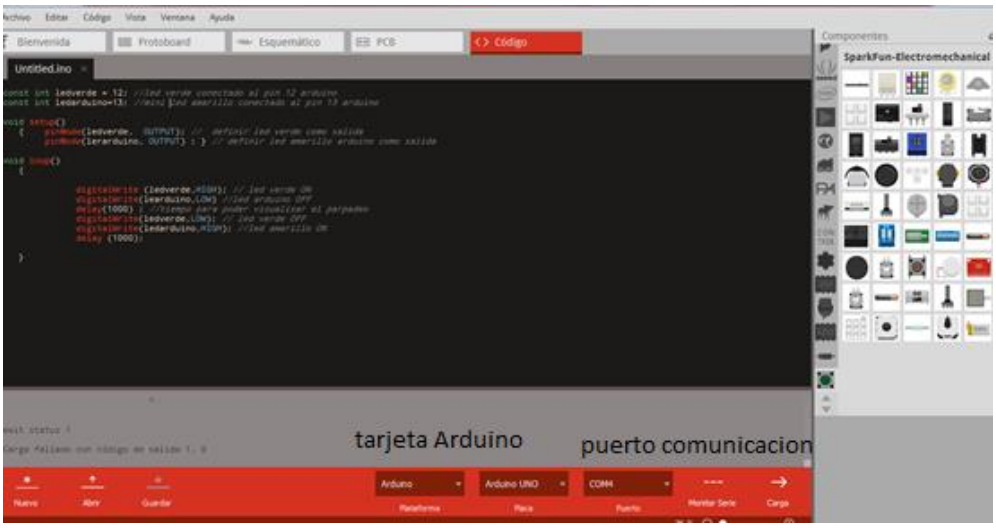


ILUSTRACIÓN 29 : CARGAR COMPILADOR CÓDIGO FRITZING

Una vez cargado arduino-debug.exe podemos compilar y cargar el código del programa Fritzing.

2.6.6. simulación prueba

A continuación, visualizamos el resultado de nuestros leds parpadeando cada 1s.

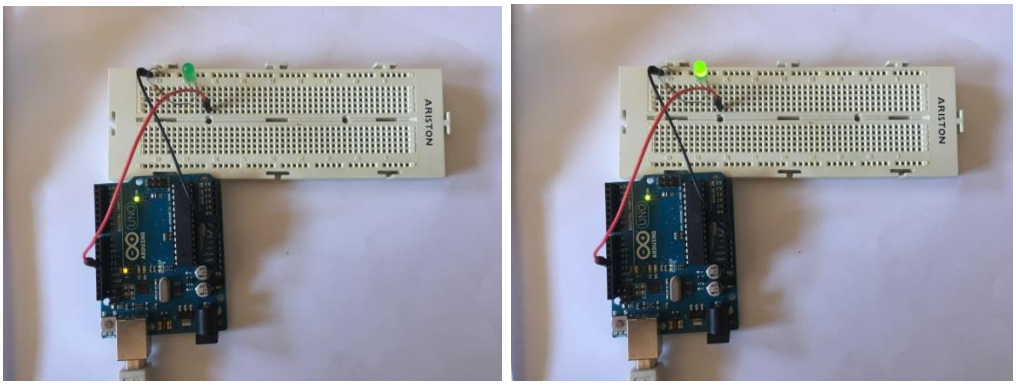


IMAGEN 48: SIMULACIÓN PRUEBA ARDUINO LED

En la primera Imagen veemos como el led interno de la tarjeta Arduino conectado al pin 13 esta encendido y el led1 conectado al pin 11 apagado.

Despues de 1 segundo se apaga el led y se enciende el led1 y asi sucesivamente.

Una vez comprobada la tarjeta Arduino procedemos con nuestra primera fase del proyecto, que consistes en diseñar una Robot seguidor de línea cuya base aprovecharemos para darle más funcionalidad a nuestro Robot en las siguientes fases del proyecto.

2.7. CONFIGURACIÓN ARDUINO CON MOTORES

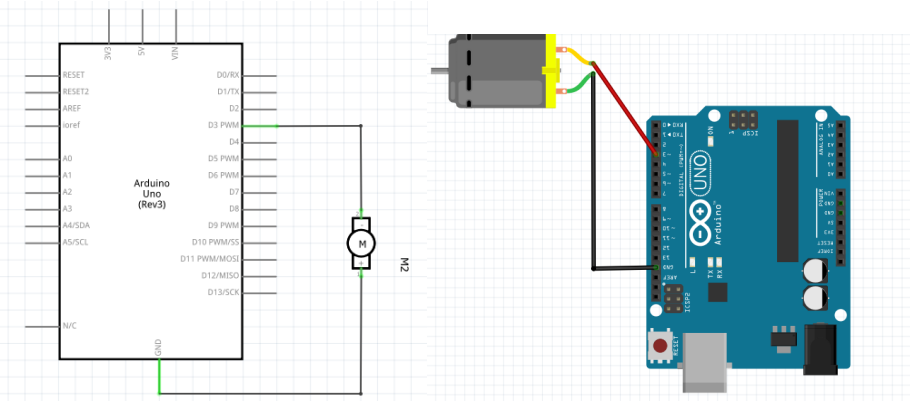


FIGURA 7: CIRCUITO CONEXIÓN ARDUINO-MOTOR DC, FRITZING.

Si intentamos conectar el motor directamente a la placa Arduino como demuestra la "figura 7", será un poco peligroso ya que los motores de corriente continua son verdaderas fuentes de interferencias que puedan dañar nuestra placa, aparte que generan una gran cantidad de ruido en relación con la tensión de alimentación, además la corriente que consumen los motores suele ser muy elevada en comparación con la que pueda proporcionar una salida digital que suele ser alrededor de 40mA.
Entonces buscamos otra solución para completar este objetivo

2.7.1. Esquema control motor a través de transistor bipolar

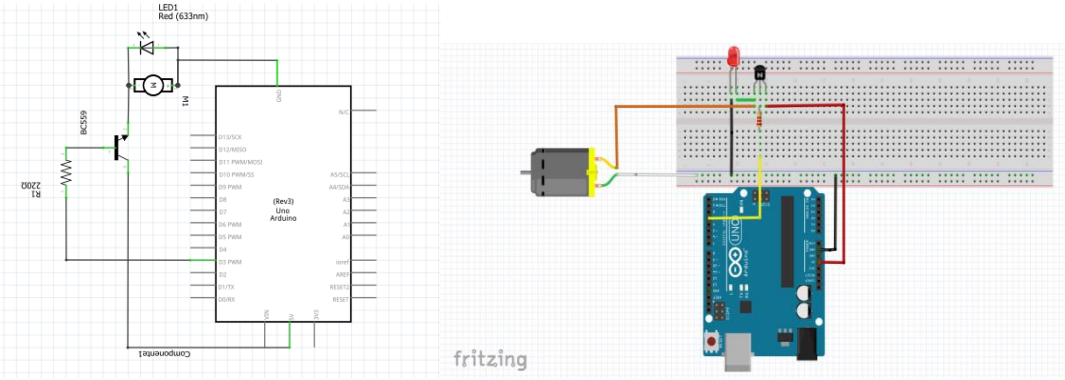


FIGURA 8: CIRCUITO DE CONTROL ARDUINO MOTOR CON TRANSISTOR/ FIGURA 9 : MONTAJE CONTROL ARDUINO MOTOR CON TRANSISTOR

En la dos "figuras8, 9" se observa que la base del transistor atreves de una resistencia es controlada por la salida 3 de la placa Arduino UNO.
Estado salido 3 esta LOW, 0V→Transistor OFF corriente que pasa es casi nula →Motor OFF
Estado salido 3 esta HIGH,1V→Transistor ON pasa corriente por el transistor →Motor ON
A continuación, vamos a crear dos códigos, el primero para mover el motor en un sentido y el segundo código para variar la velocidad del motor.

2.7.2. Código2_control motores DC a través un transistor

```
// Control velocidad Motor con transistor bipolar.
 const int control = 3;
void setup()
{
    pinMode(control,  OUTPUT) ;// declaramos pin3 como salida
}

void loop()
   {
       digitalWrite(control, HIGH);// pin3→nivel alto
       delay(2000);// 2s de espera led ON
       digitalWrite(control, LOW);// pin3→nivel bajo
       delay(1000);// 1s de espera
   }
```

2.7.3. Simulación real

Una vez compilado y cargado el código en la tarjeta Arduino, se observa que el led se enciende mientras gira el motor durante dos segundos luego se apagan durante un segundo en un ciclo infinito.

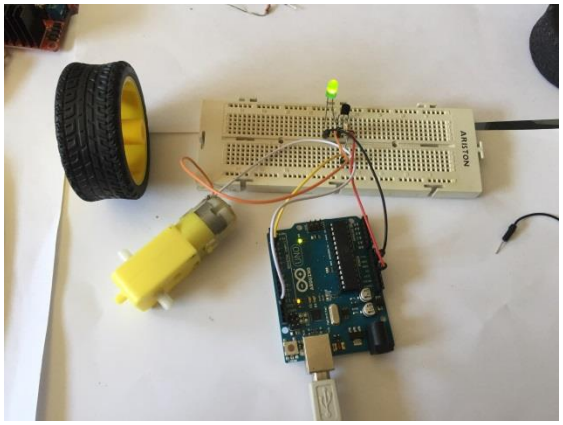


IMAGEN 49: SIMULACIÓN DE CONTROL ARDUINO MOTOR CON TRANSISTOR

Con ese código sencillo podemos ver como gira el motor en un sentido durante 2sy luego se para durante 1s. El led se enciende cuando el motor está en marcha y se apaga cuando se para el motor.

2.7.4. Codigo3_control velocidad motor DC a través un transistor

```
const int control = 11; // Control velocidad Motor con transistor bipolar.
void setup()
{   pinMode(control,  OUTPUT) ; }
void loop()
   {    for ( int n = 0 ; n < 255 ; n++)
        {
           analogWrite (control, n) ;
           delay(15) ;
       } }
```

Con el código3 variamos la tensión en la base del transistor lo que a su vez limita la corriente que lo atraviesa logrando así modificar la velocidad del motor.

2.7.5. Simulación real

Al cargar el nuevo código a nuestra tarjeta Arduino, se observa que el led se vaya iluminando más a la vez que la velocidad del motor subí como demuestra las imágenes a continuación.

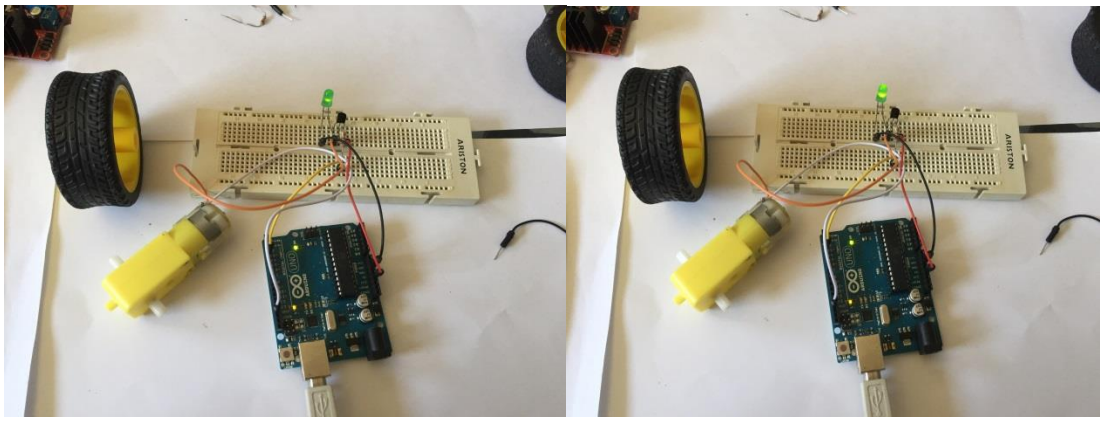


IMAGEN 50: SIMULACIÓN DE CONTROL ARDUINO VELOCIDAD DEL MOTOR CON TRANSISTOR.

2.7.6. Desventajas del Transistor

Con los transistores es más sencillo controlar un motor, pero no olvidemos de sus desventajas que nos puedan suponer un gran problema en nuestro proyecto:

- No admite altas tensiones
- Se calienta con altas corrientes.
- Sensible al ruido y puede ser dañado
- Control de sentido del motor

2.7.7. Solución puente H

Primero añadir un disipador de calor para transistor para lograr que se enfrié segundo, podemos cambiar el transistor por alguno que acepta corrientes más altas como el caso de un Mosfet en comparación con el consumo real de nuestro motor. Por último, un condensador de filtro, una vez solucionado dicho problema nos queda solucionar el doble sentido de movilidad de los motores, cosa que solo un puente en H nos pueda resolver.

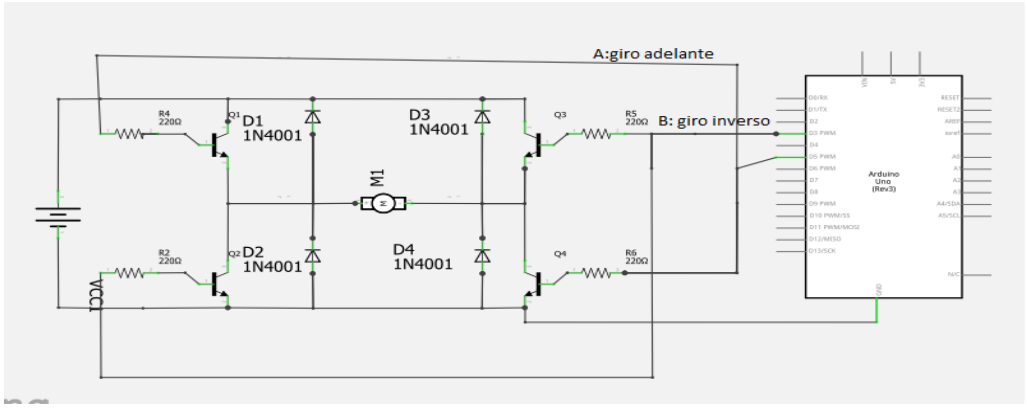


FIGURA 10: CIRCUITO PUENTE EN H CON TRANSISTORES

Como se ve en la "figura 10" consta de cuatro transistores de los cuales solo dos se activan cada vez, también aparecen los diodos que son muy importantes para proteger el circuito, su trabajo consiste en proteger los transistores de los picos de voltajes elevados que se producen en el cambio de sentido de un motor.
A continuación, veamos cómo queda nuestro montaje simulado en Fritzing

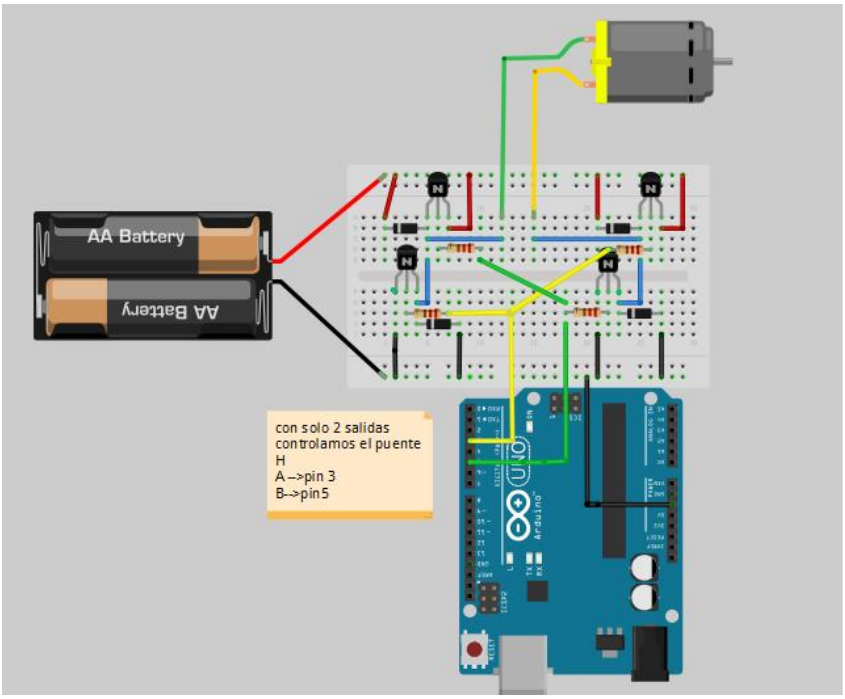


FIGURA 11: MONTAJE PUENTE EN H, FRITZING.

Ahora podemos controlar el sentido del motor en ambas direcciones, pero queda un problema de corriente y posibilidad de sobrecalentar de los transistores, además ocupan mucho sitio cosa que no nos conviene, como primera solución nos ocurre el integrado l293D que tiene 2 puentes en H-bridge y diodos de protección integrados, que vemos sus características a continuación.

2.7.8. Driver potencia con integrado L923D

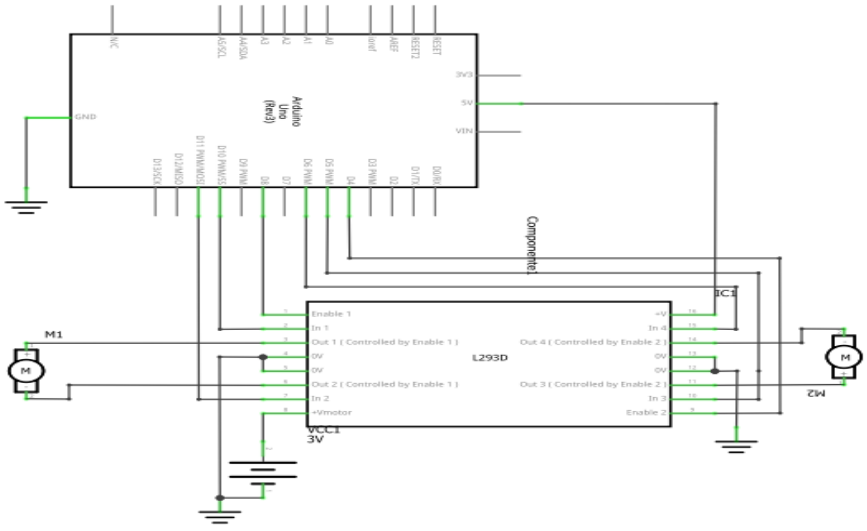


FIGURA 12: CIRCUITO ARDUINO CONTROL MOTORES CON L293D

2.7.9. Tabla de sentido de giro de los motores

D8-ENA	D10-IN1	D11-IN2	D4-ENB	D5-IN3	D6-IN4	Estado Motor1	Estado Motor2
LOW	0	0	0	0	0	Parado	Parado
HIGH	HIGH	LOW	HIGH	LOW	HIGH	Giro adelante	Giro inverso
HIGH	LOW	HIGH	HIGH	HIGH	LOW	Giro inverso	Giro adelante
HIGH	HIGH	HIGH	HIGH	LOW	LOW	Motor parado	Motor parado
HIGH	LOW	LOW	HIGH	HIGH	HIGH	Motor parado	Motor parado

TABLA 4: SENTIDO DE GIRO DE LOS MOTORES, L293D

2.7.10. Conexiones Arduino-L293D

L293D	Arduino UNO	Descripción
1	**D10**	Enable 1,2
2	**D8**	Input 1
3	**-**	Motor1+
4,5,12,13	**GND**	Masa
6	**-**	Motor 1-
7	**D11**	Input 2
8	**Vin**	Alimentación motores
9	**D5**	Enable 3,4
10	**D4**	Input 3
11	**-**	Motor 2+
14	**-**	Motor 3+
15	***D6***	Input 4
16	***+5V***	Alimentación L293D

TABLA 5: CONEXIONES ARDUINO, CON INTEGRADO L293D.

2.7.11. Circuito L293D con Arduino

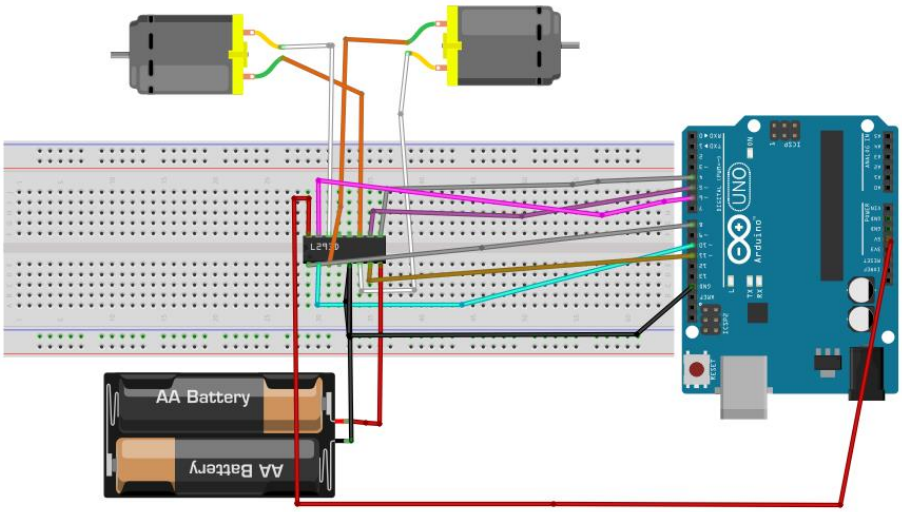


TABLA 6: MONTAJE ARDUINO CONTROL MOTORES CON L293D, FRITZING

2.7.12. Código 4_control motores DC con L293D

/*El código es muy sencillo, activamos Enable 1 y Enable 2 para arrancar motores y luego utilizamos I1, I2 con valores invertidos y lo mismo con I3, I4, así logramos el giro del motor en la dirección contraria.
El motor arranca durante 2 segundos se para, levantamos en Enable al paso de 1 segundo e intercambiamos los valores de I1 e I2 por una parte y I3 e I4 así que el giro de los motores se inicia en dirección contraria durante 2s, luego paramos los motores durante 1s y empezamos de nuevo ya que es un bucle infinito.

```
#define EN1 10 // habilitar pines motor 1
#define I1 8     // Control pin 1motor 1
#define I2 11 // Control pin 2 motor 1
#define EN2 5 // habilitar pines motor 2
#define I3 4     // Control pin 1 motor 2
#define I4 6     // Control pin 2 motor 2

void setup()
{ pinMode( 4, OUTPUT);  // Inicializamos los pines como salida
 pinMode( 5, OUTPUT);
  pinMode( 6, OUTPUT);
 pinMode( 8, OUTPUT);
 pinMode( 10, OUTPUT);
 pinMode( 11, OUTPUT);
  }
void loop()
  {   digitalWrite(E1, HIGH);     // Activamos Motor1
      digitalWrite(I1, HIGH);     // Arrancamos sentido horario
      digitalWrite(I2, LOW);
     digitalWrite(E2, HIGH);     // Activamos Motor2
      digitalWrite(I3, HIGH);     // Arrancamos sentido horario
```

```
    digitalWrite(I4, LOW);
    delay(2000);

    digitalWrite(E1, LOW);      // Paramos Motor 1
    digitalWrite(E2, LOW);      // Paramos Motor2
    delay(1000);
    digitalWrite(E1, HIGH);     // Activamos Motor1
    digitalWrite(E2, HIGH);     // Activamos Motor2

    digitalWrite(I1, LOW);   // Arrancamos con cambio de dirección antihorario
    digitalWrite(I2, HIGH);             Motor1
    digitalWrite(I3, LOW);   // Arrancamos con cambio de dirección antihorario
    digitalWrite(I4, HIGH);             Motor2
    delay(2000);

    digitalWrite(E1, LOW);      // Paramos Motor 1
    digitalWrite(E2, LOW);      // Paramos Motor 1

    delay(1000);
}
```

Observaciones:

Debido a la limitada corriente que nos proporciona el l293D que es 600mA que nos pueda perjudicar a lo largo del desarrollo del robot, aparte dicho integrado tiene mínima disipación térmica y se calienta especialmente con mayores voltajes de la batería, la mejor solución será el L298N que es como el hermano mayor de l293D cuyas características lo veamos a continuación.

2.7.13. Driver de potencia L928N Dual-H-Bridge

El L298N de la “imagen30” es un controlador de motores, que permite alimentar y controlar dos motores DC de corriente continua a través Arduino, variando sus velocidades y cambiando su sentido de giro.

En teoría la función de un microcontrolador no debe ser ejecutar acciones si no consiste en mandar y ejecutar acciones a drivers que realicen esa tarea.
A demás los microcontroladores no disponen de potencia suficiente para mover motores.

La corriente máxima que suministra el L298N es de 2A por y una tensión de alimentación de 3V a 35V.

Debido a su eficiencia baja. La electrónica supone una caída de tensión de unos 3V que se disipa en forma de calor es decir que, que no podamos obtener más de 0.8-1A por fase sin exceder el rango de temperatura de funcionamiento.

2.7.13.1. Circuito interno módulo L298N

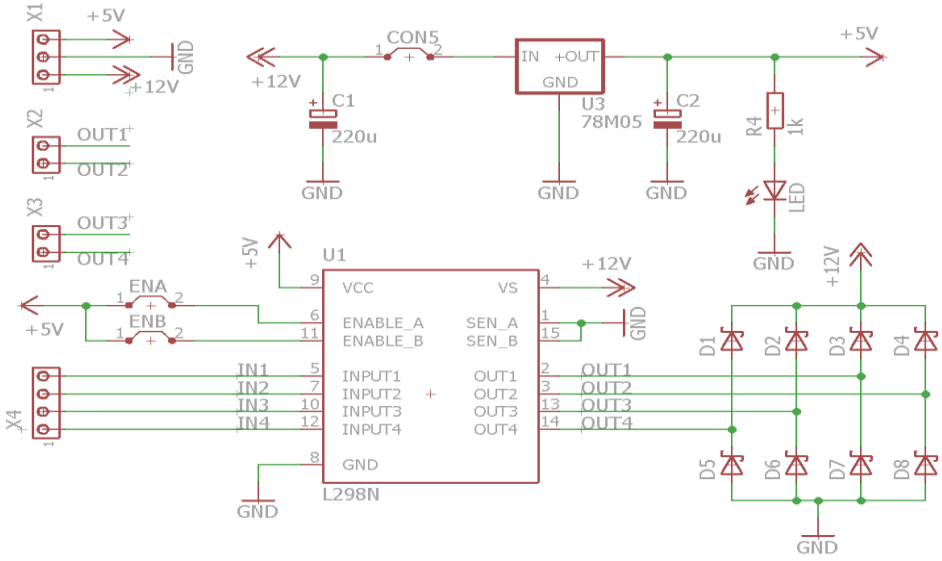


TABLA 7: CIRCUITO INTERNO MÓDULO DE POTENCIA L298N

2.7.13.2. Ventajas L298N

- ✓ Soporta una tensión máxima de 46 voltios.
- ✓ incorpora protecciones contra corrientes inducidas y efectos que pueden producirse al manejar motores DC.
- ✓ Proporciona hasta 2A de corriente de funcionamiento
- ✓ Dispone de protecciones contra picos de intensidad, temperatura, y diodos de protección contra corrientes inducidas por el cambio de giro de los motores
- ✓ Soporta picos ocasionales 3A pico repetitiva a 2,5 A.
- ✓ TTL compatible, fácil de conectar con la tarjeta Arduino.
- ✓ VCE_Sat caída de tensión típica suele ser de de 1,8 voltios,
  IL = 3.2V→ 1A, IL = 4.9V→ 2A, eso nos permite conectar una batería de 5V a 7V directamente con un motor.
- ✓ La caída de tensión VCE_Sat Puente-H será suficiente para ajustar la tensión del motor que suele ser +/- 5V.
- ✓ Calidad precio, Bajo coste.

2.7.13.3. Configuración Arduino con el L298N

Como hemos mencionado antes el driver de potencia l298N H-bridge nos permite controlar velocidad y la dirección de giro de dos motores gracias a sus dos puentes H-bridge.
Trabaja en un rango de 3V a 35V →Vin, proporciona una intensidad de hasta 2A, pero hay que tener en cuenta que los motores reciben 3V menos de la tensión de alimentación del módulo que la consume la electrónica de dicho modulo.

2.7.13.4. Configuración alimentación

Si suministramos de 6 hasta 12 V máximum por la Vcc y el jumper del regulador que se ve en la "ilustracion25" está cerrado se activa el regulador de tensión del L298N VLógica o donde tenemos 5V tendremos 5V de salida, que podremos usar para alimentar la placa Arduino u otros sensores.
Si quitamos el jumper se desactiva el regulador, y tendremos que alimentar la parte lógica del módulo, así que tendremos que meter una tensión de +5V por la conexión 5V para que el módulo funcione.
En caso de introducir corriente por V lógica de 5V teniendo el jumper activado podemos dañar seriamente nuestro modulo.
Las demás conexiones se conectan depende si el objetivo es manejar dos motores de continua o un motor paso a paso, en nuestro caso será el control de motores DC.
También podemos aprovechar la salida de 5V que nos proporciona el módulo para alimentar la mismísima placa Arduino, en nuestro caso los sensores que se ocupan de detectar la línea negra de nuestro robot.

2.7.13.5. Control motor a través del driver L298N

Las salidas OUT1, OUT2 - OUT3, OUT4 para los motores M1 y M2 consecutivamente nos proporcionan la energía necesaria para mover los motores.
La polaridad de los motores se debe tener en cuenta, para que los motores giren en el sentido correcto, así que soldamos cables a nuestros motores con el mismo color (color blanco) para polaridad inversa como se ve a continuación para facilitar conexiones y el correcto funcionamiento.

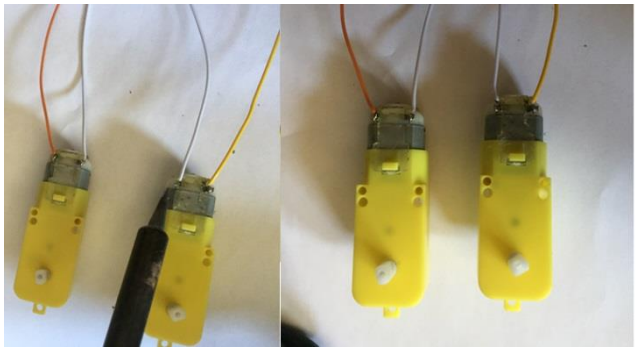


IMAGEN 51: CONECTORES PARA MOTORES

Los pines IN1 e IN2 nos sirven para controlar el sentido de giro del motor A, y los pines IN3 e IN4 el del motor B. 
Funcionan de forma que si IN1 está a HIGH→nivel alto e IN2 a LOW→nivel bajo, el motor M1 gira en un sentido, y si está IN1 →LOW e IN2 → HIGH el motor M1 gira en el otro sentido.

IN3 está a HIGH→nivel alto e IN4 a LOW→nivel bajo, el motor M2 gira en un sentido, y si está IN3 →LOW e IN4 → HIGH el motor M2 gira en el otro sentido.

2.7.13.6. Control velocidad motores

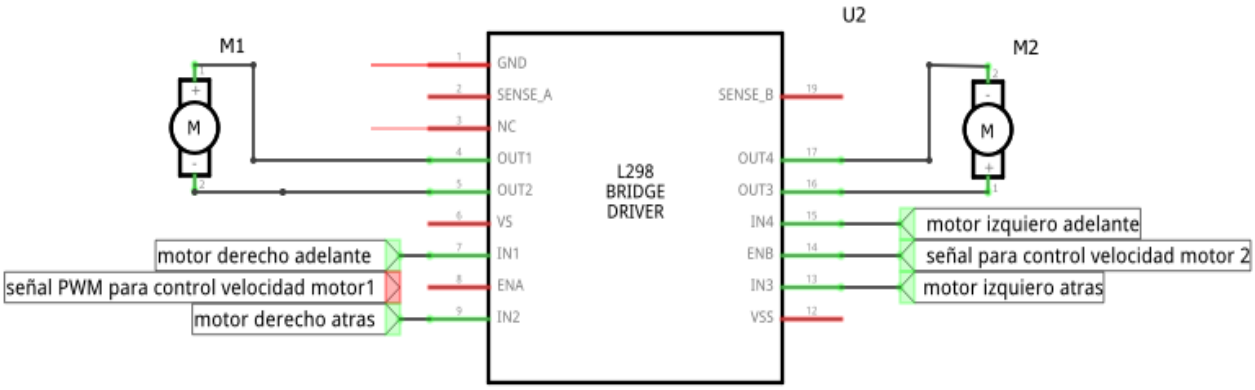


FIGURA 13: CONTROL VELOCIDAD MOTORES

La señal PWM no la podemos introducir por IN1, IN2, IN3, IN4, sino por las entradas de habitación ENA, ENB, así evitamos el freno dinámico se produce durante los tiempos en baja señal PWM, lo que provocaría resultados no deseados.
Al controlar la velocidad de los motores con señales de pulsos, podemos calcular su velocidad de giro mediante la expresión:

$V = CT * Vo$, siendo $CT = \frac{Ton}{Ton+Toff}$

V: velocidad de giro para un determinado ciclo
Vo: velocidad de giro máxima
CT: ciclo de trabajo
Ton: tiempo en nivel alto
Toff: tiempo en nivel bajo

Para el control de velocidad de los motores, hay que quitar los jumpers que habilitan los pines ENA y ENB, conectarlos a las salidas PWM de Arduino de forma que le enviemos un valor comprendido entre:

- 0→Motor parado.
- 255→Máxima velocidad.

Así logramos controlar la velocidad de giro dependiendo del dutty, cuando más grande el valor del dutty más velocidad, como explique en el apartado de PWM.

Nota: Si tenemos habilitadas las entradas (ENA, ENB)→HIGH, los motores girarán siempre a la misma velocidad.

El esquema de montaje que vamos a utilizar es el siguiente, se puede usar cualquier pin solo dejamos los pines D0 (RX), D1 (TX) que utilizamos más adelante para Comunicación Bluetooth, y debemos escoger dos pines de PWM para ENA y ENB.

2.7.13.7. Conexiones Arduino _L298N

Módulo L298N	*Arduino uno*	*descripción*
VCC	**-**	Batería externa 6V
GND	**GND**	Se debe juntar las masas al usar dos fuentes distintas
5V	**-**	Teniendo puesto el jumper de la alimentación salida 5V
ENA	**Pin 3→PWM**	Control velocidad motor1
INT1	**Pin 2**	Control sentido motor 1
INT2	**Pin4**	Control sentido motor 1
ENB	**Pin9→PWM**	Control velocidad motor 2
IN3	**Pin 7**	Control sentido motor 2
IN4	**Pin 8**	Control sentido motor 2
OUT1	-	Motor 1 adelante
OUT2	-	Motor 1 atrás
OUT3	-	Motor 2 adelante
OUT4	-	Motor 2 atrás

TABLA 8: CONEXIONES ARDUINO DRIVER L298N

2.7.13.8. Montaje y funcionamiento

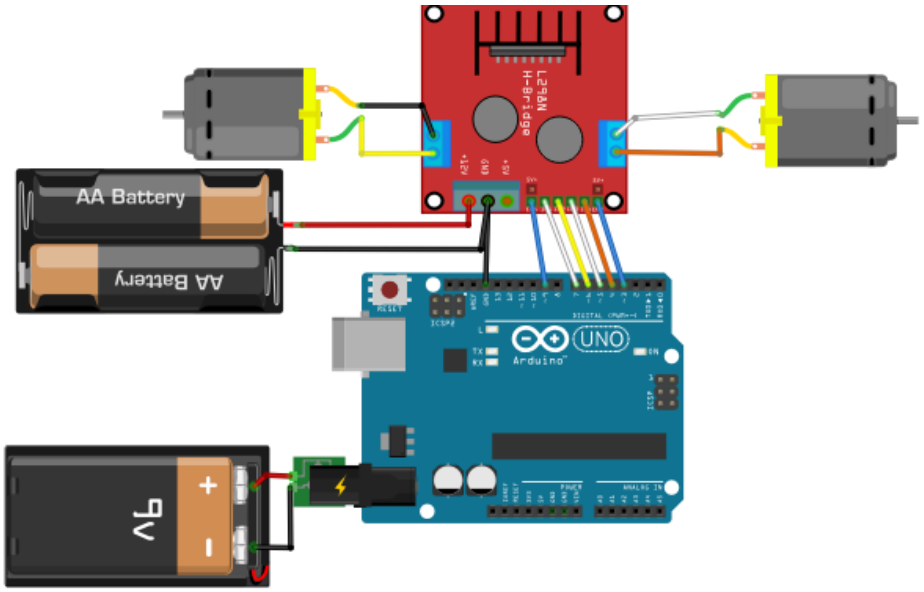


FIGURA 14 : MONTAJE CONTROL ARDUINO MOTORES CON DRIVER L298N

Como se ve en "figura 14", alimentamos la tarjeta Arduino mediante una batería de 9V a través de un conector Jack-Macho directamente a Arduino, y el driver L298N con otra batería de 6V.
A continuación, una tabla de conexiones para facilitar el montaje.

NOTA: debemos juntarlas masas GND ya que usamos dos fuentes de alimentación distintas

2.7.13.9. Código5_Control giro y velocidad motores DC con driver L298N

```
//conexiones Motor 1
int enA = 3;
int in1 = 2;
int in2 = 4;

// Conexiones Motor 2
int enB = 9;
int in3 = 7;
int in4 = 8;
int const vel=250;//  vel =velocidad de los motores
void setup()
{
  //declarando salidas control motores
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
}
void adelante_atras() // función que hace que los motores avanzan y retrocedan a  maxima velocidad .
{
// motor m1 adealnte
  digitalWrite(in1, HIGH);//
  digitalWrite(in2, LOW);//
  // Velocidad del Motor A (puede cambiar de 0 a 255 max)
  analogWrite(enA, vel);
 // Motor m2 adelante
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  // Velocidad del Motor 2 (puede cambiar de 0 a 255max)
  analogWrite(enB, vel);
  // 3s deespera
  delay(3000);
 // Los Motores m1 y m2 cambian su dirección de giro
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in4, HIGH);
  digitalWrite(in3, LOW);
  // 3s de espera
  delay(3000);
 // Parada de los dos Motores.
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay(2000);
}
```

```
void incrementa_decrementa()//función hace que los motores avanzan incrementado su velocidad y retrocedan decrementando su velocidad .
{
  // Avanzar adelante los motores M1 y M2 .
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);

 // Incrementar velocidad de 0 a 250, acelerando
  for (int i = 0; i < vel; i++) // bucle que va incrementado el valor i=0 y comparándolo hasta llegar al valor de (vel) 250
  {
    analogWrite(enA, i);
    analogWrite(enB, i);
    delay(20);
  }

  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay(3000);

  // decrementar velocidad de 250 a 0, Desacelerando
  digitalWrite(in1,LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);

  for (int i = vel; i >= 0; --i)//bucle decrementa el valor de i=250 hasta llegar al 0.
  {
    analogWrite(enA, i);
    analogWrite(enB, i);
    delay(20);
  }
  // Parada de los dos Motores.
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay(3000);
}
```

```
void loop()
{
  adelante_atras();//ejecutamos función adelante_atrás así el robo avanzará  y se retrocederá a máxima velocidad
  delay(1000);
  incrementa_decrementa();// ejecutamos función así el robo avanzará acelerando  y se retrocederá  desacelerando.
  delay(1000);
}
```

2.7.13.10. Montaje real de la parte potencia de nuestro robot

Una vez tengamos claro las conexiones y el código procedemos al montaje real como se demuestra a continuación.

- ✓ Primero empezamos con soldar los cables a los bornes de cada motor, lo mejor es hacerlo antes de montar el motor en el chasis que es más fácil.
- ✓ Cable amarillo: M1 adelante
- ✓ Cable anaranjado: M2 adelante
- ✓ Cable blanco: giro inverso motores 1, 2

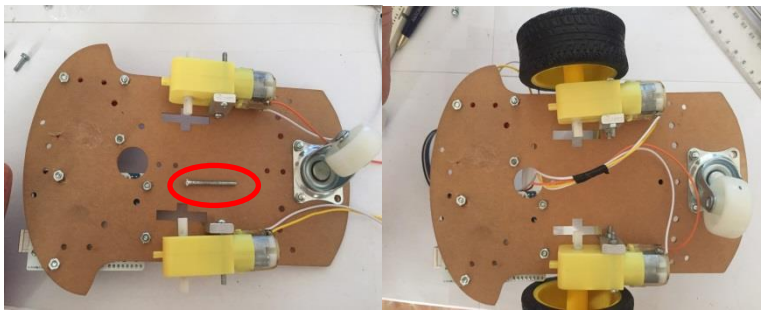


IMAGEN 52: MONTAJE DE MOTORES EN EL CHASIS

Un ves tenemos soldados los cables, sujetamos los motores a los soportes con el tornillo largo que se ve en la “imagen 52” izquierda, el hueco lo tienen en la misma cara donde se coloca las ruedas.

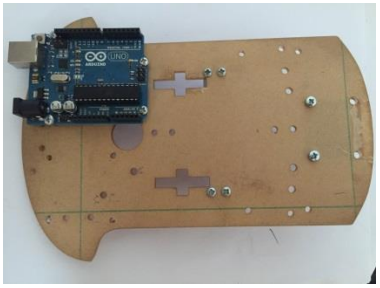


IMAGEN 53: MONTAR TARJETA ARDUINO UNO EN EL CHASIS

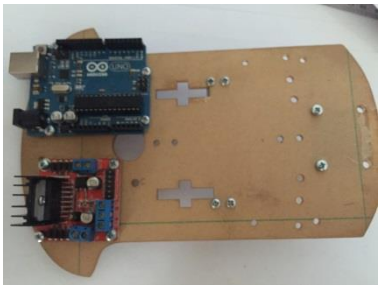


IMAGEN 54: MONTAR DRIVER L298N EN EL CHASIS

Una ves tenemos la estructura mecanica de nuestro robot echa procedemos a montar la tarjeta Arduino y el driver de potencia como se ve en la “imgene 54”,estan muy bien sujetos mediante cuatro tornillos y cuatro tuercas cada uno.

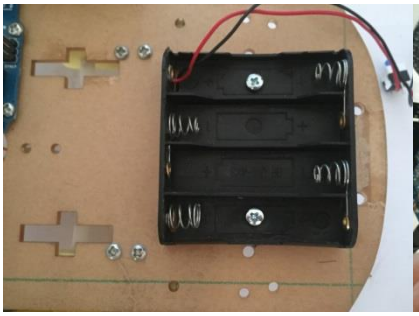


IMAGEN 55: MONTAR PORTA-PILAS

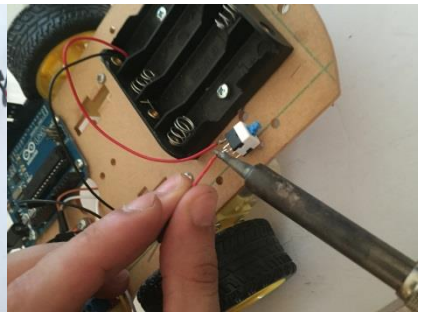


IMAGEN 56 : SOLDAR INTERRUPTOR

Finalmente el porta-pilas lo ponemos en un extremo encima de la rueda loca , el motivo es dejar espacio para la mini-borad que nos hace falta para conectar y alimentar sensores.
Tambien soldamos un pequeño interuptor ,que nos permita cortar la alimentacion a los motores si lo deseamos.

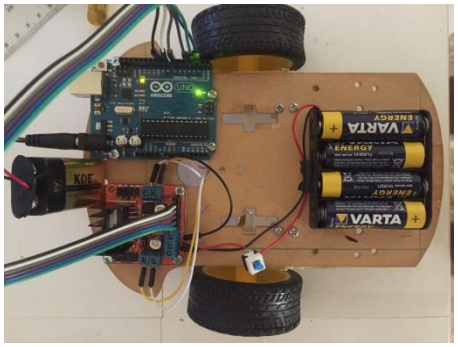


IMAGEN 57: BASE ROBOT

Finalmente, nos queda poner las cuatro pilas AA de 1.5V en el porta-pilas que alimenta el driver de potencia L298N y por último la pila de 9V conectada al Jack-macho que alimenta a su vez la placa Arduino.
Una vez tenemos dominado el control de los motores pasamos a la configuración de los sensores TCRT500.

2.8. CONFIGURACIÓN ARDUINO CON SENSORES TCRT500

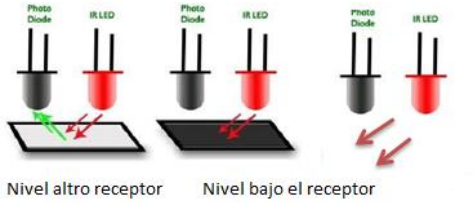


ILUSTRACIÓN 30: FUNCIONAMIENTO SENSOR TCRT500

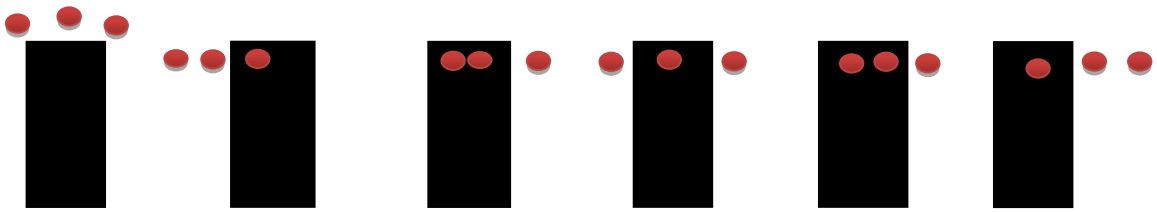
Nuestro sensor como explicamos antes está compuesto por un led emisor de infrarrojo y un fototransistor receptor, colocados uno al lado del otro como se ve en la foto anterior.
Si la luz emitida por el led se choca con una superficie blanca se reflejará y la recibirá el fototransistor, pero si no choca con ninguna superficie o choca con una superficie negra no reflejará casi luz y no llegará nada al receptor.

2.8.1. Objetivo

Nuestro objetivo principal se basa en corregir el recorrido de nuestro robot sobre la línea consta de seise órdenes:

I. Ningún sensor esta sobre la línea buscar línea
II. Sensor derecho en línea, giramos izquierda
III. Sensor derecho y sensor central en línea avanzamos girando a la izquierda
IV. Sensor central en línea avanzamos
V. Sensor central y sensor izquierdo en la línea avanzamos girando a la derecha
VI. Sensor izquierdo en línea giramos derecha

A continuación, una situación de cada orden de los sensores de izquierda 1 a derecha 6.



En el caso de los sensores elegidos, tiene montado en el módulo un integrado LM393 que se ocupa de generar una simple salida digital HIGH=nivel alto, LOW=nivel bajo, también tiene incorporado un potenciómetro que ajustara la sensibilidad del sensor a la luz.

2.8.2. Funcionamiento sensores TCRT5000

Los sensores TCRT5000 funcionan de manera que cuando la superficie es negra o más bien no se refleja ninguna luz, se genera por su salida OUT un nivel alto, y un nivel bajo si recibe luz o más bien la superficie es blanca o clara.
Los tres sensores estarán montados de tal manera que si solo un sensor esta encima de la línea negra producirá un estado alto solo.

2.8.3. Conexiones sensor TCRT5000 con Arduino

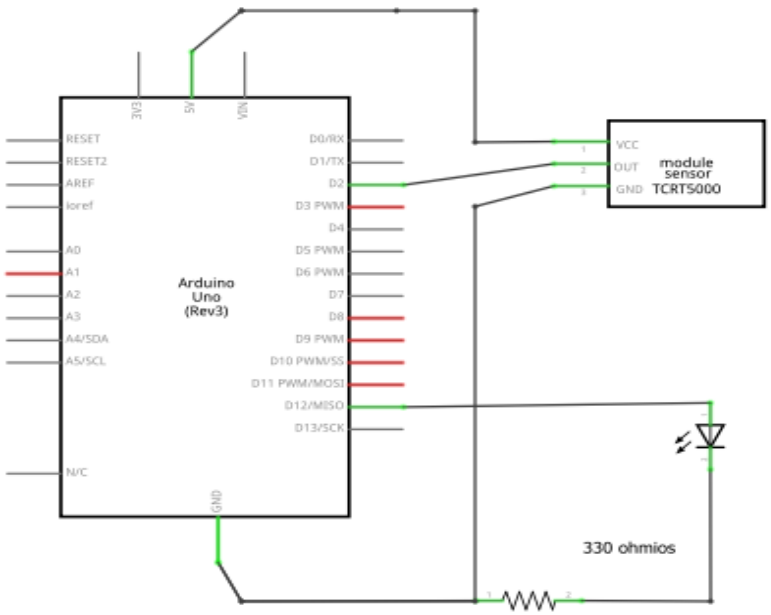


FIGURA 15: CIRCUITO ARDUINO CON SENSOR TCRT500

Para comprobar dicho sensor hemos conectado la salida OUT del sensor al pin digital 2 de la tarjeta Arduino y el pin 12 al led, así que el pin 2 recibirá un LOW si está encima de una superficie blanca y el led permanecerá apagado, en contrario recibirá un HIGH si está encima de una superficie negra y se encenderá el led.
Montamos el circuito de la "figura 15" anterior para comprobar el funcionamiento de los sensores TCRT5000.
Vamos a implementar un código simple, hará que el valor del pin 2 pasara a la entrada del led.
Superficies Blanca→pin 2 LOW (led sensor ON)→pin 12 LOW→LED OFF
Superficies Negra →pin 2 HIGH (led sensor OFF)→pin 12 HIGH→LED ON

2.8.4. Código 6_ensayo sensor TCRT500

```
int LED = 12 ; // Entrada digital conectada al LED
int TCRT5000 = 2; //Entrada digital conectada al sensor infrarrojo

void setup()
{
 pinMode( TCRT5000 , INPUT) ; // definimos el Sensor como entrada
 pinMode( LED, OUTPUT) ; //  definimos el LED Como salida

}

void loop()
{
 int valor = digitalRead(TCRT5000) ; //leemos el valor del sensor infrarrojo
 digitalWrite( LED, valor) ; // pasamos elvalor del sensor al led
}
```

2.8.5. Simulación prueba TCRT5000

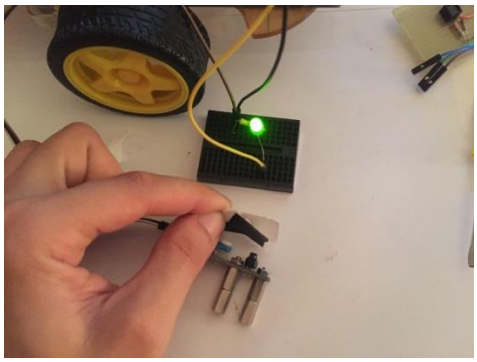


IMAGEN 58: TCRT500 ENCIMA DE SUPERFICIE NEGRA

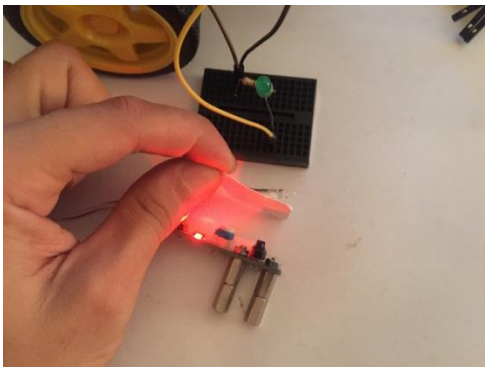


IMAGEN 59: TCRT50 ENCIMA DE SUPERFICIE BLANCA

En las Imágenes anteriores obtenemos los resultados deseados, así nos aseguramos del buen funcionamiento de los tres sensores.

2.8.6. Diseño, realización y configuración del prototipo

Una vez comprobadas las conexiones y el funcionamiento de los tres sensores procedemos a montar los sensores en el soporte diseñado específicamente para que estén posicionados de manera que cumpla nuestros objetivos.

IMAGEN 60: SENSOR TCRT500 CON SEPARADORES

Primero conectamos dos separadores en serie para cada hueco y los atornillamos a los tres sensores. Después montamos los sensores en el soporte de tal manera que quedan como la “imagen 61” siguiente.

IMAGEN 61: SENSORES TCRT500 MONTADOS EN SOPORTE

IMAGEN 62: CONECTORES SENSORES TCRT500

Una ves estan bien sujetos los sensores al soporte conectamos tres cables macho-hembra a cada sensor y lo sacamos de los huecos rediseñados para ello ,asi evitamos que toquen las ruedas.
Finalmente solo queda acoplar el soporte con los tres sensores al chasis del robot como se ve en la “imagene 64” a continuacion.

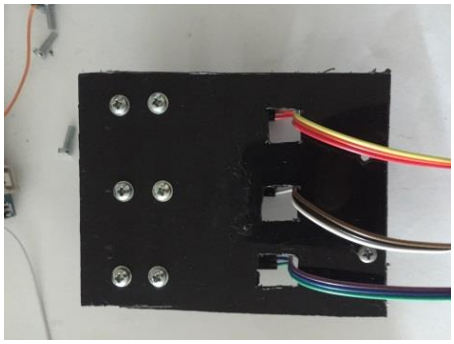


IMAGEN 63: MODULO SEGUIDOR DE LÍNEA

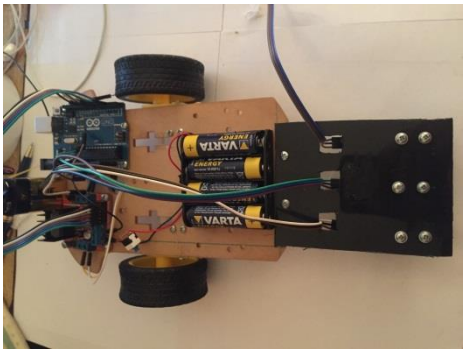


IMAGEN 64: ACOPLAR MODULO SEGUIDOR LÍNEA AL CHASIS

Despues de montar los soportes, nuestro robot empieza a tener forma de seguidor de linea solo queda alimentar los sensores y hacer la ultima prueba antes de empezar a desarollar el codigo.

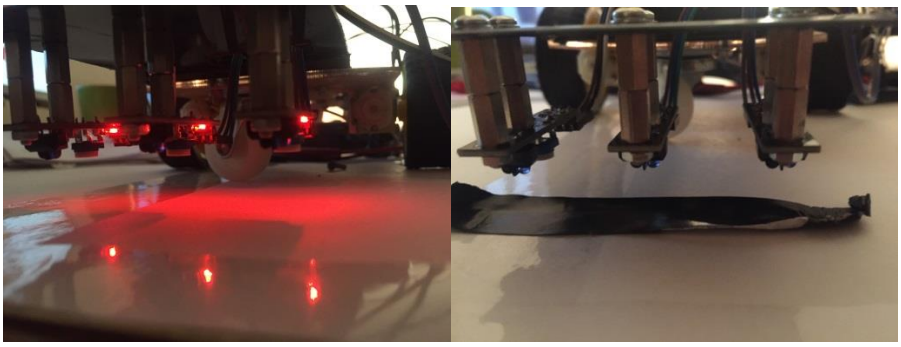


IMAGEN 65: RESPUESTA MODULO SEGUIDOR DE LÍNEA.

La ultima prueba antes de desarollar el codigo si el robot esta encima de una superficie planca los leds del sensor se encienden y tendremos IOW=0V en las salidas OUT de los sensores como se ve en la imagen superior izquierda, pero si el robot esta encima de una superficie negra obsrevamos los leds internos apagados y las salidas de los sensores tendran un nivel alto
HIGH =5V.

2.8.7. Conexiones Arduino _Sensores TCR5000

Sensores	Arduino UNO	descripción
OUT sensor izquierdo	**Pin 11**	Sensor 1
OUT sensor central	**Pin 12**	Sensor 2
Out sensor central	**Pin 13**	Sensor 3
Vcc	**5V**	Alimentación sensores
Gnd	**Gnd**	Tierra común

TABLA 9: CONEXIONES ARDUINO MODULO SEGUIDOR LÍNEA

NOTA : se mantienen las mismas conexiones de la "tabla 8",para conectar el driver de potencia con la placa Arduino y los motores.

2.8.8. Circuito seguidor de linea

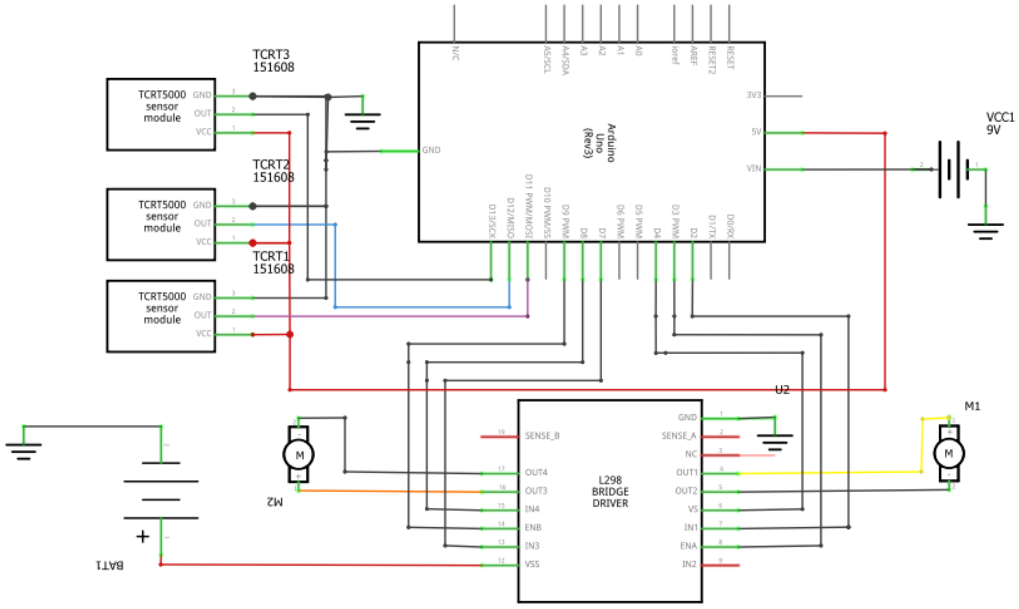


FIGURA 16: CIRCUITO SEGUIDOR DE LÍNEA

2.8.8.1. funcionamiento seguidor de línea

Como se muestra el circuito anterior tenemos montado nuestro circuito de tal manera que los sensores y el control de los motores será lo siguiente:

- ✓ S1 está solo encima de la línea negra el motor izquierdo M2→ON mientras el M1→OFF, robot girara a la derecha para corregir se recorrido con velocidad.
- ✓ S1 y S2 están encima de la línea negra, motores M1, M2→ON, pero el M2 tendrá más velocidad que el M1, así logramos avanzar el robot mientras corrige el recorrido a la derecha.
- ✓ S2 solo encima de línea negra, motores M1, M2→ON con la misma velocidad adelante.
- ✓ S2, S3 encima de línea negra, motores M1, M2→ON, motor M1 avanzara con más velocidad para corregir el recorrido a la izquierda mientras avanza el robot.
- ✓ S3 solo encima de línea negra, motor M1→ON, motor M2→OFF, el robot gira a la izquierda para corregir el recorrido.

Definir variables para simplificar el proceso:

Será como demuestra la "imagen 66" a continuación.

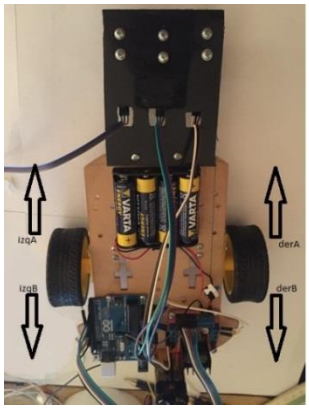


IMAGEN 66: SENTIDO GIRO MOTORES ROBOT

2.8.8.2. Tabla de verdad seguidor de línea

La siguiente variable es de la velocidad de los motores que mandamos por los pines ENA y ENB como señal de pulsos.

Vel1 < Vel2< Vel3

S1 sensor izquierdo	*S2 sensor central*	*S3 sensor derecho*	*Estado motor2*	*Estado motor1*	*izqA*	*izqB*	*derA*	*derB*	*ENB*	*ENA*
0	*0*	*0*	*retroceder*	*retroceder*	*0*	*1*	*0*	*1*	*Vel1*	*Vel1*
0	*0*	*1*	*OFF*	*ON*	*0*	*0*	*1*	*0*	*0*	*Vel1*
0	*1*	*0*	*ON*	*ON*	*1*	*0*	*1*	*0*	*Vel3*	*Vel3*
0	*1*	*1*	*ON*	*ON*	*1*	*0*	*1*	*0*	*Vel1*	*Vel2*
1	*0*	*0*	*ON*	*OFF*	*1*	*0*	*0*	*0*	*Vel1*	*0*
1	*0*	*1*	*OFF*	*OFF*	*0*	*0*	*0*	*0*	*0*	*0*
1	*1*	*0*	*ON*	*ON*	*1*	*0*	*1*	*0*	*Vel2*	*Vel1*
1	*1*	*1*	*OFF*	*OFF*	*0*	*0*	*0*	*0*	*0*	*0*

TABLA 10: TABLA DE VERDAD SEGUIDOR DE LÍNEA

NOTA : se mantienen las mismas conexiones de la "tabla 8",para conectar el driver de potencia con la placa Arduino y los motores.

2.8.9. Montaje seguidor de línea

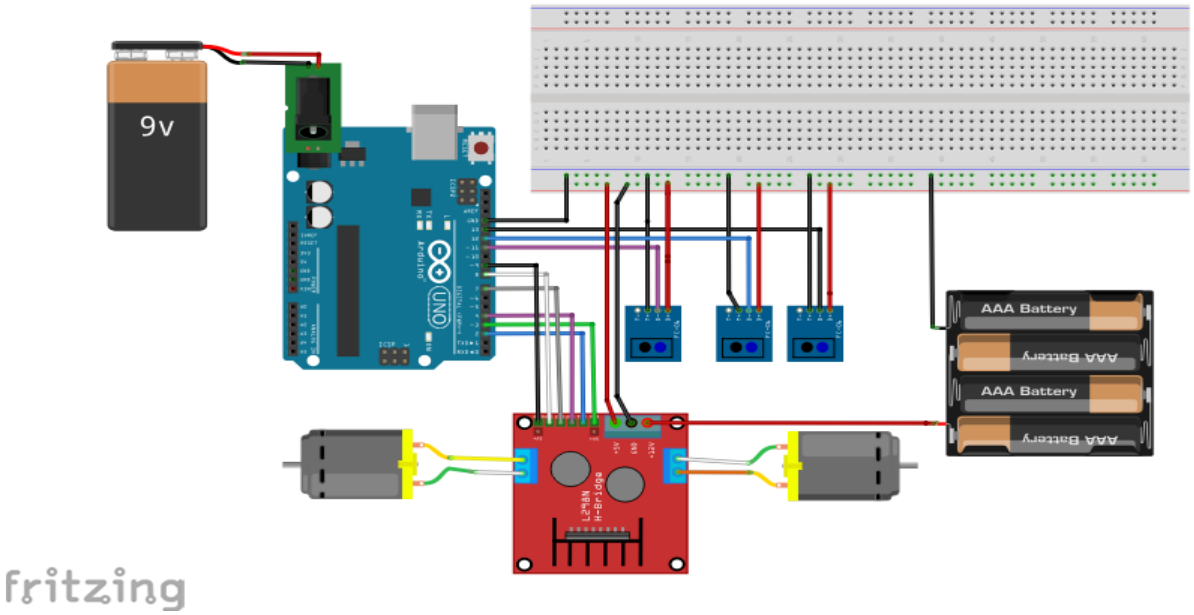


FIGURA 17: MONTAJE SEGUIDOR DE LÍNEA, FRITZING

2.8.10. Diagrama de bloques

Nuestro seguidor de línea funciona de la manera siguiente:

Mientras que el sensor central SCE este encima de una línea negra, el Robot seguirá avanzando en línea recta, pero si el sensor izquierdo IZQ también se pone encima de la línea negra el Robot seguirá avanzando al mismo tiempo corrigiendo su ruta hacia la derecha, pero si el sensor derecho SDR es el que se asuma al central encima de la línea negra el Robot avanzara corrigiendo su ruta a la izquierda.

Si solo el sensor izquierdo SIZ está encima de la línea negra el Robot girara a la derecha.
Si solo el sensor derecho SDR está encima de la línea negra el Robot girara a la izquierda.
A continuación, creamos un diagrama de bloques que nos ayudara a la hora de programar.

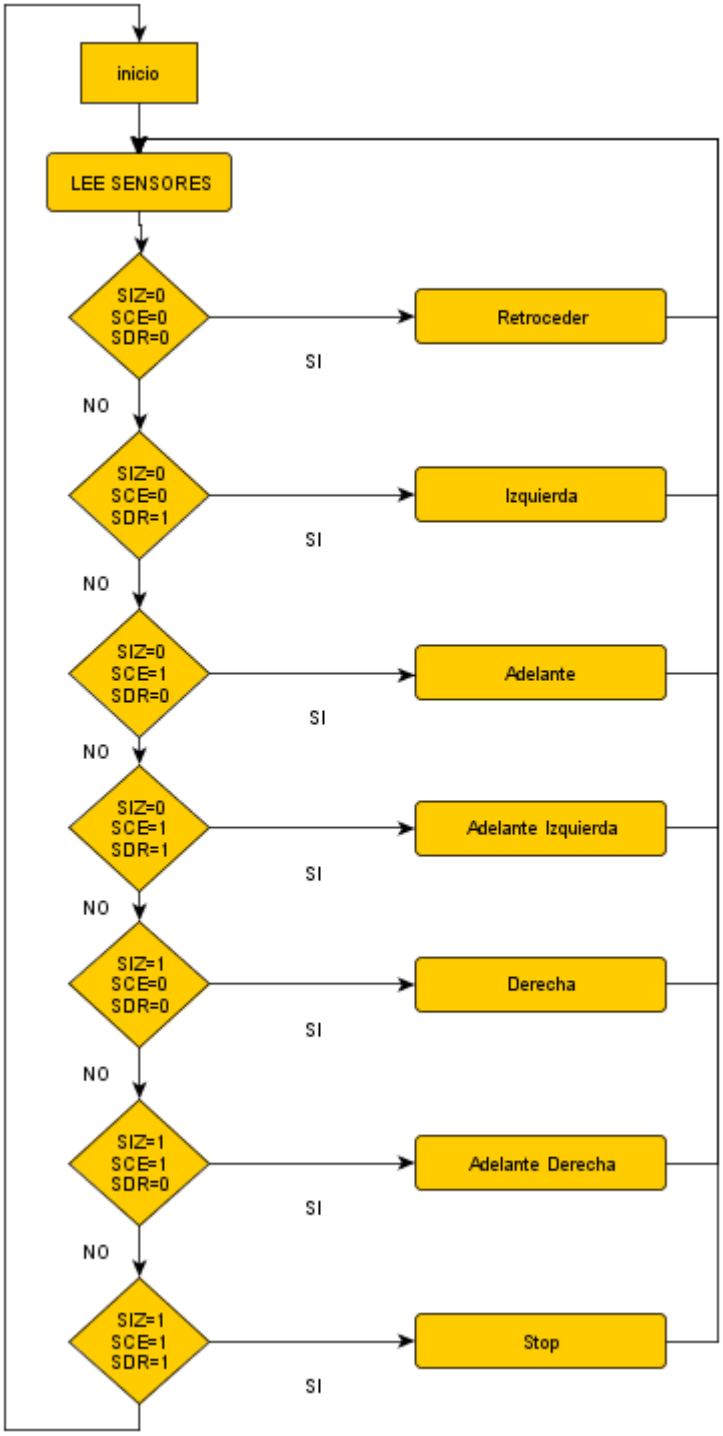


DIAGRAMA DE BLOQUES 1: DIAGRAMA SEGUIDOR DE LÍNEA

2.8.11. Codigo7_Seguidor de línea

```
const int sen1 = 11; // Izquierdo
const int sen2 = 12; //Centro
const int sen3 = 13; //Derecho
//Control de los motores
int ENA=3; //Velocidad motor 1
int ENB=9; //Velocidad motor 2

int derA = 2; //avanzar motor 1
int derB = 4; // atrás motor 1
int izqA = 7; //avanzar motor 2
int izqB = 8;  // atrás motor 2
int vel1 = 90; // Especificar velocidad de Motores
int vel2 =100;
 int vel3 = 120;

void setup()
{
  Serial.begin(9600);
 pinMode(sen1, INPUT);
 pinMode(sen2, INPUT);
 pinMode(sen3, INPUT);
 pinMode(derA, OUTPUT);
 pinMode(derB, OUTPUT);
 pinMode(izqA, OUTPUT);
 pinMode(izqB, OUTPUT);
pinMode(ENA, OUTPUT);
pinMode(ENB, OUTPUT);
}
void loop(){
  int ValSen1 = 0; // Valor del sensor izquierdo
  int ValSen2 = 0;// valor del sensor central
  int ValSen3 = 0;// valor del sensor derecho

 analogRead(sen1 );  //lectura digital de pin -->S1 izquierdo
 analogRead(sen2 );  //lectura digital de pin -->S2 central
 analogRead(sen3 );  //lectura digital de pin-->S3derecho
 delay(500);
//Lectura y registro de los valores de los sensores
  ValSen1 = digitalRead(sen1 );  //                          izquierdo
  ValSen2 = digitalRead(sen2 );  //lectura valor entrada del Sensor central
  ValSen3 = digitalRead(sen3 );  //                           derecho
// lectura de sensores 0= superficie Blanca
  //                1= superficie negra
//para poder visualizar datos obtenidos de los sensores por el puerto serial.
      Serial.println(ValSen1);  //zona oscura=1
      Serial.println(ValSen2);  //
      Serial.println(ValSen3);  //
```

```
if ((ValSen1==0)&&(ValSen2== 0)&&(ValSen3== 0))// Atras
{
    digitalWrite(derB, HIGH);
    digitalWrite(izqB,HIGH );
    digitalWrite(derA, LOW);
    digitalWrite(izqA, LOW);
     analogWrite(ENA,vel2);//motor derecho pwm
     analogWrite(ENB,vel2);//motor izquierdo pwm
      delay(300);
}
if ((ValSen1==1)&&(ValSen2== 0)&&(ValSen3== 0))//DERECHA
{
      digitalWrite(derB, LOW);
     digitalWrite(izqB, LOW);
      digitalWrite(izqA, HIGH);
      digitalWrite(derA, LOW);
     analogWrite(ENA,LOW);//motor derecho pwm
     analogWrite(ENB,vel2);//motor izquierdo pwm
      delay(300);
}
if ((ValSen1==1)&&(ValSen2== 1)&&(ValSen3== 0))//ADELANTE+DERECHA
{
     digitalWrite(derB,LOW);
     digitalWrite(izqB, LOW);
     digitalWrite(derA, HIGH );
    digitalWrite(izqA, HIGH);
     analogWrite(ENA,vel1);//motor derecho pwm
     analogWrite(ENB,vel2);//motor izquierdo pwm
      delay(300);
   }
if ((ValSen1==0)&&(ValSen2== 1)&&(ValSen3== 0))//ADELANTE
{
      digitalWrite(derB, LOW);
      digitalWrite(izqB, LOW);
      digitalWrite(izqA, HIGH);
      digitalWrite(derA, HIGH);
      analogWrite(ENA,vel3);//motor derecho pwm
      analogWrite(ENB,vel3);//motor izquierdo pwm
       delay(300);
       }
if ((ValSen1==0)&&(ValSen2== 0)&&(ValSen3== 1))//IZQUIERDA
{
    digitalWrite(derB, LOW);
    digitalWrite(izqB, LOW);
     digitalWrite(derA, HIGH);
    digitalWrite(izqA, LOW);
     analogWrite(ENA,vel2);//motor derecho pwm
     analogWrite(ENB,LOW);//motor izquierdo pwm
      delay(300);
}
```

```
if ((ValSen1==1)&&(ValSen2== 0)&&(ValSen3== 1))//STOP
{
     digitalWrite(derB, LOW);
     digitalWrite(izqB, LOW);
     digitalWrite(derA, LOW);
     digitalWrite(izqA, LOW);
      analogWrite(ENA,LOW);//motor derecho pwm
     analogWrite(ENB,LOW);//motor izquierdo pwm
      delay(300);
}
if ((ValSen1==1)&&(ValSen2== 1)&&(ValSen3== 0))//ADELANTE +IZQUIERDA
 {  digitalWrite(derB, LOW);
    digitalWrite(izqB, LOW);
    digitalWrite(derA, HIGH);
    digitalWrite(izqA, HIGH);
     analogWrite(ENA,vel2);//motor derecho pwm
     analogWrite(ENB,vel1);//motor izquierdo pwm
     delay(300);
}
if ((ValSen1==1)&&(ValSen2== 1)&&(ValSen3== 1))//STOP
{   digitalWrite(derB, LOW);
    digitalWrite(izqB, LOW);
    digitalWrite(derA,LOW);
    digitalWrite(izqA, LOW);
     analogWrite(ENA,LOW);//motor derecho pwm
     analogWrite(ENB,LOW);//motor izquierdo pwm
      delay(300);
}
}
```

2.9. RESUMEN

Cuando mayor es la resistencia del potenciómetro de los sensores TCRT5000 menor es la corriente que circula por el fototransistor, es decir que la luz infrarroja emitida mucho menor, en contrario si el valor de la resistencia es menor circulara más corriente y se emitirá mucha luz infrarroja, pero se puede quemar si pasa más de 100mA.

La colocación de los sensores es fundamental ya que, si no están en un sitio ideal, el robot tendrá dificultad en seguir la línea.

El seguimiento de línea ha sido satisfactorio, anqué tuvimos bastantes problemas con los motores a bajas velocidades.

Se ha logrado el objetivo general de esta fase del proyecto, también logramos tener una estructura sólida y equilibrada para nuestro robo.

Ahora que tenemos nuestro seguidor de linea, que es como la puerta de entrada al mundo de robotica , procedemos a mejorarle un poco mas para que sea lo mas automata posible ,a continuacion en el la siguiente fase del proyecto haremos que nuestro robot pueda evitar obstaculos y calcular mejores rutas alternativas.

En esta primera fase del proyecto hemos obtenido importantes conclusiones, la más relevante es que para realizar un robot no se necesita grandes inversiones económicas ni tecnología punta.

Además, al trabajar con microcontroladores concluyamos que las aplicaciones más complejas son llevadas a cabo sin mayor problema, lo que demuestra que se puede aplicar el control automático a cualquier proceso o conjunto de procesos.

3. FASE-DOS ROBOT EVITA OBSTÁCULOS

En cualquier camino o recorrido, a veces es necesario evitar obstáculos, como cualquier elemento u objetivo encontrado en el recorrido de un robot. En el caso de robot industrial puede ser catastrófico no tener esta opción en los robots ya que a tamaños industriales el daño de colisión puede provocar daños graves en materiales o incluso vidas humanas.

3.1. OBJETIVO GENERAL

Nuestro objetivo principal es conocer los conceptos básicos de un robot esquiva obstáculos y crear un robot móvil capaz de detectar obstáculo, evadirlos calculando mejores rutas alternativas.
El sensor elegido ha sido el HC-SR04 que puede medir distancia detectando obstáculos en el camino y con el control de los motores DC accionamos nuestro robot para esquivar el obstáculo, avanzando, girando o retrocediendo, logrando así una conciencia espacial para nuestro robot y evitamos que se colisiona.

3.1.1. Objetivos específicos

Los sensores ultrasónicos HC-SR04 tienen un Angulo efectivo de detección de <15°, es decir su capacidad de detectar objetos es limitada por ese rango, así que debemos de buscar una solución para evitar que nuestro robot se colisiona por los principales ángulos ciegos que puedan afectar a su movilidad.

3.2. FUNDAMENTOS Y ARQUITECTURA

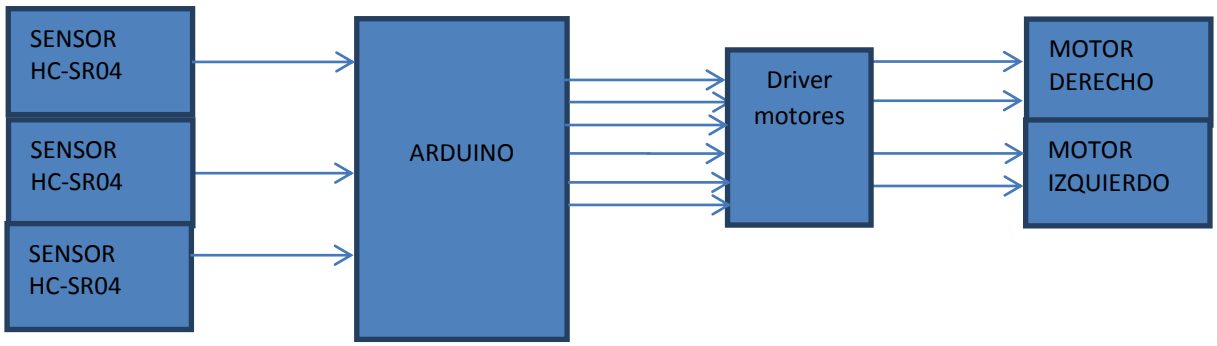


ILUSTRACIÓN 31: ARQUITECTURA ROBOT EVITA OBSTÁCULOS

3.3. MATERIALES

- ✓Chasis robot (base robot, soporte motores, rueda loca)
- ✓Placa Arduino UNO
- ✓Motores DC
- ✓ruedas
- ✓Módulo L298N
- ✓Cable macho-macho
- ✓Cable hembra-macho
- ✓Conector Jack-macho + batería 9V
- ✓soporte pilas
- ✓Protoboard-mini
- ✓Sensor HC-SR04
- ✓Soporte sensores HC-SR04
- ✓Tornillos y tuercas

3.3.1. Sensor HC-SR04

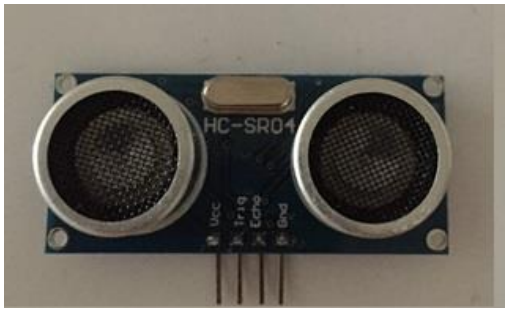


IMAGEN 67: SENSOR HC-SR04

El sensor ultrasónico HC-SR04 es un dispositivo para medir distancias en aplicaciones robóticas, envía un pulso de alta frecuencia que rebota en los objetivos cercanos y lo recibe un micrófono que dispone el sensor adecuado para esa frecuencia, así crear una imagen del entorno recorrido por los robots autómatas que facilita su desplazamiento.
A continuación, explicamos el funcionamiento y como se utiliza el sensor ultrasónico para medir distancias mediante un eco ultrasónico.

3.3.2. Funcionamiento sensor HC_SR04

Como habíamos dicho este sensor funciona como un radar ya que envía pulsos a alta frecuencia en este caso ultrasonidos que viajan el aire a una velocidad más o menos constante, así podemos estimar la distancia del objeto el cual reboto la señal con la siguiente formula:

$$V=\frac{\Delta s}{\Delta t}$$; Δs: diferencia en la posición

Δt: tiempo que tarda en desplazarse la señal

Como la trayectoria es casi recta debido que los sensores tienen limitado el Angulo efectivo de trabajo podemos utilizar la siguiente formula.

$v=\frac{d}{\Delta t}$ → d = v × t

d: distancia recorrida por la señal ultrasónica
t: tiempo transcurrido de la señal.
v: velocidad del sonido
Sabiendo que la velocidad del sonido es

$v=343^{m}/_{s}$ $=343\times\frac{100cm}{1000000us}=\frac{1cm}{29{,}2us}$ → el sonido tarda 29.2us en recorrer un centímetro.

Así que podemos obtener la distancia a partir del tiempo que se tarda en emitir y recibir la señal sonora mediante la siguiente ecuación:

$$\text{Distancia (cm)}=\frac{tiempo\,(us)\times velocidad\,^{cm}/_{us}}{2}$$

Dividimos la distancia por 2 ya que medimos lo que tarda en emitir y recibir el pulso, por lo que la distancia recorrida es el doble de la deseada.
Tiempo =2 × (distancia / velocidad)

3.3.3. Características sensor HC-SR04

El sensor ultrasónico tiene únicamente cuatro pines los hay de tres y de cinco, pero en nuestro caso tenemos:

I. VCC: alimentación 4.5 a 5.5V, suporta corriente máxima de 20mA
II. Trig: pin de disparo
III. Echo: pin de eco
IV. GND: tierra

Este sensor tiene un rango de medición de 2cm a 5m de distancia y un ángulo efectivo de 15°, la señal de disparo es de 10us como demuestra la figura a continuación extraída del "datasheet" del sensor.

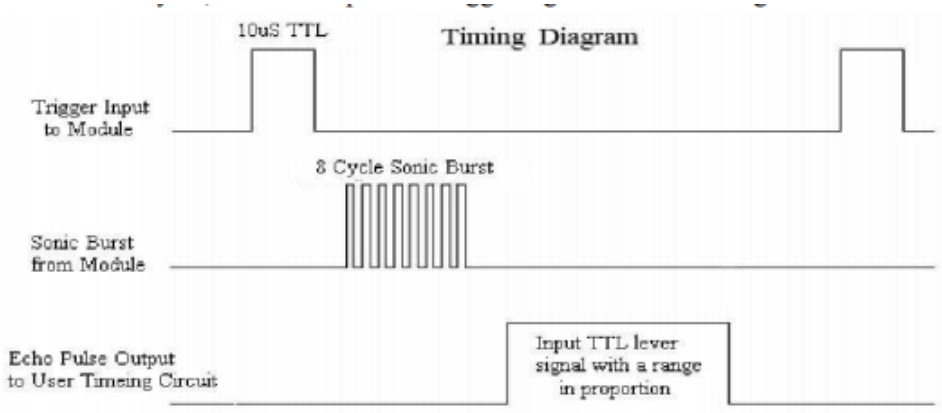


ILUSTRACIÓN 32: CANALES SENSOR ULTRASÓNICO," DATASHEET HC-SR04"

Como podemos observar en la “Ilustracion32” tenemos tres canales en ese orden:

I. Canal de disparo donde la señal será emitida
II. Canal donde se emite la señal desde el modulo
III. Canal del eco

Funciona de la siguiente manera: una vez se envía el pulso de 10us el sensor comenzara a trabajar enviando ocho pulsos ultrasónicos como demuestra la figura, una vez enviados los ocho pulsos, el pin eco enviara la distancia del objeto en forma de pulsos que van desde 150us hasta 38ms, es decir dependiendo del ancho de pulso se detendrá distintos valores de la distancia.

3.3.4. Angulo efectivo HC-SR04

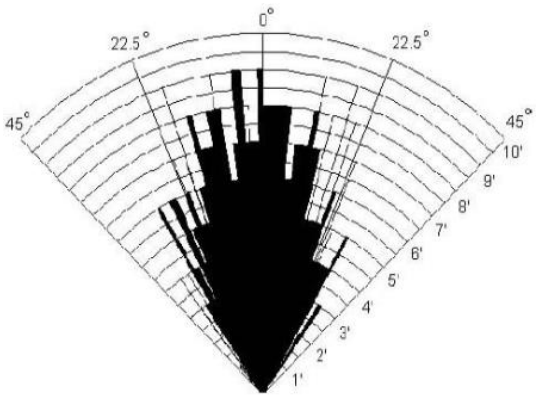


ILUSTRACIÓN 33: ANGULO EFECTIVO HC-SR04

En la figura de arriba podemos ver la distribución de dB en función de un punto de referencia a lo largo de un radio que no cambia, así podemos saber a qué distancia se puede detectar un objeto en función del ángulo.
Podemos observar en la “ilustración33” que el ángulo efectivo del sensor es de 15 grados a la derecha e izquierda del ángulo 0.

3.4. CONFIGURACIÓN ARDUINO CON SENSOR UTLTRASONIDO

Como vamos a utilizar tres sensores y consideramos el objetivo final de juntar las 3 fases del proyecto, he decido conectar los sensores ultrasónicos a los pines analógicos A0…A5 ya que nuestra tarjeta Arduino no tiene más salidas digitales y nos permite usar la salida analógica como digitales, a continuación, haremos un pequeño montaje, para asegurar el buen funcionamiento del sensor y después procedemos con el diseño del robot evita obstáculos.

3.4.1. Materiales prueba

- ✓ Placa Arduino UNO
- ✓ Cables macho-hembra
- ✓ Cables macho-macho
- ✓ Protoboard-Mini
- ✓ Led
- ✓ Resistencia

3.4.2. Software prueba

- ✓ IDE Arduino
- ✓ Fritzing

3.4.3. Conexiones Arduino Uno, con sensor HC-SR04

Sensor HC_04	Arduino UNO	Descripción
Vcc	**5 V**	Alimentación salida 5v Arduino
Trig	**A0**	Pin de disparo
Echo	**A1**	Pin del eco
Gnd	**Gnd**	Tierra, Ground

TABLA 11: CONEXIONES ARDUINO HC-SR04

3.4.4. Esquema eléctrico de prueba

La siguiente figura muestra cómo se conecta el sensor ultrasónico HC-SR04 con los pines analógicos de la tarjeta Arduino Uno, además ponemos un led que nos avisa de la proximidad del obstáculo.

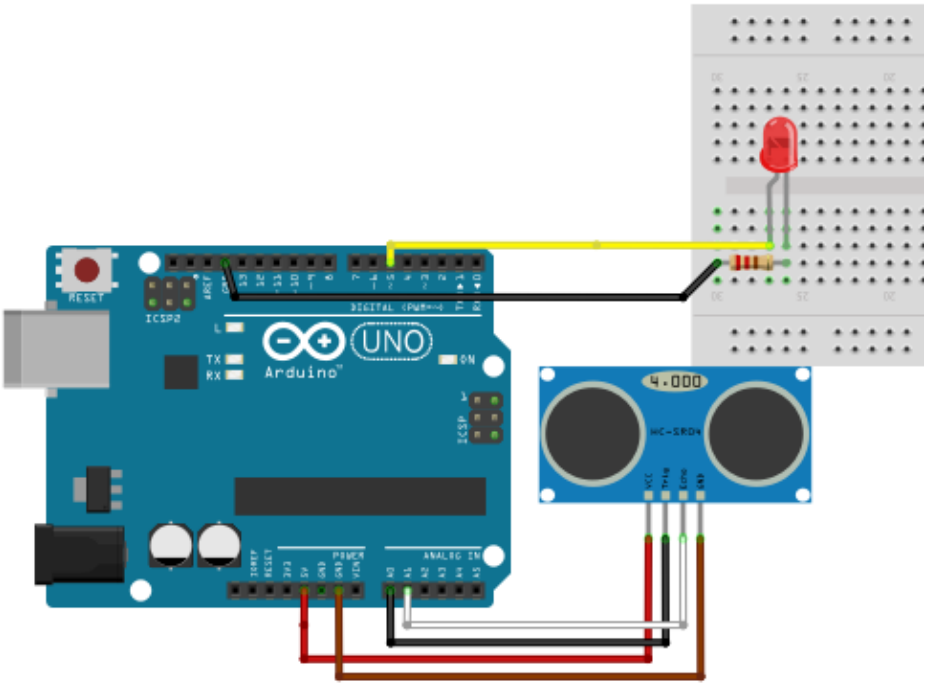


FIGURA 18: MONTAJE ARDUINO SENSOR ULTRASÓNICO, FRITZING

Con el esquema de la "figura 18" pretendemos que el sensor HC-SR04 empiece a detectar la distancia del objeto que tiene enfrente, si está muy cerca o mejor dicho en la distancia requerida se encenderá el led avisando de estamos a una concreta distancia del obstáculo.
El pin A0 se declaró como salida ya que generara la medición que se vea reflejada en cm atreves del monito serial con el comando Serial.println(distancia), la patita ECHO conectado al pin A1 de Arduino UNO es el que recibe la señal reflejada.

3.4.5. Codigo8_ensayo sensor ultrasónico

```
const int Trig = A0;//pin disparo
const int Echo = A1;// pin eco
const int Led=5;// led conectado al pin digital 5, se enciende si la distancia está menos de 10cm

long Distancia; // variable donde se almacena valor distancia
long Dis;// tiempo que tarda en reflejar la señal


//A continuación creamos una función cuyo objetivo es calcular la distancia y grabarla en la variable distancia.
void ultrasonido () {

  // pin Trig se establece a LOW
  digitalWrite (Trig, LOW);
  delay(10);

  //lazamos los 8 pulsos
  digitalWrite (Trig, HIGH);
  delay(10);
  digitalWrite (Trig, LOW);

// Midiendo el tiempo que tarda la señal en regresar podemos calcular la distancia.
  Dis = pulseIn(Echo, HIGH);// al utilizar la función pulseIn() obtenemos el valor de tiempo que tarda la señal, no la distancia.
  Distancia = Dis / 58; // calculamos la distancia mediante esta fórmula demostrada anteriormente
  delay(100);
}
void setup() {
   Serial.begin(9600); // iniciamos comunicación puerto serial
   pinMode (Trig, OUTPUT); // envió de señal como salida
  pinMode (Echo, INPUT);// recibir señal como entrada
   pinMode (Led,OUTPUT);// pin 5 como salida para encender al led si la distancia <10cm
}
void loop() {

  ultrasonido (); // llamamos a la función que nos devuelva el valor de la distancia
  Serial.println(Distancia);// visualizar valor distancia a través del monitor serial

   if(Distancia < 10){ // si la distancia <10cm se enciende el led
    digitalWrite(Led,HIGH);
     }

else
 {digitalWrite(Led,LOW); // si la distancia>10cm vuelve a apagar el led
 }

  delay (100);
}
```

3.4.6. Simulaciones

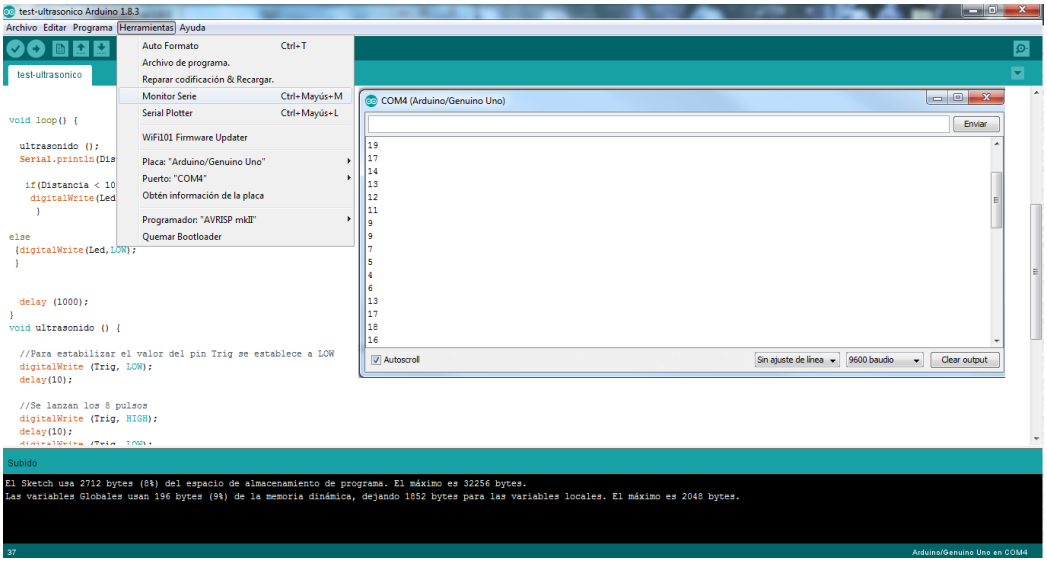


ILUSTRACIÓN 34: MEDIR DISTANCIA HC-SR04

Una vez compilamos el programa y lo cargamos en nuestra placa Arduino podemos visualizar las distancias a través del monitor serie como muestra la foto anterior.

3.4.7. Simulación real

Primero ponemos un obstáculo a 18 cm, visualizamos la distancia en el monitor serie y vamos acercando el objetivo al sensor ultrasónico, una vez llegamos a la distancia de 10 cm, se enciende el led como lo tenemos programado.

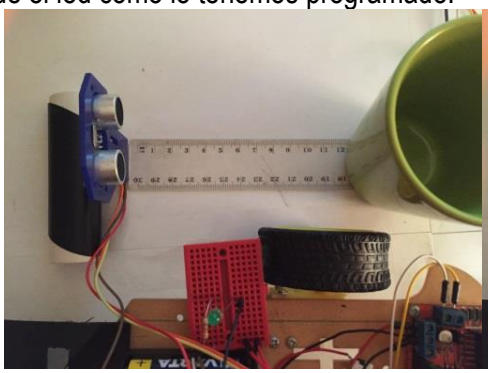


IMAGEN 68: PRUEBA DISTANCIA >10 CM

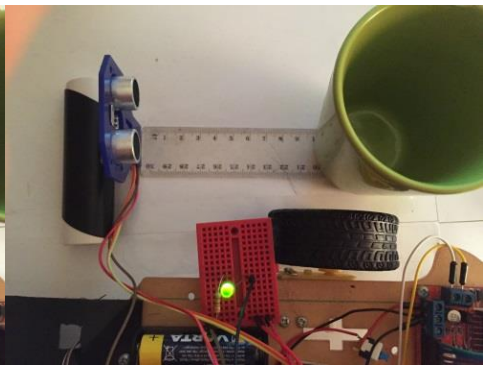


IMAGEN 69: PRUEBA DISTANCIA <=10CM

A continuación, vamos alejando el objetivo mientras el led sigue encendido, pero una vez nos alejamos más de 10cm el led se apaga.

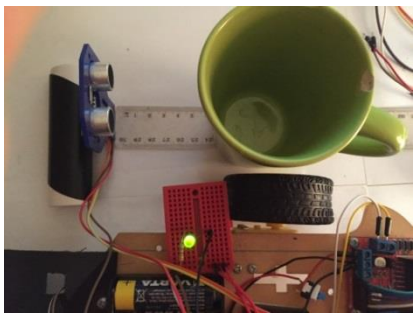


IMAGEN 70: PRUEBA DISTANCIA <10CM

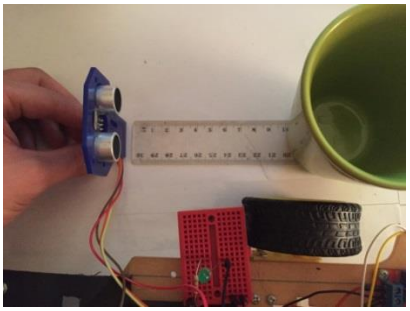


IMAGEN 71: PRUEBA DISTANCIA >10CM

3.5. DISEÑO, REALIZACIÓN Y CONFIGURACIÓN DEL PROTOTIPO

Sabiendo que el sensor tiene un ángulo efectivo muy limitado de 15°, la idea es montar tres sensores, de manera que nuestro robot tendrá mucho más ángulo de trabajo y así puede tener mejor imagen del entorno que le rodea, así facilita su desplazamiento

Para montar los tres sensores en el robot haremos uso de unos soportes para mantener su estabilidad y poder enfocarlos correctamente.

IMAGEN 72: SENSORES HC-04 ACOPLADOS A LOS SOPORTES

3.5.1. Conexión Arduino sensores

La idea es que cada vez que nuestro robot encuentra un obstáculo en su recorrido debe cambiar dirección, por lo cual creamos un sistema que el robot no solo pueda moverse detectando obstáculos sino también comprobar la dirección de giro alternativa libre de obstáculos, así elegirá un mejor recorrido, depende de la colocación de los sensores HC-SR04 y el desarrollo del código que veamos a continuación.

3.5.2. Pines de conexión Robot evita obstáculos

Sensores HC-SR4		Arduino UNO	Descripción
Sensor Izq	Vcc	5V	Alimentación 5V salida Arduino uno
	Trig	A0	Pin de disparo izquierdo
	Echo	A1	Pin de eco izquierdo
	Gnd	Gnd	Tierra común
Sensor Cen	Vcc	5V	Alimentación 5V salida Arduino uno
	Trig	A2	Pin de disparo central
	Echo	A3	Pin de eco central
	Gnd	Gnd	Tierra común
Sensor Der	Vcc	5V	Alimentación 5V salida Arduino uno
	Trig	A4	Pin de disparo derecho
	Echo	A5	Pin de eco derecho
	Gnd	Gnd	Tierra común

TABLA 12: CONEXIÓN ARDUINO SENSORES ULTRASÓNICOS

NOTA : se mantienen las mismas conexiones de la "tabla 8",para conectar el driver de potencia con la placa Arduino y los motores.

3.5.3. Esquema robot evita obstáculos

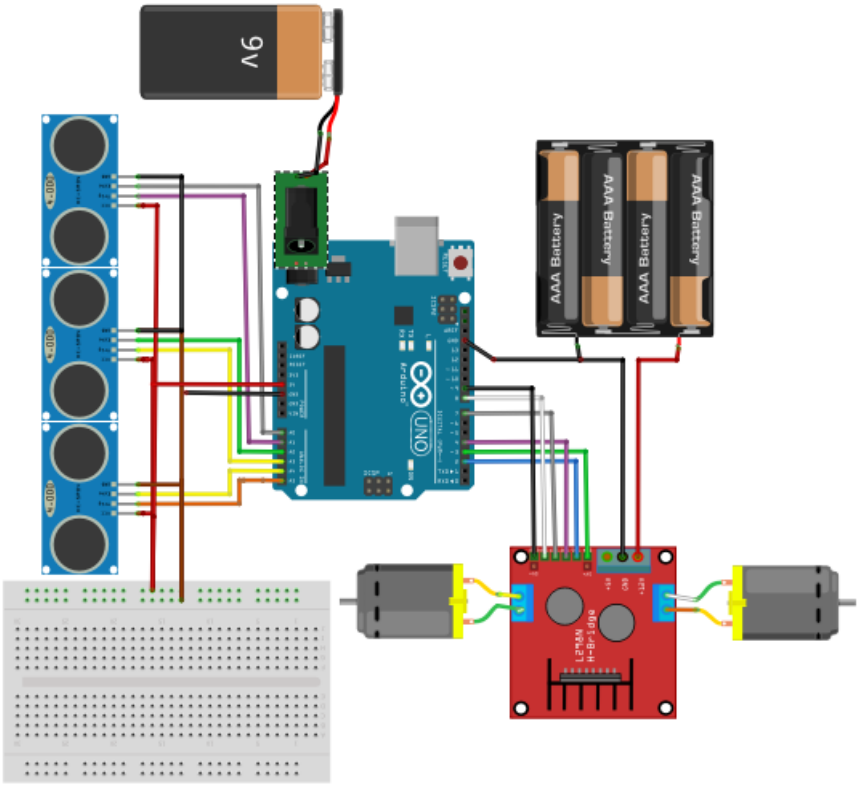


FIGURA 19: CIRCUITO ROBOT EVITA OBSTÁCULOS

3.5.4. Entorno de programación Arduino

Este proyecto puede ser mejorado por otros alumnos o aficionados a la robótica, así que intento simplificar más los esquemas y dejar disponibles las E/S digitales PWM.
Así que conectamos los pines Triger y Echo de los sensores, a través de E/S analógicas ya que se pueden usar como digitales, lo que nos simplifica la lectura de los sensores.

3.5.5. diagrama de bloques

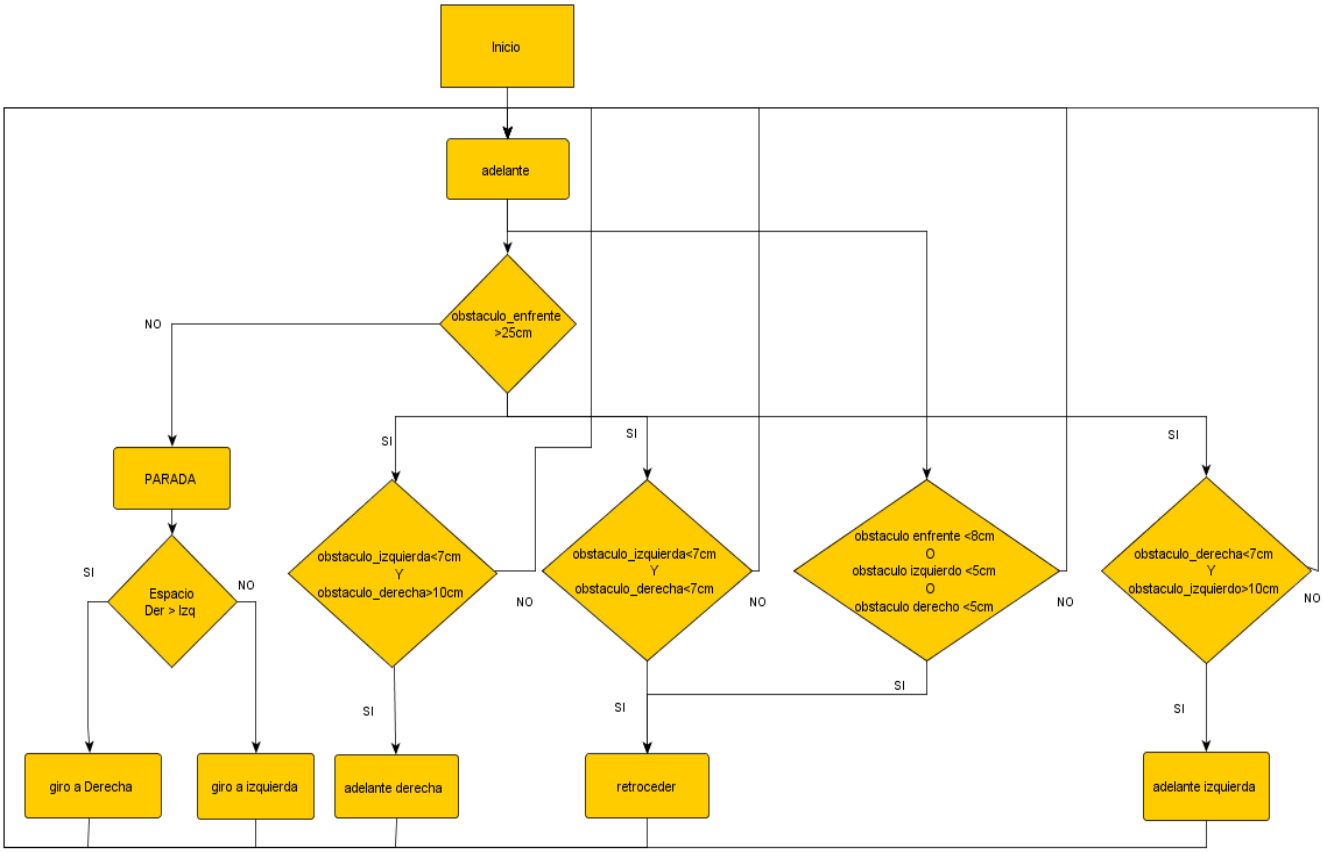


DIAGRAMA DE BLOQUES 2: DIAGRAMA ROBOT EVITA OBSTÁCULOS

3.5.6. desarrollo código evita obstáculos

Como resume el diagrama de bloques de figura anterior, empezamos como siempre nuestro código declarando las variables de control de motor y la variable del sensor HC-04 que nos permitan calcular la distancia entre nuestro Robot y los objetos a su alrededor.
Después hacemos dos pequeños algoritmos que facilitaran al Robot alejarse de los obstáculos y posible colisión.
Nuestro Robot siempre avanzara en línea recta sino detecta obstáculo, solo en el caso de que se acerca demasiado a un muro u obstáculo por la derecha, el Robot corregirá su sentido suministrando más velocidad al motor derecho que el izquierdo hasta que esté lejos del obstáculo, lo mismo por el lado izquierdo más velocidad al lado izquierdo hasta que se corrija el recorrido.
Finalmente, si llega a un camino cerrado, hará marcha atrás y buscará nuevo camino.
A continuación, creamos un pequeño código donde utilizamos funciones para el control de sentido del Robot y también para medir distancia.

3.5.7. Codigo9_Evita obstáculos

```
/ Definición de variables control motor derecho
const int derA = 2;  // Pin digital 11 para controlar sentido giro motor izquierdo
const int derB= 4;  // Pin digital 10 para controlar sentido giro motor izquierdo
const int velocidad_M1=3;
// Definición de variables control motor izquierdo
const int izqA = 7;  // Pin digital 13 para controlar sentido giro motor izquierdo
const int izqB = 8;  // Pin digital 12 para controlar sentido giro motor izquierdo
const int velocidad_M2= 9;

   int trigger_der = A0;// impulso enviado sensor derecho
    int echo_der = A1;

    int trigger_cen = A2; // impulso sensor central
    int echo_cen = A3;

    int trigger_izq = A4;// impulso sensor izquierdo
    int echo_izq = A5;//

long tiempo1,tiempo2,tiempo3;  // Almacena el tiempo de respuesta de los sensores
float  distancia_izq,distancia_cen,distancia_der;  // Almacena la distancia en cm a la que se encuentra el obstaculo
int vel=110;// velocidad motores
int vel1=135;
// Función que se ejecuta una sola vez al cargar el programa
void setup()
{
  // Se declaran todos los pines como salidas
  pinMode (derA, OUTPUT);// rueda derecha sentido horario
  pinMode (derB, OUTPUT);// rueda derecha sentido antihorario
  pinMode (izqA, OUTPUT);// rueda izquierda sentido horario
  pinMode (izqB, OUTPUT);// rueda izquierda sentido antihorario
  pinMode(velocidad_M1,OUTPUT);// velocidad motor-derecha
  pinMode(velocidad_M2,OUTPUT);// velocidad motor-izquierda

       pinMode(trigger_cen, OUTPUT);
      pinMode(echo_cen, INPUT);

       pinMode(trigger_izq, OUTPUT);
       pinMode(echo_izq, INPUT);

       pinMode(trigger_der, OUTPUT);
       pinMode(echo_der, INPUT);
  Serial.begin(9600); // Inicio puerto de comunicaciones en serie
}
```

```
void loop()
{
  sensordis_cen(); // función detecta obstáculo enfrente
  sensordis_izq(); //función detecta obstáculo a la izquierda
  sensordis_der(); //función detecta obstáculo a la derecha
  Adelante();
  if(distancia_cen<25)
  {
 Stop();
   if(distancia_izq<distancia_der)
 {
 Derecha();
  }
  else {
    Izquierda();
         }
  }
    if((distancia_izq<7)&& (distancia_cen<25)&& (distancia_der<7))
 {
  Stop();
Retroceder();
  }
if((distancia_izq<5)||(distancia_cen<8)||(distancia_der<5))
{
Retroceder();
}
 if((distancia_izq>10)&& (distancia_cen>25)&& (distancia_der<7))
  {
    // Motor izquierdo
  digitalWrite (derA, HIGH);// rueda derecha sentido horario
  digitalWrite (derB, LOW);// rueda derecha sentido antihorario
  // Motor derecho
  digitalWrite (izqA, HIGH);// rueda izquierda sentido horario
  digitalWrite (izqB, LOW);// rueda izquierda sentido antihorario.
    analogWrite (velocidad_M1,vel1);// Robot corrige su recorrido a la izquierda, vel1> vel.
    analogWrite (velocidad_M2,vel);// velocidad motor-izquierda
  }
    if((distancia_izq<7)&& (distancia_cen>25)&& (distancia_der>10))
  {
    // Motor izquierdo
 digitalWrite (derA, HIGH);
  digitalWrite (derB, LOW);
  // Motor derecho
    digitalWrite (izqA, HIGH);
  digitalWrite (izqB, LOW);
    analogWrite (velocidad_M1,vel);// velocidad motor-derecha
    analogWrite (velocidad_M2,vel1);//Robot corrige su recorrido a la derecha, vel1> vel.
 }
  }
```

```
// Función sensordis : nos mide el intervalo de tiempo transcurrido entre el envío del pulso
//ultrasónico hasta que el sensor recibe el rebote, así logramos medir la distancia de objetos que
//rodean nuestro robot

void sensordis_cen()
{
   // Se inicializa el sensor
   digitalWrite(trigger_cen,LOW);  // Para estabilizar
   delayMicroseconds(10);
   // enviamos una señal activando la salida trigger durante 10 microsegundos
   digitalWrite(trigger_cen, HIGH);  // envío del pulso ultrasónico
   delayMicroseconds(10);
   tiempo1=pulseIn(echo_cen, HIGH); //  tiempo que tarda la señal
   distancia_cen= int(0.017*tiempo1); // Fórmula para calcular la distancia en cm
   Serial.println("El valor de la distancia_ce es ");
   Serial.println(distancia_cen);
   delay(100);
   }
    void sensordis_izq()
{
     // Se inicializa el sensor
   digitalWrite(trigger_izq,LOW);  // Para estabilizar
   delayMicroseconds(10);
  // Enviamos una señal activando la salida trigger durante 10 microsegundos
   digitalWrite(trigger_izq, HIGH);  // envío del pulso ultrasónico
   delayMicroseconds(10);
   tiempo2=pulseIn(echo_izq, HIGH); //  tiempo que tarda la señal
   distancia_izq= int(0.017*tiempo2); // Fórmula para calcular la distancia en cm
   Serial.println("El valor de la distancia_izq es ");
   Serial.println(distancia_izq);
   delay(100);
    }
void sensordis_der()
{
     // Se inicializa el sensor
   digitalWrite(trigger_der,LOW);  // Para estabilizar
   delayMicroseconds(10);
   // Se envía una señal activando la salida trigger durante 10 microsegundos
   digitalWrite(trigger_der HIGH);
   delay (10);
   tiempo3=pulseIn(echo_der, HIGH);
   distancia_der= int(0.017*tiempo3); // calculo la distancia en cm
   Serial.println("El valor de la distancia_der es ");
   Serial.println(distancia_der);
   delay(100);
}
```

```
// Función Adelante: esta función hará que ambos motores se activen a máxima potencia
  por lo que el robot avanzará hacia delante
void Adelante()
{
  // Motor izquierdo
  digitalWrite (derA, HIGH);//rueda derecha sentido horario
  digitalWrite (derB, LOW);//
  // Motor derecho
  digitalWrite (izqA, HIGH);// rueda izquierda sentido horario
  digitalWrite (izqB, LOW);//
  analogWrite (velocidad_M1,vel);// velocidad motor-derecha
  analogWrite (velocidad_M2,vel);// velocidad motor-izquierdo
}
//Función Retroceder: esta función hará que ambos motores se activen a máxima potencia en
sentido contrario al anterior por lo que el robot avanzará hacia atrás.
void Retroceder()
{
  // Motor derecho
  digitalWrite (derA, LOW);
  digitalWrite (derB, HIGH);// rueda derecha sentido antihorario
  // Motor izquiero
  digitalWrite (izqA, LOW);
  digitalWrite (izqB, HIGH);// rueda izquierda sentido antihorario
  analogWrite (velocidad_M1,vel);// velocidad motor-derecha
  analogWrite (velocidad_M2,vel);// velocidad motor-izquierdo
}
 // Función Izquierda: esta función accionará el motor derecho y parará el izquierdo
  por lo que el robot girará hacia la izquierda
void Izquierda()
{
  digitalWrite (derA, HIGH);// rueda derecha sentido horario
  digitalWrite (derB LOW,);
  digitalWrite (izqA, LOW);
  digitalWrite (izqB, LOW);
  analogWrite (velocidad_M1,vel);// velocidad motor-derecha
  analogWrite (velocidad_M2,vel);// velocidad motor-izquierdo
}
 // Función Derecha: esta función accionará el motor izquierdo y parará el derecho, por lo que el
robot girará hacia la derecha
void Derecha ()
{
//Motor izquierdo
  digitalWrite (derA, LOW);
 digitalWrite (derB, LOW);
  // Motor derecho
  digitalWrite (izqA, HIGH);//rueda izquierda sentido horario
  digitalWrite (izqB, LOW);
  analogWrite (velocidad_M1,vel);// velocidad motor-derecha
  analogWrite (velocidad_M2,vel);// velocidad motor-izquierdo
}
```

```
// Función Stop: esta función parará ambos motores por lo que el robot se queda inmovilizado.
void Stop()
{
  // Se para el motor izquierdo
  digitalWrite (derA, LOW);
  digitalWrite (derB, LOW);
 // Se para el motor derecho
  digitalWrite (izqA, LOW);
  digitalWrite (izqB, LOW);
}
```

3.6. Resumen

La parte esencial de esta fase del proyecto son los sensores, por lo que hay que tener en cuenta los rangos deseados para la detección de obstáculos, por lo que hay que calibrar las distancias de los tres sensores, ya que depende del ángulo en el que están colocados como muestra la "imagen 73".

Se recomienda trabajar en rangos de distancia alejados del punto crítico del sensor para evitar problemas de detección.

También se recomienda regular la velocidad del robot dependiendo de los sensores conectados para evitar maniobras bruscas entrando a veces en la zona muerta de los sensores

Fue necesario utilizar tres sensores ultrasónicos para garantizar el buen funcionamiento del robot, debido a que el sensor HC-SR04 tiene un Angulo efectico de 15 grados lo que le deja un rango de detección de 30 grados y de utilizar un solo sensor quedarían puntos ciegos esenciales en el recorrido del robot que podrían ocasionar colisiones y daño graves a nuestro robot.

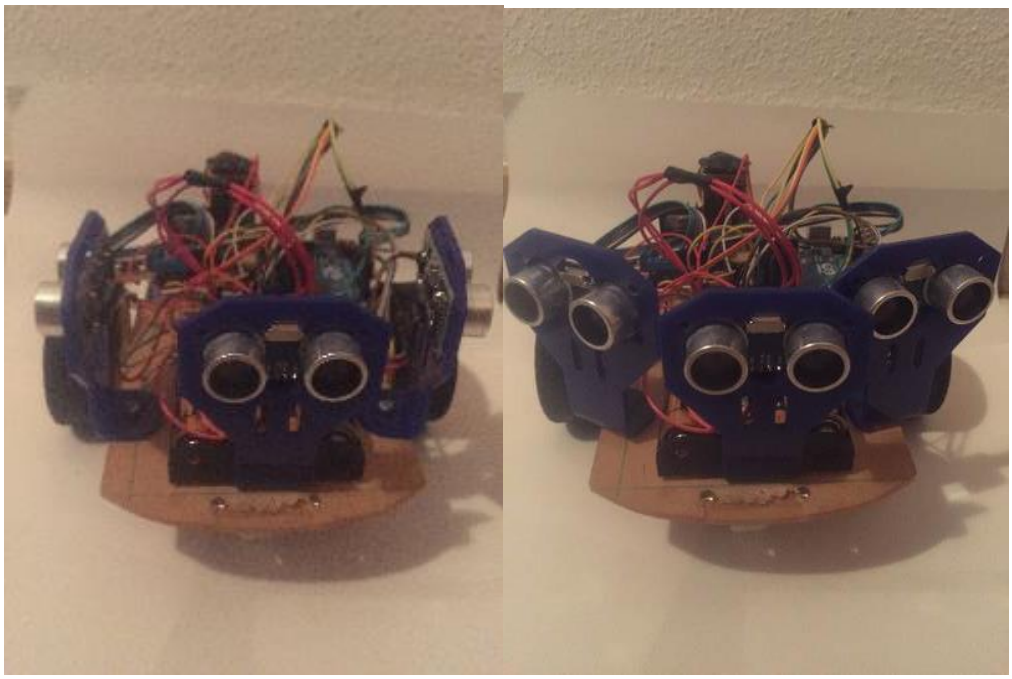


IMAGEN 73: ROBOT EVITA OBSTÁCULOS

4. FASE-TRES CONTROL ROBOT BLUETOOTH

El control remoto manual de un robot es un elemento esencial en el desarrollo de robots multifunciones.
Esta fase de proyecto consiste en dos partes un robot y un control remoto, como ya tenemos echa la primera parte pasamos directamente al control remoto, que consiste en controlar el modo de funcionamiento del robot y sus movimientos, también sirve para estimulo mental e introducirlo al estudio formal de la robótica.
Para entender el funcionamiento de un robot por control remoto es necesario conocer las partes que lo forman y su forma de actuar.

4.1. OBJETIVO PRINCIPAL

Nuestro objetivo principal es diseñar un sistema robótico controlado mediante un componente electrónico que es Arduino, atreves de la tecnología de comunicación (APP-Android→Arduino uno) inalámbrica conocida como Bluetooth que sirve de control remoto de un robot a distancias de menos 10m.

4.1.1. objetivos específicos

- ✓ Adquirir nuevos conocimientos de comunicación
- ✓ Crear circuitería necesaria para la comunicación, hacerla funcionar en el Protoboard
- ✓ Inventar nuevas cosas que funcionen con tecnología de computación.
- ✓ Crear aplicación propia con todos los comandos del robot.

4.2. FUNDAMENTO Y ARQUITECTURA

La idea es mandar diversas órdenes del móvil con sistema operativo Android, que serán recibidas por el módulo Bluetooth HC-06 que a su vez se la pasa a la tarjeta Arduino UNO para ejecutar cierta orden, la arquitectura del sistema robótico esta simplificada en la siguiente figura.

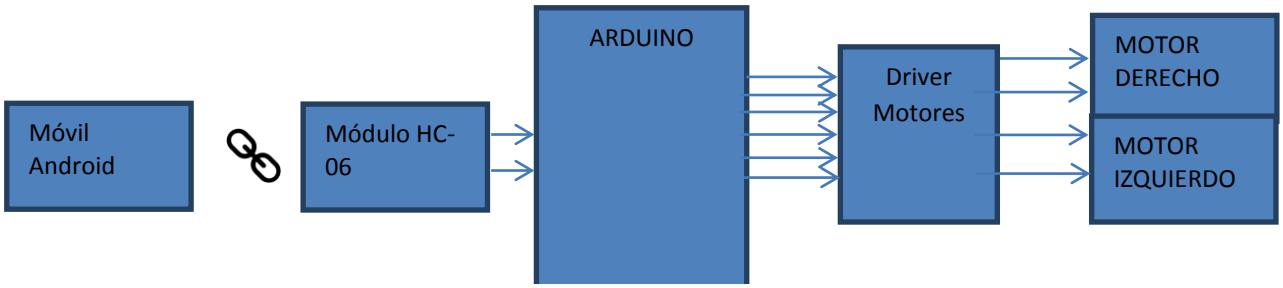


ILUSTRACIÓN 35: ARQUITECTURA CONTROL REMOTO ROBOT

4.3. MATERIALES

- ✓ Chasis robot (base robot, soporte motores, rueda loca)
- ✓ Placa Arduino UNO
- ✓ Motores DC
- ✓ ruedas
- ✓ Módulo L298N
- ✓ Cable macho-maho
- ✓ Cable hembra-macho
- ✓ Conector Jack-macho + bateria 9V
- ✓ soporte pilas
- ✓ 4 baterias 1.5V
- ✓ Protoboard-mini
- ✓ Sensor HC-06
- ✓ Móvil con sistema Android

Software:

- ✓ IDE Arduino
- ✓ Fritzing
- ✓ App inventor2

4.3.1. SENSOR HC-06

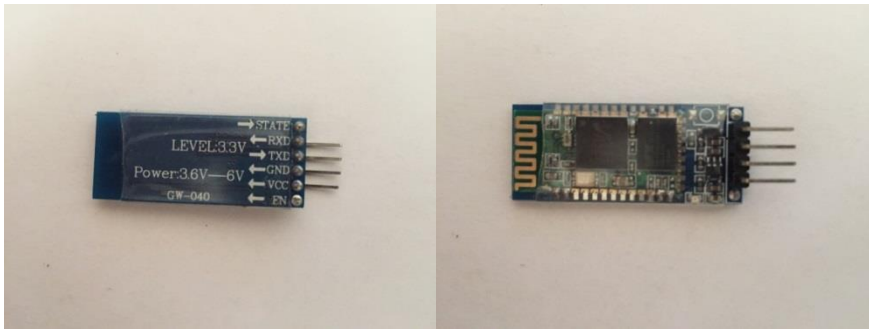


IMAGEN 73: SENSOR HC-06

Modulo Bluetooth HC-06 solo opera en modo esclavo tiene 2 pines menos que el modelo HC-05 y dispone de un juego reducido de instrucciones a las que atiende en comparación con el HC-05, así que es un módulo sencillo e ideal para recibir comandos enviados desde el sistema operativo Android de un móvil o Tablet hacia Arduino.

4.4. CONFIGURACIÓN MODULO BLUETOOTH HC-06

La ventaja del módulo HC-06 es que viene configurado de fábrica como esclavo y no se puede cambiar, pero si podemos cambiar otras características usando comandos AT como, por ejemplo:
Código de emparejamiento, velocidad, nombre.

4.4.1. Funcionamiento módulo Bluetooth HC-06

Este módulo funciona en dos modos:

I. Modo desconectado:

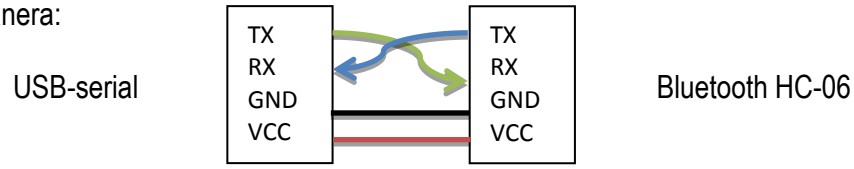
Una vez alimentado el módulo entra enseguida en este modo y siempre cuando no se ha estabilizado ninguna conexión Bluetooth con otro dispositivo.
En este modo el led del módulo debe de estar parpadeando y es cuando se puede configurar el módulo enviando comandos AT.

II. Modo conectado:

Una vez establecida una conexión bluetooth con otro dispositivo entra en modo conectado.
En este modo el led permanece encendido, el módulo no puede interpretar comandos AT en este modo y todos los datos recibidos por el Pin RX del módulo HC-06 se trasmiten al dispositivo conectado, así lo datos recibidos se devuelven por el Pin TX de Arduino y se ejecuta la orden.

4.4.2. Configuración módulo Bluetooth

La conexión de Arduino y el módulo Bluetooth será a través del os pines TX y RX.
Permite conexión sencilla mediante comandos AT a través de una puerta serie de la siguiente manera:



4.4.3. conexiones Arduino HC-06

Tenemos que tener mucho cuidado de no permutar los pines de alimentación Vcc y Gnd, ya que al hacerlo el módulo se averiara inmediatamente.

Modulo Bluetooth HC-06	Arduino UNO	Descripción
RXD	D1(TXD)	Recepción←trasmisión
TXD	D0 (RX)	Transmisión →recepción
GND	GND	Tierra comun
VCC	3.3V	Alimentación modulo salida 3.3V Arduino

TABLA 13: CONEXIÓN ARDUINO CON MODULO BLUETOOTH HC-06

4.4.4. Esquema Arduino_HC-06

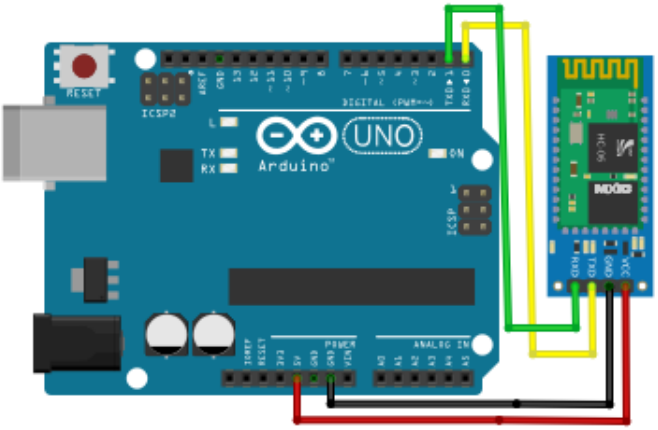


FIGURA 20: MONTAJE ARDUINO, BLUETOOTH

Una vez tenemos las conexiones cargamos un pequeño programa en el cual podemos cambiar alguna configuración del módulo Bluetooth y visualizar los cambios a través del monito Serial de nuestro IDE.
Creamos un pequeño programa a continuación que nos permita comunicar Arduino con el Bluetooth HC-06.

4.4.5. Codigo10_Confguracion módulo HC-06

```
#include <SoftwareSerial.h>   // Incluimos la librería SoftwareSerial
SoftwareSerial BT(6,5);    // Definimos los pines RX (TX del HC-06) y TX (RX del HC-06) del Arduino conectados al Bluetooth

void setup()
{
  Serial.begin(9600); // inicializamos comunicación puerto serie
  BT.begin(9600);       // Inicializamos el puerto serie BT que hemos creado

}

void loop()
{
  if(BT.available())    // Si llega una señal por el puerto BT se envía al monitor serial
  {
    Serial.write(BT.read());

  }

  if(Serial.available())  // Si llega una señal por el monitor serial se envía al puerto BT
  {
     BT.write(Serial.read());

  }
}
```

4.4.6. Comandos AT

Los comandos **Hayes,** conocidos como comandos **AT** es un lenguaje desarrollado por la compañía "Hayes Communications" que prácticamente se convirtió en estándar abierto de comandos para configurar y parametrizar módems.[13]
En nuestro caso utilizaremos un conversor USB serial PL2303 que se ha instalado como puerto serial COM4, así que debemos de seleccionar dicho puerto antes de abrir el monitor serial.

Nuestro modulo Bluetooth debe estar en Modo AT, que se puede verificar viendo el LED del HC-06 parpadeando, esto significa que no hay conexión Bluetooth establecida con otro dispositivo.

Una vez hechas las conexiones correspondientes, abrimos el Monitor serial del IDE de Arduino.

Después, en la parte inferior del monitor serial debemos seleccionar (No hay fin de línea) y la velocidad (9600 baud) ya que es la velocidad por defecto del módulo HC-06.

A continuación, comprobamos si nuestro Bluetooth responde a los comandos AT
Enviar: AT → Recibes: OK
Si recibimos un OK entonces procedamos a cambiarle el nombre con el comando:
Enviar: AT+NAME → Recibes: OKsetname

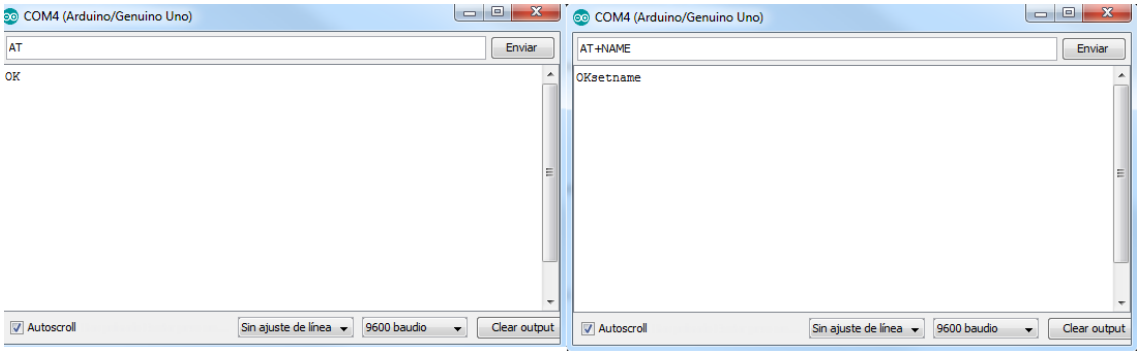


ILUSTRACIÓN 36: PROBANDO COMANDOS AT

Ahora que hemos comprobado la comunicación cambiamos el nombre del HC-06 a "ROBOT_MULTIFUNCION" se puede poner hasta 20 caracteres, y también su código ya que por defecto es el 1234.
Cambio Nombre: Enviar: AT+NAME<ROBOT MULTIFUNCCION> → Recibes: OKsetname
Cambio Pin: Enviar: AT+PIN7777 → Recibes: OKsetPIN

[13] https://es.wikipedia.org/wiki/Conjunto_de_comandos_Hayes

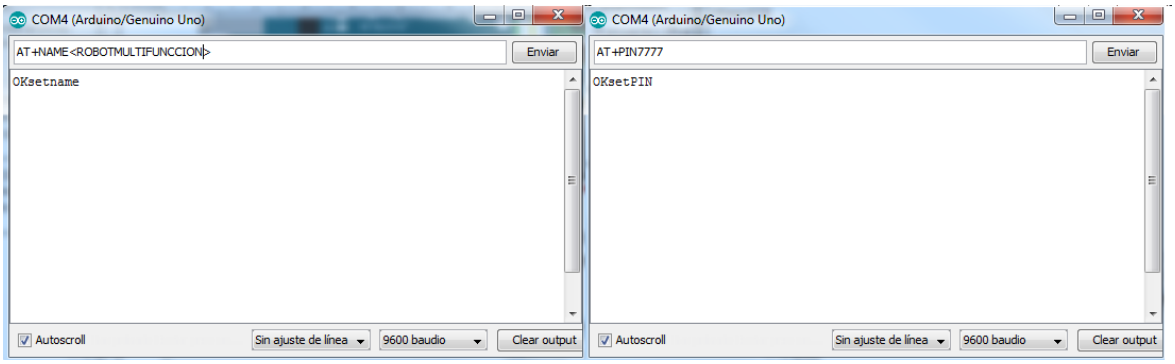


ILUSTRACIÓN 37: CAMBIO NOMBRE Y PIN BLUETOOTH HC-06

Una vez comprobada la comunicación del módulo Bluetooth, procedamos a conectar el driver de potencia l298N y los motores a la tarjeta Arduino UNO. Con las mismas conexiones de la "tabla 8".

4.5. Configuración control Robot

En el apartado anterior hemos visto como configurar un módulo Bluetooth y conectarlo a la placa Arduino, ahora diseñamos un esquema del conjunto robot + modulo Bluetooth HC-06 que sirva para el control remoto a través de la aplicación que crearemos una vez tengamos el esquema electico.

4.5.1. Conexiones Arduino modulo Bluetooth HC-06

Modulo Bluetooth HC-06	Arduino UNO	Descripción
RXD	**D6**	Recepción←trasmisión
TXD	**D5**	Transmisión →recepción
GND	**GND**	Tierra común
VCC	**3.3V**	Alimentación modulo salida 3.3V Arduino

TABLA 14: CONEXIONES ARDUINO CON HC-06

Conexión Arduino – driver –motores siempre la misma que la "tabla8".

4.5.2. Esquema eléctrico control remoto Arduino

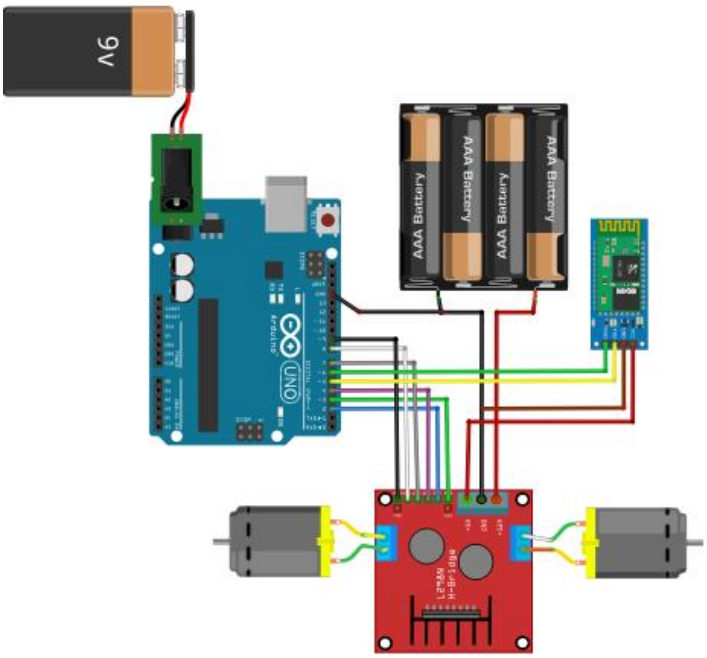


FIGURA 21: MONTAJE CONTROL ROBOT BLUETOOTH

Ahora que tenemos el esquema eléctrico, empezaremos con la programación de la aplicación móvil mediante la cual controlaremos los movimientos del robot enviando ordenes vía Bluetooth, una vez tengamos la aplicación echa acabaremos programando la placa Arduino para que reciba y ejecuta la orden enviada correctamente.

4.6. EMPAREJAR MÓVIL CON MODULO BLUETOOTH HC-06

Para hacer esta tarea solo necesitamos, un móvil Android y un módulo Bluetooth conectado a la tarjeta Arduino.
Una vez tenemos el esquema montado, alimentamos el módulo Bluetooth y nos aseguramos de que está en modo AT y el led parpadeando para verificarlo, después accederemos a la configuración del dispositivo Android.
Encendemos el teléfono móvil y activamos la conexión Bluetooth, nos saldrá una pantalla que indica los dispositivos disponibles en nuestra proximidad como demuestra la “ilustración 38”.

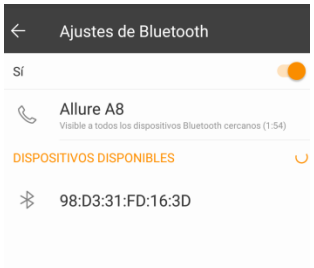


ILUSTRACIÓN 38: BUSCAR MÓDULO HC-04

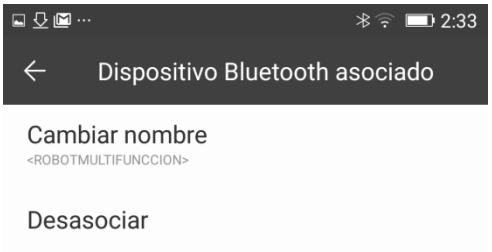


ILUSTRACIÓN 39: EMPAREJAR ANDROID - HC-04

Nuestro dispositivo detecta esta dirección MAC que al pulsar nos sale en nombre ROBOTMULTIFUNCCION como demuestra la “ilustración a 39” que pusimos a nuestro módulo HC-06, pulsamos vincular e introducimos el pin cambiado anteriormente→ PIN: 7777.
.

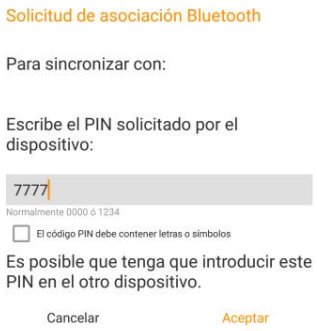


ILUSTRACIÓN 40 : CÓDIGO SINCRONIZACIÓN BLUETOOTH

Una vez pulsamos aceptar, tendremos al Bluetooth del móvil y Arduino sincronizados, acto seguido podemos crear nuestra aplicación en APPinventor2.

4.7. CREAR APLICACIÓN DE CONTROL REMOTO PARA ANDROID

Existen diversas aplicaciones de control remoto en Android que nos puedan sirven igual, pero son limitadas a la hora de enviar ordenes, así que crearemos una aplicación que se ajusta a nuestro proyecto.
El mando de control constara de aplicación que crearemos a continuación para poder enviar órdenes pulsando botones.

4.7.1. App inventor 2

Es una aplicación web diseñada para crear aplicaciones de móviles, es una herramienta de software libre y fui creada por Google Labs en 2011, esta aplicación nos permite crear aplicaciones Android de manera visual y fácil gracias a un conjunto de herramientas básicas.
El desarrollo de la aplicación se hace de la manera siguiente:

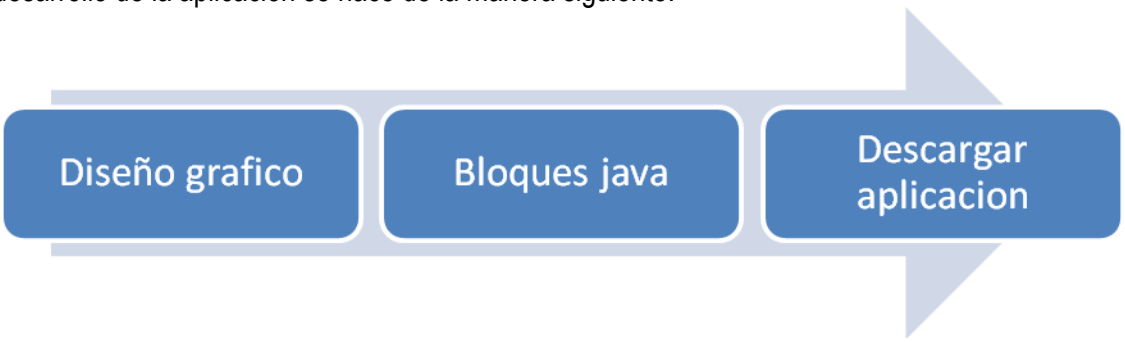


ILUSTRACIÓN 41: PROCESO CREACIÓN APLICACIÓN, APP INVENTOR 2

Como demuestra la "ilustración 41" primero se empieza diseñando la aplicación mediante un entorno de desarrollo gráfico, después para la programación se usa bloques de java, en estos bloques hay funciones, bucles y muchos elementos que se usan en los lenguajes de programación, una vez finalizamos el diseño y la programación de nuestra aplicación se descarga para poder instalarla en un dispositivo Android.
Para abrir la aplicación app inventor hay que tener una cuenta Gmail y después acceder a esta dirección[14]

[14] www.ai2.appinventor.mit.edu.

4.7.2. App inventor_2 proyecto nuevo

Al abrir un nuevo proyecto saldrá el App inventor designer donde diseñamos la interface de la aplicación.

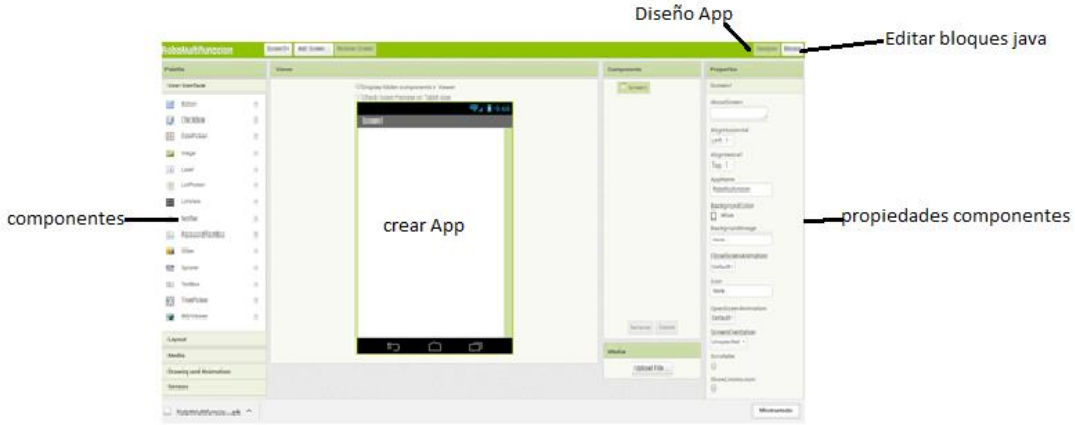


ILUSTRACIÓN 42: INTERFACE APP INVENTOR 2

Como se ve en la "ilustracion 42", tenemos en el centro una pantalla de un dispositivo Android donde iremos poniendo todos los componentes que necesitamos, los componentes están en la parte izquierda donde pone Paleta tenemos botones, tablas, imágenes, sensores, etc.
Solo tenemos que ir arrastrando componentes y poniéndolos en la pantalla de visualización, finalmente cuando tengamos todos los componentes en dicha pantalla, pasamos a la parte derecha de la App inventor donde pone Propiedades así podemos cambiar propiedades de los componentes como el nombre, tamaño, valores, color de texto y fondo y muchas más propiedades.

4.7.3. App inventor Control Robot

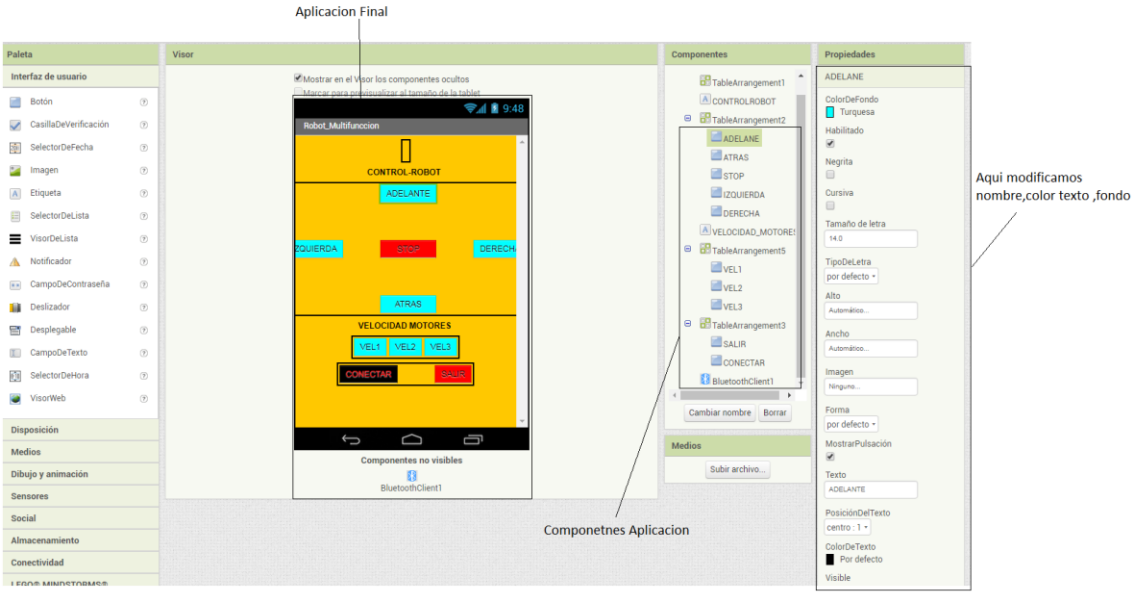


ILUSTRACIÓN 43: APLICACIÓN CONTROL-REMOTO DE ROBOT

Como se ve en la "ilustración 43" ,hemos puesto un total de 3 tablas, diez botones y dos etiquetas.

4.7.4. Funcionamiento interface aplicación

En esta aplicación hemos puesto un solo botón para conectarse y desconectarse al Arduino a través de Bluetooth, con un clic al botón conectar se conectan los dispositivos, pero si mantenemos un largo clic sobre conectar, se desconecta el Bluetooth cambiando color y texto del botón.
Cinco botones para el control del Robot: adelante, izquierda, derecha, atrás y stop. Con estos botones podemos mover nuestro robot en cualquier dirección.

Tres botones de velocidades de Motor: Vel1< Vel2<Vel3 así podemos aplicar a nuestro robot tres velocidades distintas.
Por último, apretar botón salir para salir de la aplicación.
Una vez terminamos el diseño de la aplicación, pulsamos el botón Bloques para pasar a la programación que se hace con bloques de java.

4.7.5. Funcionamiento bloques de programación

En este apartado es donde programamos la aplicación creada en forma de puzle, se empieza con una hoja en blanco donde se va poniendo los bloques que están en la parte izquierda, donde también hay también funciones, sentencias y muchos elementos comunes en los lenguajes de programación.

Empezamos por lomas fácil que es el botón Salir que al pulsarlo salimos se cierra la aplicación.
Antes de seguir con los otros botones, vamos a hacer una función que al iniciar la aplicación los botones estarán desactivados para evitar errores.
Así que inicializamos con todos los botones deshabilitados menos CONECTAR Y SALIR que sirven para inicializar la aplicación.

ILUSTRACIÓN 44: INICIO APLICACIÓN

Ahora que aseguramos un inicio sin errores de nuestra aplicación, procedamos con el botón Conectar.
La idea es usar el mismo botón para conectar y desconectar los dispositivos, así que usamos las dos funciones CONECTAR. Clic y CONECTAR.ClicLargo
Si pulsamos Conectar la aplicación se conectara con la dirección: **98:D3:31:FD:16:3D** obtenida al emparejar el móvil con el HC-06 lo que hace que aplicación se conecte al Arduino.

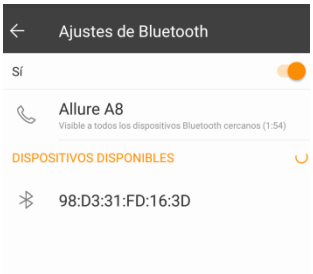


ILUSTRACIÓN 45: DIRECCIÓN MAC MÓDULO HC-06

Ahora tenemos que volver a activar todos los botones que deshabilitamos antes, ya que una vez estén conectados los dispositivos necesitamos que los controles estén habilitados.
También cambiamos el color del botón CONECTAR a negro y el texto a rojo una vez esté conectado.
Para desconectar hay que mantener pulsado el botón Conectar un rato, hasta que el botón se cambia de color rojo→negro y el texto de negro→rojo y también el texto de CONECTADO A DESCONECTADO, que a su vez deshabilitara todos los botones para evitar errores en la próxima conexión bluetooth.

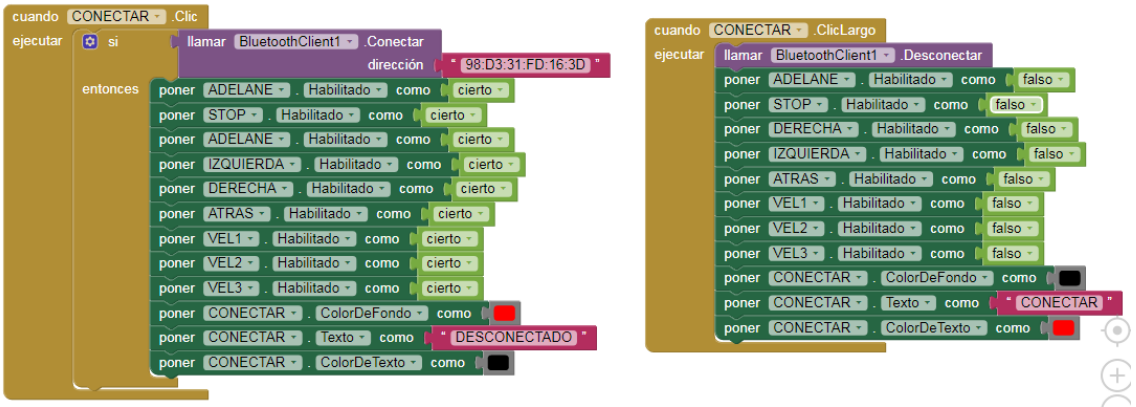


ILUSTRACIÓN 46: BLOQUES CONEXIÓN/DESCONEXIÓN BLUETOOTH.

Finalmente condicionamos las funciones de control robot mediante caracteres y velocidades mediante números como se muestra en la tabla siguiente:

Desplazamiento Robot	Velocidades Robot
Adelante : **A** Izquierda : **I** Stop : **S** Derecha : **D** Retroceder : **R**	Vel1 :**1** Vel2 :**2** Vel3 :**3**

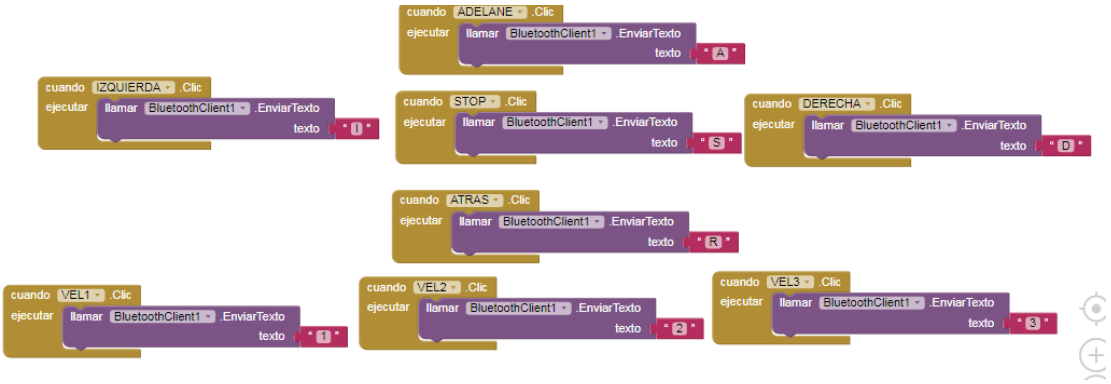


ILUSTRACIÓN 47: BLOQUES COMANDOS DIRECCIÓN ROBOT

4.7.6. Descargar Aplicación código QR

Ahora que terminamos nuestra aplicación solo queda descargarla, la opción más sencilla es en formato QR, ya que solo escaneado el código se puede descargar la aplicación al dispositivo Android.

ILUSTRACIÓN 48: CÓDIGO QR APLICACIÓN CONTROL-ROBOT

4.8. Diagrama de bloques

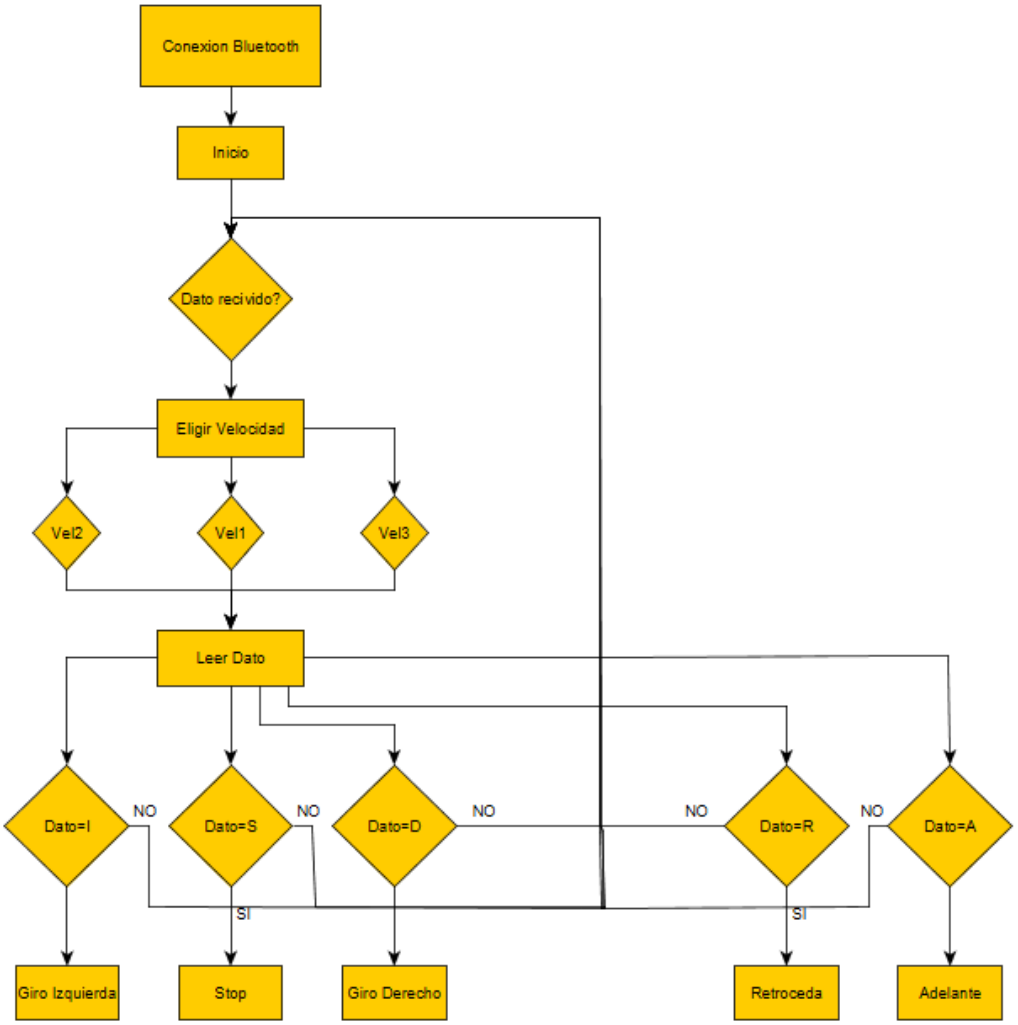


DIAGRAMA DE BLOQUES 3: DIAGRAMA CONTROL REMOTO ROBOT

Nuestro robot tendrá un funcionamiento igual al que esta reducido en el diagrama de bloques, primero definimos las variables de control de motores y definimos las velocidades del que va a mover nuestro Robot, una vez echo abrimos un setup() donde habilitamos comunicación puerto serie y declaramos lo pines de control motores como salida.
Segundo creamos una función para cada movimiento, así simplificamos nuestro código y será más sencillo el algoritmo de control:
Funciones:

- Adelante (); mover línea recta
- Retroceder (); hacer marcha atrás
- Izquierda (); girar a la izquierda
- Derecha (); girar a la derecha
- Stop (); motores parados

Estas funciones de ocupan de los movimientos de nuestro Robot, cuyas velocidades variables podemos cambiar según seleccionamos, vel1<vel2<vel3.
Así que siempre debemos seleccionar una velocidad nada más establecer la conexión, sino los motores tendrán velocidad 0 de inicio.
A continuación, creamos un pequeño código que cumpla estas condiciones y que nos sirva de plataforma para dar más funcionalidad a nuestro Robot.

4.9. CÓDIGO 11_Control manual Robot

```
// Motor derecho 1
int velocidad_M1 = 3;
int derB = 4;
int derA = 2;
// Motor izquierdo 2
int velocidad_M2 = 9;
int izqB = 8;
int izqA = 7;
int vel; //vel→tendrá el valor de vel1 o vel2 o vel3.
int vel1=250;
int vel2=160;
int vel3=110;
void setup()
{
  Serial.begin(9600) ; // comunicación puerto serie
  pinMode (velocidad_M1, OUTPUT); // Declaramos pines como salidas
  pinMode (velocidad_M2, OUTPUT);
  pinMode (derA, OUTPUT);
  pinMode (derB, OUTPUT);
  pinMode (izqA, OUTPUT);
  pinMode (izqB, OUTPUT);
}
void loop()
{
   if (Serial.available())// comprobamos si llega un dato
    switch (Serial.read())// si llega un dato lo leemos y lo comprobamos con los datos del switch
    {
      case'1':
        vel=vel1;
        break;
        case'2':
        vel=vel2;
        break;
      case'3':
        vel=vel3;
        break;
      case 'A':
        Adelante();
        break;
      case 'R':
        Retroceder();
        break;
      case 'D':
        Derecha();
        break;
      case 'I':
        Izquierda();
        break;
```

```
      case 'S':
        Stop();
        break;
    }
}
void Adelante () // función para mover robot en línea recta
{
  //Direccion motor A
  digitalWrite (derA, HIGH);
  digitalWrite (derB, LOW);
  analogWrite (velocidad_M1, vel);
  //Direccion motor B
  digitalWrite (izqA, HIGH);
  digitalWrite (izqB, LOW);
  analogWrite (velocidad_M2, vel);
}
void Retroceder () // función marcha Atrás Robot
{
  //Direccion motor A
  digitalWrite (derA, LOW);
  digitalWrite (derB, HIGH);
  analogWrite (velocidad_M1, vel);  //Velocidad motor A
  //Direccion motor B
   digitalWrite (izqA, LOW);
  digitalWrite (izqB, HIGH);
  analogWrite (velocidad_M2, vel); //Velocidad motor B
}
void Izquierda () // función que permite al Robot girar a la izquierda
{
  //Direccion motor A
  digitalWrite (derA, HIGH);
  digitalWrite (derB, LOW);
  analogWrite (velocidad_M1, vel); //Velocidad motor A
  //Direccion motor B
  digitalWrite (izqA, LOW);
  digitalWrite (izqB, HIGH);
  analogWrite (velocidad_M2, vel); //Velocidad motor A
}
void Derecha () // función que permite al Robot girar a la derecha
{
  //Direccion motor A
  digitalWrite (derA, LOW);
  digitalWrite (derB, HIGH);

  analogWrite (velocidad_M1, vel);
  //Direccion motor B
  digitalWrite (izqA, HIGH);
  digitalWrite (izqB, LOW);
  analogWrite (velocidad_M2, vel);
}
```

```
void Stop () // función para detener los motores del Robot
{
  //Direccion motor A
  digitalWrite (derA, LOW);
  digitalWrite (derB, LOW);
  analogWrite (velocidad_M1, 0);

  //Direccion motor B
  digitalWrite (izqA, LOW);
  digitalWrite (izqB, LOW);
  analogWrite (velocidad_M2, 0);
}
```

3.1 RESUMEN

En esta fase hemos creado una aplicación mediante “App inventor 2 “que a su vez instalamos en un dispositivo Android con Bluetooth, que nos permita controlar remotamente nuestro robot.

Gracias al módulo Bluetooth HC-06 logramos conectar el dispositivo Android con la tarjeta Arduino y controlar los movimientos del robot.

Los resultados han sido muy satisfactorios, además logramos variar la velocidad del robot remotamente.

A continuación, vamos a darle diversos modos de empleo a nuestro robot, creando un código capaz de ejecutar todas las funciones desarrolladas en las fases anteriores y mejorando la aplicación creada a través de App inventor para poder tener un correcto control remoto del robot.

5. FASE CUATRO, CONTROL REMOTO DE ROBOT_MULTIFUNCION

En esta fase se procederá a la proyección de lo visto anteriormente, resumiendo las tres fases en una sola, montando un solo Robot multifunción que será capaz de ejecutar las distintas tareas, seguidor de línea, evita obstáculos y también la opción de control manual para poder dirigirlo al lugar exacto donde queremos que ejecute un modo u otro.

5.1. OBJETIVO GENERAL

Nuestro objetivo principal es crear un solo Robot Multifunción con tres modos de funcionamiento, primero que se pueda controlarlo manualmente, pueda seguir línea y finalmente evita obstáculos.

5.1.1. objetivos específicos

- diseñar y crear un circuito de un Robot Multifunción que cumpla las funciones de todas las fases anteriores.
- Sincronizar Android con Arduino vía Bluetooth a través del HC-06
- Crear aplicación en App Inventor para control Remoto del Robot Multifunción
- Diseñar código Arduino capaz de recibir y ejecutar las distintas órdenes.

5.2. Materiales

- ✓ Chasis robot (base robot, soporte motores, rueda loca)
- ✓ Placa Arduino UNO
- ✓ Motores DC
- ✓ ruedas
- ✓ Módulo L298N
- ✓ Cables macho-macho
- ✓ Cables macho-hembra
- ✓ Conector Jack-macho + batería 9V
- ✓ soporte pilas
- ✓ baterías 1.5V
- ✓ Protoboard-mini
- ✓ Sensor HC-06
- ✓ Sensor HC-SR04
- ✓ Sensor TCRT500
- ✓ Móvil con sistema Android
- ✓ interruptor

Software

- ✓ IDE Arduino
- ✓ Fritzing
- ✓ App inventor2

5.3. ESQUEMA ELÉCTRICO ROBOT_MULTIFUNCION

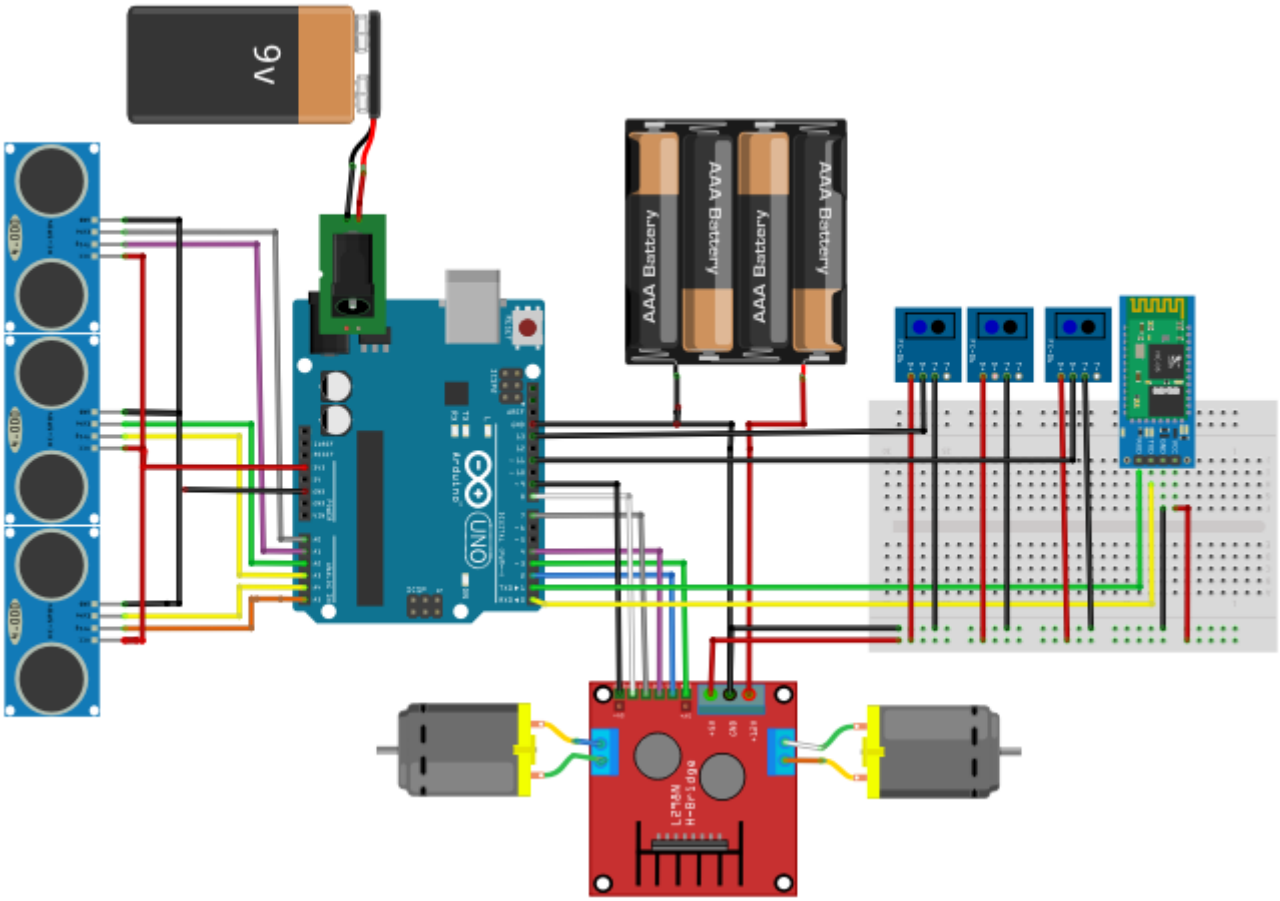


FIGURA 22 : ESQUEMA ROBOT MULTIFUNCIÓN

5.4. CREAR APLICACIÓN DE CONTROL REMOTO PARA ANDROID

En este apartado podemos modificar la aplicación creada anteriormente en la fase tres del proyecto para cumplir nuestros objetivos.
La nueva aplicación tendrá un apartado nuevo de modo de empleo, sirve para elegir entre modo Manual, Seguidor de línea, evasor de obstáculos

5.5. APP IVENTOR

La aplicación final creada, será un poco mejorada ya que le podemos añadir un fondo además introducir imágenes en los botones así logramos una interface más sofisticada.

ILUSTRACIÓN 49: APLICACIÓN CONTROL ROBOT MULTIFUNCIÓN

Como se ve en la ilustración 47, hemos intentado hacer un interface más docente y moderna partiendo de la aplicación echa en la fase dos, añadiendo tres botones más que nos sirven de elección del modo de empleo.

Bloques JAVA:

Como hemos hecho en la fase anterior empezamos deshabilitando todos los botones menos el de CONECTAR Y SALIR para evitar errores no deseados a la hora de sincronizar la Aplicación con el nuestro ROBOT.

Botón Salir para cerrar la aplicación.

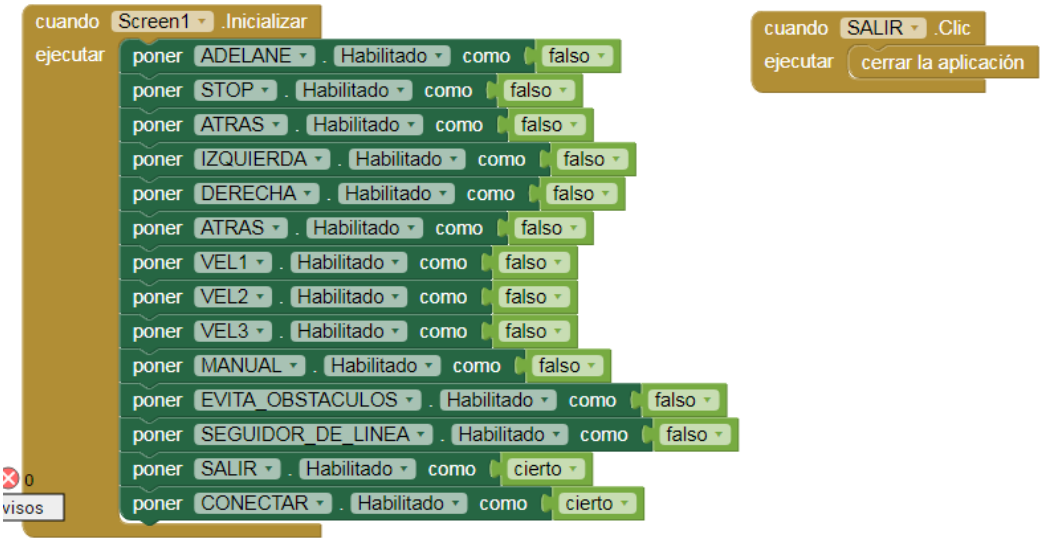


ILUSTRACIÓN 50: INICIO BLOQUES APLICACIÓN ROBOT

Despues configuramos el boton CONECTAR ,con un clic se conecta la aplicación a nuestro Robot a traves del HC-06 cuya MAC 96:D3:31:FD:16:3D , habilitando todos los botones de paso.

Al mantener apretado el boton CONECTAR un rato se desincronizan la aplicación y el Robot Tambien se desabilitan todos los botones , asi evitamos errores en la proxima sincronizacion.

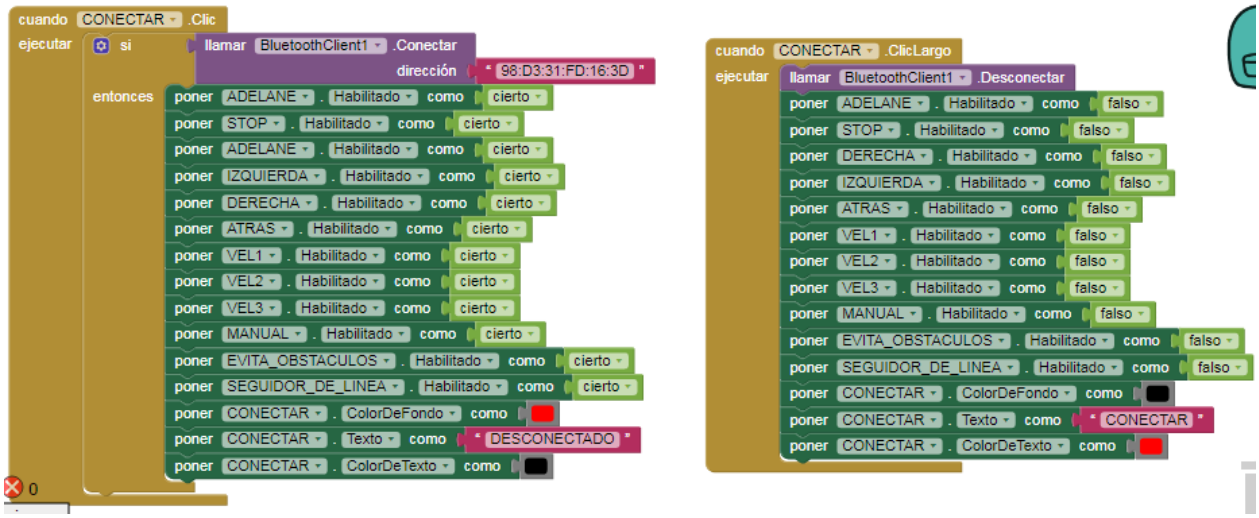


ILUSTRACIÓN 51: CONEXIÓN/DESCONEXIÓN APP ROBOT

Mantenemos las mismas órdenes de la fase Anterior como se demuestra a continuación

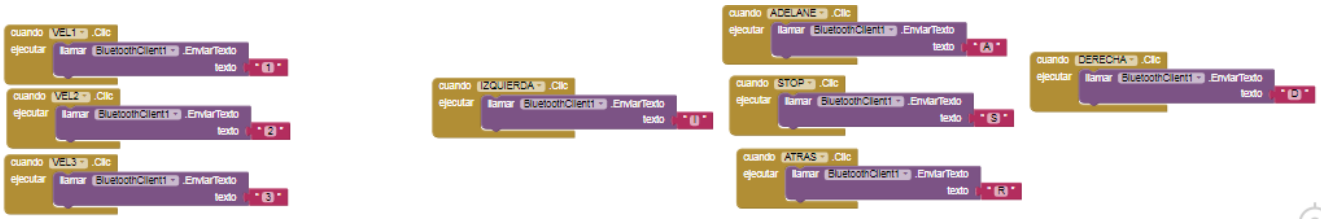


ILUSTRACIÓN 52: COMANDOS DESPLAZAMIENTO ROBOT

Finalmente añadimos los tres botones correspondientes a los tres modos de trabajo, habilitando los botones de desplazamiento del robot solo para el modo manual, así aseguramos un buen funcionamiento de los tres modos.

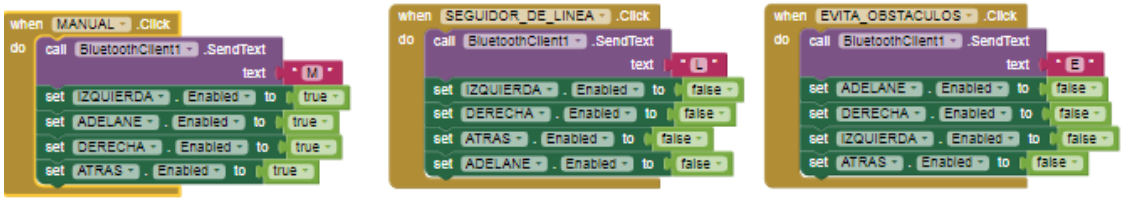


ILUSTRACIÓN 53: MODO TRABAJO ROBOT

Una vez hecha la parte de bloques java, procedemos a descargar nuestra aplicación para tenerla lista para el uso.

5.6. DESCARGAR APLICACIÓN ROBOT_MULTIFUNCION

ILUSTRACIÓN 54: APLICACIÓN FINAL ROBOT

5.7. DIAGRAMA DE BLOQUES

Nuestro diagrama será de la siguiente manera:

Primero restablecer conexión Bluetooth apretando al botón CONECTAR, después comprobamos la sincronización del robot con la aplicación.

Segundo elegimos un modo de trabajo, en el caso de Control Manual debemos seleccionar la velocidad deseada entres tres valores Vel1, Vel2, Vel3.

Una vez finalizado el uso de Robot, mantenemos apretado el botón Conectar un segundos, hasta que cambie de color y se desconecta la aplicación del Robot.

Finalmente, con el botón salir, se sale de la aplicación.

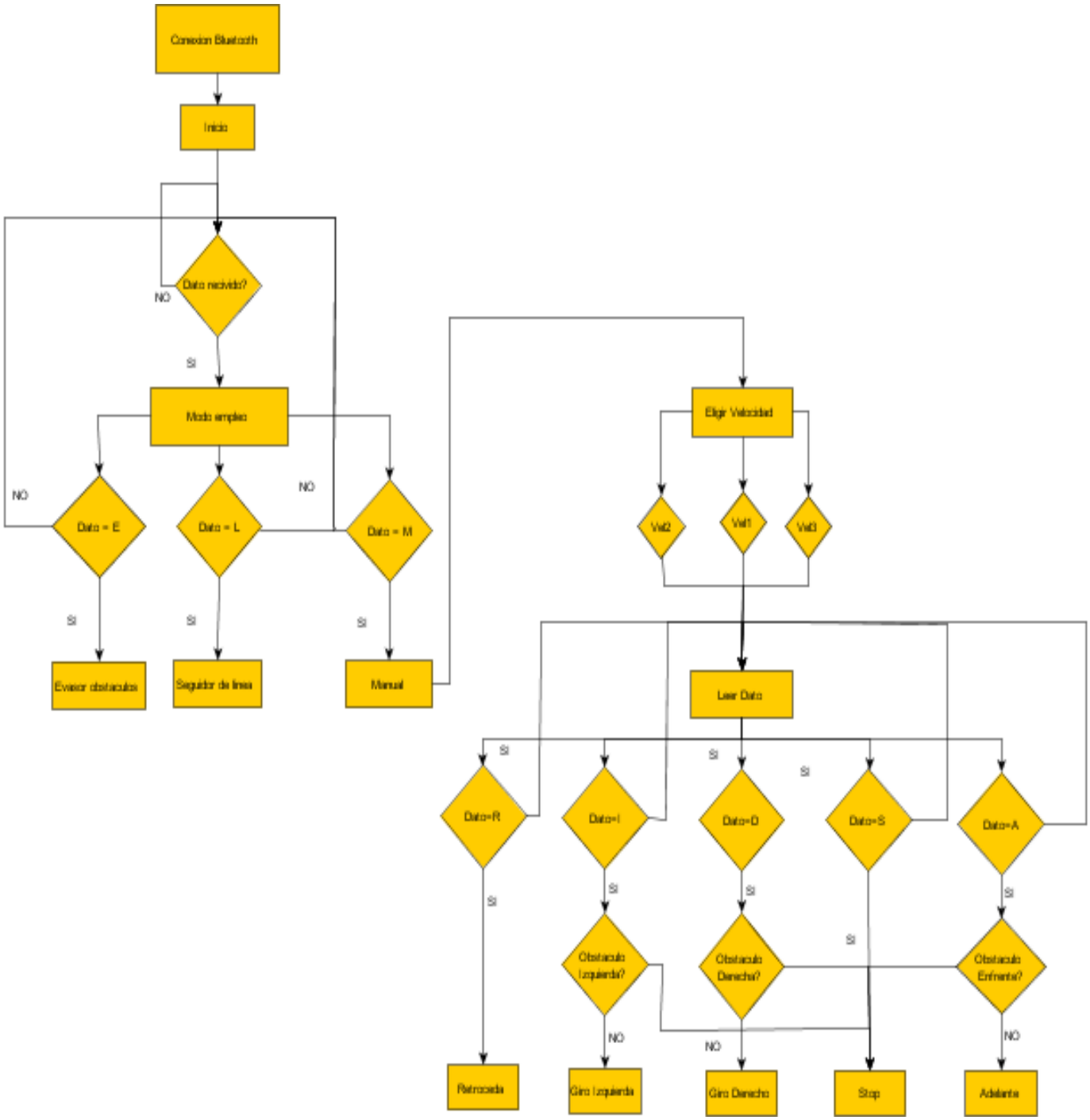


DIAGRAMA DE BLOQUES 4: ROBOT MULTIFUNCIÓN

5.8. Código12_Robot Multifunción

```
// Definición de variables control motor derecho
const int derA = 2;
const int derB= 4;
const int velocidad_M1=3;
// Definición de variables control motor izquierdo
const int izqA = 7;
const int izqB = 8;
const int velocidad_M2= 9;

   int trigger_der = A0;// impulso enviado sensor derecho
   int echo_der = A1;

   int trigger_cen = A2; // impulso sensor central
   int echo_cen = A3;

   int trigger_izq = A4;// impulso sensor izquierdo
   int echo_izq = A5;//

long tiempo1,tiempo2,tiempo3;  // Almacena el tiempo de respuesta de los sensores
float distancia_izq, distancia_cen, distancia_der;  // Almacena la distancia en cm a la que se
encuentra el obstáculo
int vel0=180;// velocidad motores evita obstáculos
int vel10=200;
int vel=0; //vel que tendrá el valor de (vel1, vel2 o vel3) según la velocidad seleccionada.
int vel1=250;
int vel2=180;
int vel3=120;
// vaiables sensores infrarojos
const int Sen1 = 11; // Izquierdo
const int Sen2 = 12; //centro
const int Sen3 = 13; //Derecho
int ValSen1 = 0; // Valor de los sensores
int ValSen2 = 0;
int ValSen3 = 0;
int dato='M';
void setup()
{
  //  variables Entradas/Salidas
  pinMode (derA, OUTPUT);// rueda derecha sentido horario
  pinMode (derB, OUTPUT);// rueda derecha sentido anti-horario
  pinMode (izqA, OUTPUT);// rueda izquierda sentido horario
  pinMode (izqB, OUTPUT);// rueda izquierda sentido anti-horario
  pinMode(velocidad_M1,OUTPUT);// velocidad motor derecha
  pinMode(velocidad_M2,OUTPUT);// velocidad motor izquierda
```

```
  pinMode(trigger_cen, OUTPUT);
   pinMode(echo_cen, INPUT);
   pinMode(trigger_izq, OUTPUT);
   pinMode(echo_izq, INPUT);
   pinMode(trigger_der, OUTPUT);
   pinMode(echo_der, INPUT);

  pinMode(Sen1, INPUT);
  pinMode(Sen2, INPUT);
  pinMode(Sen3, INPUT);
     Serial.begin(9600);
}
void loop() {
   if (Serial.available()>0)
   {// comprobamos si llega un dato
    dato= Serial.read();// si llega un dato lo leemos y lo comprobamos  .
   }
       if(dato=='1')
       {
        vel=vel1;
       }
       if(dato=='2')
       {
        vel=vel2;
       }
       if(dato=='3')
       {
        vel=vel3;
       }
       if(dato=='A')
       {
        Adelante_M();
       }
             if(dato=='R')
       {
       Retroceder_M();
       }
             if(dato=='D')
       {
       Derecha_M();
       }
             if(dato=='I')
       {
       Izquierda_M();
       }
             if(dato=='S')
       {
        Stop();
       }
```

```
      if(dato=='E')
      {
      evita_obstaculos();
      }
   if(dato=='L')
   {
   seguidor_de_linea();
    }
}

void evita_obstaculos()
{
sensordis_cen();
sensordis_der();
sensordis_izq();
Adelante();
if(distancia_cen<23)
{
  Stop();
 if(distancia_der>distancia_izq)
 {
Derecha();
 }
else
{
  Izquierda();
  }
}
 if((distancia_der<12)&&(distancia_cen<12)&&(distancia_izq<12))
 {
  Retroceder_despacio();
  delay(400);
  }
 if((distancia_izq<6)||(distancia_cen<12)||(distancia_der<6))
 {
   Retroceder_despacio();
   delay(400);
 }
if(distancia_izq<14)
{
  Derecha();
}
if(distancia_der<14)
{
  Izquierda();
}
}
```

```
 void seguidor_de_linea()
 {
ValSen1 = digitalRead(Sen1);//                  izquierdo
ValSen2 = digitalRead(Sen2);// valor entrada que lee el infrarrojo central
ValSen3 = digitalRead(Sen3);//                  derecho
// lectura de sensores 0=Blanco
 //                1=negro
    Serial.println("sensor izquierdo infrarojo ");
   Serial.println(Sen1);
    Serial.println("sensor central infrarojo ");
   Serial.println(Sen2);
    Serial.println("sensor Derecho infrarojo ");
   Serial.println(Sen3);

if ((ValSen1==0)&&(ValSen2== 0)&&(ValSen3== 0))
{
  Retroceder();
}
if ((ValSen1==0)&&(ValSen2== 0)&&(ValSen3== 1))
{
  Derecha();
}
if ((ValSen1==0)&&(ValSen2== 1)&&(ValSen3== 0))
{
  Adelante();
}
if ((ValSen1==1)&&(ValSen2== 0)&&(ValSen3==0 ))
{
 Izquierda();
}
if((ValSen1==1)&&(ValSen2== 1)&&(ValSen3==1))
{
  Stop();
}
if((ValSen1==1)&&(ValSen2== 1)&&(ValSen3==0))
{
  Adelante_der();
}

if((ValSen1==0)&&(ValSen2== 1)&&(ValSen3==1))
{
 Adelante_izq();
}
}
```

```
void Adelante()
{
 // Motor derecho
  digitalWrite (derA, HIGH);//rueda derecha sentido horario
  digitalWrite (derB, LOW);// rueda derecha sentido anti-horario
// Motor izquierdo
  digitalWrite (izqA, HIGH);// rueda izquierda sentido horario
  digitalWrite (izqB, LOW);// rueda izquierda sentido anti-horario
    analogWrite (velocidad_M1,vel0);// velocidad motor derecha
    analogWrite (velocidad_M2,vel0);// velocidad motor izquierdo
}
void Adelante_M()
{
  if(distancia_cen<20)
  {
Stop();
  }
  else
  {
  digitalWrite (derA, HIGH);//rueda derecha sentido horario
  digitalWrite (derB, LOW);// rueda derecha sentido anti-horario
 digitalWrite (izqA, HIGH);// rueda izquierda sentido horario
  digitalWrite (izqB, LOW);// rueda izquierda sentido anti-horario
    analogWrite (velocidad_M1,vel);// velocidad motor derecha
    analogWrite (velocidad_M2,vel);// velocidad motor izquierdo
}
}
void Adelante_der()
{
// Motor derecho
  digitalWrite (derA, HIGH);//rueda derecha sentido horario
  digitalWrite (derB, LOW);// rueda derecha sentido anti-horario
// Motor izquierdo
   digitalWrite (izqA, HIGH);// rueda izquierda sentido horario
   digitalWrite (izqB, LOW);// rueda izquierda sentido anti-horario
    analogWrite (velocidad_M1,vel10);// velocidad motor derecha
    analogWrite (velocidad_M2,vel0);// velocidad motor izquierdo
}
void Adelante_izq()
// Motor derecho
  digitalWrite (derA, HIGH);//rueda derecha sentido horario
  digitalWrite (derB, LOW);// rueda derecha sentido anti-horario
// Motor izquierdo
  digitalWrite (izqA, HIGH);// rueda izquierda sentido horario
  digitalWrite (izqB, LOW);// rueda izquierda sentido anti-horario
    analogWrite (velocidad_M1,vel0);// velocidad motor derecha
    analogWrite (velocidad_M2,vel10);// velocidad motor izquierdo
}
```

```
// Función Retroceder: esta función hará que ambos motores se activen a máxima potencia
  en sentido contrario al anterior por lo que el robot avanzará hacia atrás
void Retroceder()
{
  // Motor derecho
  digitalWrite (derA, LOW);// rueda derecha sentido horario
  digitalWrite (derB, HIGH);// rueda derecha sentido anti-horario
  // Motor izquierdo
  digitalWrite (izqA, LOW);// rueda izquierda sentido horario
  digitalWrite (izqB, HIGH);// rueda izquierda sentido anti-horario
   analogWrite (velocidad_M1,vel0);// velocidad motor derecha
    analogWrite (velocidad_M2,vel0);// velocidad motor izquierdo
}
void Retroceder_despacio()
{
  // Motor derecho
  digitalWrite (derA, LOW);// rueda derecha sentido horario
  digitalWrite (derB, HIGH);// rueda derecha sentido anti-horario
  // Motor izquierdo
  digitalWrite (izqA, LOW);// rueda izquierda sentido horario
  digitalWrite (izqB, HIGH);// rueda izquierda sentido anti-horario
   analogWrite (velocidad_M1,vel10);// velocidad motor derecha
    analogWrite (velocidad_M2,vel10);// velocidad motor izquierdo
}
void Retroceder_M()
{
  // Motor derecho
  digitalWrite (derA, LOW);// rueda derecha sentido horario
  digitalWrite (derB, HIGH);// rueda derecha sentido anti-horario
  // Motor izquierdo
  digitalWrite (izqA, LOW);// rueda izquierda sentido horario
  digitalWrite (izqB, HIGH);// rueda izquierda sentido anti-horario
   analogWrite (velocidad_M1,vel);// velocidad motor derecha
    analogWrite (velocidad_M2,vel);// velocidad motor izquierdo
}
 // Función Izquierda: esta función accionar el motor derecho y parará el izquierdo por lo que el
robot girará a la izquierda.
void Izquierda()
{
  // Motor derecho
  digitalWrite (derA, HIGH);// rueda derecha sentido horario
  digitalWrite (derB, LOW);//rueda derecha sentido anti-horario
// Motor izquierdo
 digitalWrite (izqA, LOW);//rueda izquierda sentido horario
  digitalWrite (izqB, LOW);// rueda izquierda sentido anti-horario
  analogWrite (velocidad_M1,vel10);// velocidad motor derecha
  analogWrite (velocidad_M2,vel10);// velocidad motor izquierdo
}
```

```
void Izquierda_M()
{
  // Motor derecho
  digitalWrite (derA, HIGH);// rueda derecha sentido horario
  digitalWrite (derB, LOW);//rueda derecha sentido anti-horario
// Motor izquierdo
  digitalWrite (izqA, LOW);//rueda izquierda sentido horario
  digitalWrite (izqB, LOW);// rueda izquierda sentido anti-horario
  analogWrite (velocidad_M1,vel);// velocidad motor derecha
  analogWrite (velocidad_M2,vel);// velocidad motor izquierdo
}

// Función Derecha: esta función accionar el motor izquierdo y parará el derecho por lo que el
robot girará hacia la izquierda.
void Derecha ()
{
// Motor derecho
 digitalWrite (derA, LOW);// rueda derecha sentido horario
  digitalWrite (derB, LOW);//rueda derecha sentido anti-horario
// Motor izquierdo
 digitalWrite (izqA, HIGH);//rueda izquierda sentido horario
  digitalWrite (izqB, LOW);// rueda izquierda sentido anti-horario
  analogWrite (velocidad_M1,vel10);// velocidad motor derecha
  analogWrite (velocidad_M2,vel10);// velocidad motor izquierdo
}
void Derecha_M ()
{
// Motor derecho
  digitalWrite (derA, LOW);// rueda derecha sentido horario
  digitalWrite (derB, LOW);//rueda derecha sentido anti-horario
  // Motor izquierdo
 digitalWrite (izqA, HIGH);//rueda izquierda sentido horario
  digitalWrite (izqB, LOW);// rueda izquierda sentido anti-horario
  analogWrite (velocidad_M1,vel);// velocidad motor derecha
  analogWrite (velocidad_M2,vel);// velocidad motor izquierdo
}
  //Función Stop: esta función parará ambos motores
  por lo que el robot se parará.
void Stop()
{
// Motor derecho
 digitalWrite (derA, LOW);// rueda derecha sentido horario
  digitalWrite (derB, LOW);//rueda derecha sentido anti-horario
// Motor izquierdo
  digitalWrite (izqA, LOW);//rueda izquierda sentido horario
  digitalWrite (izqB, LOW);// rueda izquierda sentido anti-horario

}
```

```
void sensordis_cen()
{
   // Se inicializa el sensor de infrasonidos
   digitalWrite(trigger_cen,LOW);  // Para estabilizar
   delayMicroseconds(50);
   // enviamos una señal activando la salida trigger durante 10 microsegundos
   digitalWrite(trigger_cen, HIGH);  // envío del pulso ultrasónico
   delayMicroseconds(50);
   tiempo1=pulseIn(echo_cen, HIGH); //  tiempo que tarda la señal
   distancia_cen= int(0.017*tiempo1); // Fórmula para calcular la distancia en cm
   Serial.println("El valor de la distancia_ce es ");
   Serial.println(distancia_cen);
   delay(10);
   }
    void sensordis_izq()
{
     // Se inicializa el sensor de infrasonidos
   digitalWrite(trigger_izq,LOW);  // Para estabilizar
   delayMicroseconds(10);
  // enviamos una señal activando la salida trigger durante 10 microsegundos
   digitalWrite(trigger_izq, HIGH);  // envío del pulso ultrasónico
   delayMicroseconds(10);
   tiempo2=pulseIn(echo_izq, HIGH); //  tiempo que tarda la señal
   distancia_izq= int(0.017*tiempo2); // Fórmula para calcular la distancia en cm
   Serial.println("El valor de la distancia_izq es ");
   Serial.println(distancia_izq);
   delay(10);
    }
void sensordis_der()
{
     // Se inicializa el sensor de infrasonidos
   digitalWrite(trigger_der,LOW);  // Para estabilizar
   delayMicroseconds(10);
   // Se envía una señal activando la salida trigger durante 10 microsegundos
   digitalWrite(trigger_der, HIGH);  // envío del pulso ultrasónico
   delayMicroseconds(10);
   tiempo3=pulseIn(echo_der, HIGH); //  tiempo que tarda la señal
   distancia_der= int(0.017*tiempo3); // Fórmula para calcular la distancia en cm
   Serial.println("El valor de la distancia_der es ");
   Serial.println(distancia_der);
   delay(10);
}
```

5.9 Conclusiones

Como conclusiones cabe destacar que se han cumplido los objetivos propuestos en el siguiente proyecto, desarrollando una plataforma sencilla y de bajo coste.

En cuanto al desarrollo de este proyecto, se han hecho prácticas para llevar a cabo el control de los distintos modos de empleo de nuestro Robot en varias fases, ya que al intentar hacerlo directamente en una fase obtenemos muchos errores y fallos difícil de averiguar y corregir.

También obtenemos varios fallos de hardware y software, al conectar todos los componentes a la tarjeta Arduino, así que para proyectos similares se recomienda utilizar otra tarjeta Arduino mucho más potente como el "Arduino mega".

Primero, se han desarrollado prácticas y ensayos para llevar acabo un mejor control de los motores a pesar de su baja resolución, se ha obtenido resultados satisfactorios para asegurar la movilidad y estabilidad del Robot.

Segundo, se han hecho prácticas y ensayos de todos los sensores y dispositivos que forman parte de este proyecto para entender y asegurar su buen funcionamiento.

Tercero, En cuanto al desarrollo del software hemos intentado hacer un código simple, creando funciones propias sencillas para entender mejor el código evitando usar funciones creadas por la comunidad Arduino.

Finalmente, por otra parte, uno de los objetivos principales consistía en crear una interfaz de control remoto de los diferentes modos de empleo de nuestro Robot final, usando la aplicación App inventor 2 se ha cumplido con creces este objetivo.

IMAGEN 74: ROBOT MULTIFUNCIÓN Y APLICACIÓN DE CONTROL FINAL

5.10 Trabajos Futuros

A pesar de que en este proyecto se ha utilizado un módulo Bluetooth en la plataforma para control remoto del robot desarrollado, por lo tanto, será mucho mejor sustituir el cable USB por dicho modulo, cosa que facilitará la tarea de la depuración de los programas diseñados al no tener que estar conectando cada vez las placas Arduino mediante el cable USB al PC.

Otra posible mejora para la plataforma será cambiar los motores por otros con mejores rendimiento y resolución, ya que a velocidad pequeñas además del peso de la plataforma les resulta casi imposible mover, también sería una mejora considerable poner algún tipo de batería con más autonomía e un indicador de carga que nos avise a la hora de baja batería.

También podemos acoplar a nuestro robot un brazo capaz de coger pequeños objetos o muestras desde lugares contaminados o de difícil acceso.

Finalmente, cambiar la tarjeta Arduino UNO por otra tarjeta más potente y con más entradas salidas, servirá para dar más funcionalidad a nuestro Robot conectando más sensores, dando más opciones de control remoto y geolocalización, además de poner una cámara que nos facilita el control manual del Robot en zonas de difícil acceso.

6. PLIEGO DE CONDICIONES PARTICULARES O ESPECIFICACIÓN TÉCNICA

En este documento exponemos todas las condiciones técnicas de montaje, soluciones aportadas además de especificaciones a cumplir por los elementos y materiales normalizados y comerciales que se deben considerar para poder llevar acabo la realización del proyecto.
En el caso de no seguir las instrucciones según las condiciones tal y como se muestran en este documento, el proyectista no se responsabilizará de los posibles averías y fallos que puedan dañar seriamente distintos dispositivos del robot a la hora de ejecutarlo.
La descripción del diseño de la fase del proyecto su conexión y su control de funcionamiento están detallados en el documento de la memoria y en los correspondientes esquemas y tablas de conexiones.

6.1. Condiciones generales

La realización del presente proyecto titulado “sistemas integrados con Arduino “, tiene como propósito finalizar con éxito los estudios de ingeniería industrial en la especialidad de electrónica.
Este proyecto se ajusta en su desarrollo a las normas y reglamentos electrónicos vigentes de circuitos de baja tensión.
Se puede mejorar u realizar modificaciones al proyecto siempre bajo la supervisión del proyectista, considerando el proyecto modificado parte integrante del proyecto definido, sujeto a las especificaciones aprobadas por el ministerio de industria.

6.2. Normas, leyes y reglamentos

Nuestro proyecto, seguirá los reglamentos electrónicos de baja tensión que se tendrá en cuenta las siguientes normativas.

- UNE 20-050-74 (I). Código resistencia y condensadores, valores y tolerancias
- UNE 20-524 (I). Equipos electrónicos y sus componentes. Soldabilidad de circuitos impresos.
- UNE 20-524-75 (I). Técnicas circuitos impresos. Sistemas de cuadricula.
- UNE 20-524-77 (II). Técnicas circuitos impresos. Terminología
- UNE 20-531-73 . Valores nominales para resistencias y condensadores
- UNE 20-543-85 (I). Condensadores fijos en equipos electrónicos.
- UNE 20-545-89 . Resistencias fijas para equipos electrónicos.
- UNE 20916: 1995: Estructuras mecánicas para equipos electrónicos. Terminología.
- UNE 21302-2:1973: Vocabulario electrotécnico. Electrónica de potencia.
- UNE 213302 -551: 1996: Vocabulario electrotécnico internacional.
- UNE 21352:1976: Explicación de las cualidades y funcionamiento de equipos de media electrónicos
- UNE-EN61000-4-3-1998: Compatibilidad electromagnética.
- EN60852-4: 1996: Dimensiones extremas de transformadores e inductancias destinadas a equipos electrónicos y de telecomunicación.
- EN123500: 1992: Especificación intermedia: placas de circuitos impresos flexibles con taladros para la inserción de componentes.
- DIN 40801 referente a circuitos impresos, fundamentos y espesores.
- DIN 40803 referente a circuitos impresos, conceptos.
- DIN 40804 referente a circuitos impresos, conceptos.
- DIN 41494 referente a la forma de construcción para dispositivos electrónicos.

- M.I B.T.029, la cual se refiere a instalaciones de pequeñas tensiones inferiores a 50 Voltios.
- M.I B.T.031, la cual se refiere a las condiciones generales de instalaciones, de utilización, así como de los requisitos a cumplir a la hora del diseño.

6.3. LIBRO DE ÓRDENES

El ensamblaje del proyecto se realizará atendiendo a los documentos, esquemas y conexiones del mismo.
En el caso de realizar alguna modificación, se realizará bajo el consentimiento del propio proyectista.

6.4. CONDICIONES DE EJECUCIÓN Y MONTAJE

En primer lugar, se adquieren los diversos componentes teniendo en cuenta sus especificaciones técnicas, a continuación, se hacen pruebas y ensayos mediante circuitos montados en una placa Protoboard, después se obtiene placas de circuito impreso basándose en las pautas anteriores.
Finalmente, montaje y cableado y soldadura de los diversos elementos que componen nuestro Robot.

6.4.1. Condición de fabricación del circuito impreso

En el caso de optar por la fabricación de la tarjeta Arduino propia, a partir de los diseños de referencia ofrecidos por la página web de Arduino, el diseño y la fabricación del circuito impreso debe seguir la norma UNE-621-80, donde se especifican los ensayos que debe ser sometido el circuito impreso y los métodos para la ejecución de dichos ensayos explicados en la fase uno del proyecto.

6.4.2. Ensayos del montaje de la tarjeta Arduino

En el caso de diseñar nuestra propia tarjeta Arduino u optar por comprar un modelo comercial se debe proceder a realizar ciertos ensayos para asegurar el buen funcionamiento de la tarjeta Arduino.

6.4.2.1. Prueba de alimentación

La tarjeta Arduino se debe alimentar a través de los 5V del USB o mediante una batería de 9V conectada al Jack macho de la tarjeta Arduino u con transformador de 230V a 7...12V.

También debemos comprobar que al conectar ambas tensiones no se produzca ningún cortocircuito o fallo entre las alimentaciones.

En el caso de conectar una alimentación externa, se debe comprobar que el pin Vin tenga la misma tensión que la que saca el transformador o la batería, independientemente del modo en que este alimentado, donde tendremos en los pines de alimentación marcados como 5V y 3V3, 5V y 3.3 V consecutivamente.

Las tarjetas Arduino suelen tener un Led verde que se ilumina cuando la tarjeta está siendo alimentada correctamente, en el caso de producir algún cortocircuito al realizar alguna conexión se activará el limitador de corriente interno de la tarjeta Arduino haciendo que LED desvanecerá lentamente indicando que está produciendo algún cortocircuito.

6.4.2.2. Prueba de la placa

Para comprobar el buen funcionamiento de nuestra placa debemos utilizar el IDE Arduino, cargando un pequeño programa que hace parpadear el Led interno que la tarjeta Arduino tiene conectado al pin 13, esta prueba es ideal para asegurar que si se produce algún fallo sea de hardware y no esté siendo enmascarado por un fallo de software.
La prueba está realizada en el apartado “2.5 configuración de Hardware”

6.4.2.3. Conexionado de los circuitos

Las conexiones de los circuitos que intervienen en el sistema se realizaran siguiendo las instrucciones provistas por el fabricante del hardware.

6.4.3. Ensayos y pruebas de los sensores

El proyecto final está compuesto por tres fases donde comprobamos el funcionamiento de los diversos sensores en cada apartado para minimizar y solucionar cualquier fallo antes de proceder a realizar el montaje final.

6.4.3.1. Conexionado de los sensores

Las conexiones de los sensores que intervienen en el sistema se realizaran siguiendo las tablas de conexiones provistas por el ingeniero proyectista.

6.4.4. Descripción de proceso de ejecución

Paso1**:**
Comprobación de la tarjeta Arduino UNO.

Paso2:
Comprobación y configuración de los motores DC para darle movilidad a nuestro Robot.

Paso3:
Comprobación de los sensores TCRT500

Paso4:
Comprobación de los sensores HC-SR04

Paso5:
Comprobación de la comunicación Bluetooth a través del módulo HC-06

Paso6:
Comprobación y montaje de todos los elementos del Robot Multifunción.

Paso7:
Crear aplicación de control remoto para nuestro Robot Multifunción.

Paso8:
Crear y cargar código para la ejecución de los diferentes modos de trabajo de nuestro Robot.

6.4.4.1. Mando

Para el mando se creará una aplicación específica para los diferentes modos de trabajo que será descargada en un dispositivo con sistema operativo Android, para el control remoto del Robot Multifunción.

6.4.4.2. Modos de trabajo

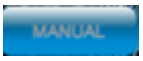
Nuestro Robot tendrá 3 modos de marcha:

Modo seguidor de línea 

Cuya finalidad es seguir una línea negra, encima de una superficie blanca u clara.
En este modo (Adelante, Retroceder, Izquierda, Derecha)→ deshabilitados

Modo evita obstáculo 

Cuya finalidad es detectar y evitar obstáculos y buscar mejores rutas.
En este modo (Adelante, Retroceder, Izquierda, Derecha)→ deshabilitados

Modo manual 

Cuya finalidad es control manual de los movimientos del Robot en los cuatro sentidos Adelante, atrás, derecha, izquierda y parada.
En este modo (Adelante, Retroceder, Izquierda, Derecha)→ habilitados

opción de variar la velocidad, velocidad inicial es por lo que el robot se inicia parado.

VEL1 < VEL2 < VEL3

Parada de emergencia
STOP: botón stop se mantendrá habilitado en todos los modos, para poder parar el robot en caso de emergencia.

6.4.5. Conservación

Para el montaje y puesta en marcha de los diversos dispositivos y materiales que componen nuestro Robot Multifunción, se sigue todas las indicaciones recomendaciones y las especificaciones que se dan en el pliegue de condiciones, memoria y anexos, la vida útil de los elementos estará sujeta al fabricando siguiendo su recomendación de mantenimiento.

6.4.6. Mantenimiento y vida útil del Robot

Nuestro Robot no necesita un mantenimiento riguroso, siempre que su uso se realice dentro de los límites especificados por el fabricante de cada componente, así que nuestro robot estará listo para funcionar permanentemente.
El deterioro de los materiales que componen nuestro robot puede ser debido al entorno donde se emplea u el envejecimiento de los elementos con el paso del tiempo.

6.4.7. Condiciones técnicas de los materiales

Los dispositivos empleados en este proyecto deberán cumplir todos y cada uno las normas descritas en el presente documento, asegurando su correcto funcionamiento mediante diferentes pruebas y ensayos. En el caso de no seguir las normas específicas o no realizarse dichas pruebas, el ingeniero proyectista quedara exento de responsabilidad en el posible deterioro o mal funcionamiento de los dispositivos durante su utilización.
En el caso de remplazar algún dispositivo por otro o alguno nueve, se deben tener las mismas características que los que reemplazan, inhibiéndose de cualquier responsabilidad por posibles fallos si estos requisitos no se cumplen.

6.4.8. Condiciones técnicas del material informático

Para poder ejecutar nuestro software diseñado se debe tener unos requisitos necesario por tal fin, que son los siguientes:

- ✓ Ordenador personal Pentium 4 con procesador a 2.0 GHz o equivalente.
- ✓ Un mínimum de 512 MB de memoria RAM.
- ✓ Un mínimum de 100MBde espacio en el disco duro para la instalación del software IDE Arduino
- ✓ Sistema operativo Windows XP, Mac OS, Linux.
- ✓ Un puerto USB libre para la conexión y alimentación de la tarjeta Arduino.

Para el desarrollo de este proyecto se ha utilizado la siguiente configuración de hardware y software:

- ✓ Ordenador personal con procesador Intel Pentium 2.13GHz, 4 GB de RAM con disco duro de 450GB.
- ✓ Sistema operativo Microsoft Windows 7 Home Premium.
- ✓ Microsoft Word 2007
- ✓ IDE de Arduino
- ✓ Paint
- ✓ Yed Graph Editor
- ✓ Fritzing
- ✓ Navegador de internet Google Chrome
- ✓ Adobe Acrobat Reader X

6.4.9. Clausulas sobre garantía y plazo de ejecución

Las cláusulas de garantía intentan proteger a las partes de posibles fallos u errores de manipulación del equipo diseñado, también se debe establecer un periodo de garantía de funcionamiento del equipo.

El cliente tiene derecho a utilizar para el desarrollo de este proyecto los materiales que cumplen las condiciones indicadas en el pliegue de condiciones sin necesidad de reconocimiento previo del proyectista, siempre y cuando se trate de materiales de procedencia reconocida y características similares.

El cliente no tendrá derecho a indemnización por causas de averías o perjuicios ocasionados en el desarrollo del proyecto.
EL contratista se hará cargo de todos los accidentes que puedan suceder, durante la instalación del equipo electrónico y cualquier avería o accidente.

6.4.10. Plazos de ejecución

Se indicará a continuación el plazo necesario en ejecutar este proyecto, que será muy reducido a la hora de fabricación en serie del producto.
Los retrasos debidos por causas ajenas a la voluntad de esta serán considerados como prorrogas.
En el caso de retraso injustificado sobre los plazos fijados, se puede imponer una multa sobre el presupuesto asignado como pago valorado.
Cualquier cuestión que pueda surgir sobre la interpretación u cumplimiento de las condiciones del contrato entre ambas partes serán resueltas por la comisión arbitral.

El plazo de ejecución del prototipo esta detallado en la tabla siguiente:

Actividad	Inicio	Fin	Días	Horas/Días	Total, Horas
Elección del proyecto	24/06/2017	29/06/2017	5	4	20
Diseño del proyecto	30/06/2017	24/07/2017	24	4	96
Programación	25/07/2017	06/08/2017	12	5	60
Ensayos	03/08/2017	12/08/2017	9	3	27
Montaje Robot	13/08/2017	14/08/2017	1	3	3
Documentación	14/08/2017	01/09/2017	18	5	90
		Total			296

TABLA 15: PLAZO EJECUCIÓN ROBOT MULTIFUNCIÓN

7. Diagrama de Gantt

Actividad	Inicio	Fin	Duracion
Eleccion del proyecto	24/06/2017	29/06/2017	5
Diseño del proyecto	30/06/2017	25/07/2017	25
Programacion	25/07/2017	06/08/2017	12
Ensayos	03/08/2017	12/08/2017	9
Montaje Robot	13/08/2017	14/08/2017	1
Documentacion	14/08/2017	01/09/2017	18

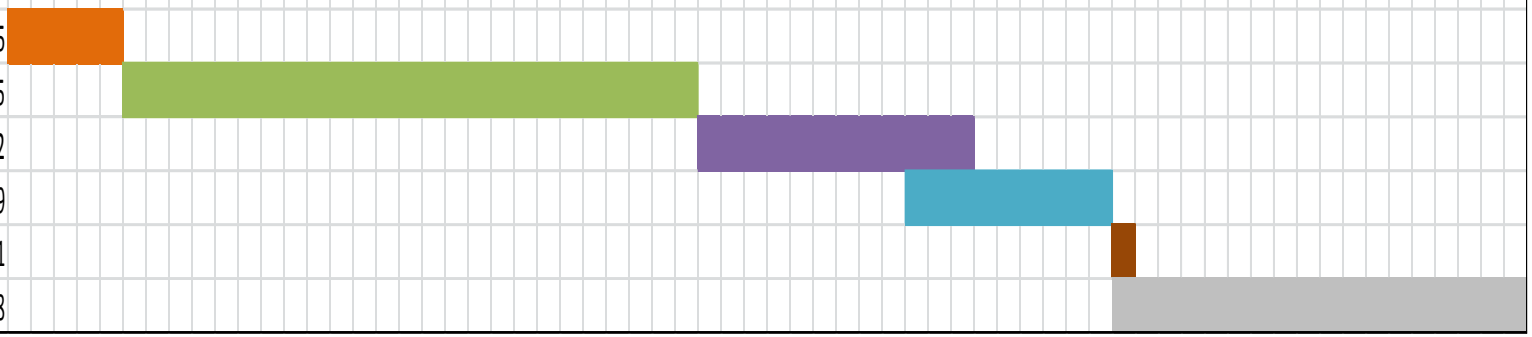


TABLA 16: DIAGRAMA DE GANT

8. PRESUPUESTO ECONOMICO

8.1. INTODUCCION

El presupuesto está divido en tres partes, los costes de la parte mecánica, la parte de hardware y finalmente los costes de desarrollo del proyecto, que incluye gastos originados por el diseño del proyecto, programación, costes por la elaboración del proyecto y los costes de ejecución del mismo.
La parte mecánica se compró en conjunto que incluye varios componentes incluidos.
La parte de hardware incluye todos los costes originados por la compra de los componentes necesarios para la ejecución del proyecto.
En Los costes de desarrollo del proyecto, se han tenido en cuenta los costes derivados de los honorarios del ingeniero proyectista.

8.2. PARTE MECANICA

En la parte mecánica se compró un chasis muy utilizado para hacer prototipos de robot móviles, influye en conjunto varios componentes que veamos en la tabla a continuación el precio incluye IVA.

Componente	Cantidad	Precio
Chasis base	1	
Rueda omnidireccional	1	
Ruedas 65mm x 20mm	2	
Soporte motores	2	
Tornillos M3 x 6	20	
Tornillos M3 x 10	2	
Tornillos M3 x 30	4	
Tuerca M3	10	
Separadores L10	10	
Total		21 €

TABLA 17 : PRESUPUESTO PARTE MECÁNICA

8.3. PARTE HARDWARE

Algunos componentes están comprados en internet y otros en tiendas, el precio están incluidos los portes y el IVA.

Componente	Precio unitario	Cantidad	Precio Total
Arduino Uno	19.82 €	1	19.82 €
Motores DC	2.2 €	2	4.40 €
Sensor TCRT500	2.95 €	3	8.85 €
Sensor HC-SR04	3.50 €	3	10.5 €
Bluetooth HC-06	4.99 €	1	4.99 €
Módulo L298N	3.95 €	1	3.95 €
Mini Breadboard	2.18 €	1	2.18 €
Cables macho-macho	3.99 €	1 (bolsa)	3.99 €
Cables macho-hembra	2.5 €	1 (bolsa)	2.50 €
Batería 9V 500mAh	2.2 €	1	2.2€
Batería 1.5V AA	0.9 €	4	3.6 €
Jack macho	0.30 €	1	0.3 €
interruptor	0.85 €	1	0.85 €
Porta-pilas	1	1	1.8 €
Soporte HC-SR04	1.2 €	3	3.6 €
Total			**73.53 €**

TABLA 18 : PRESUPUESTO PARTE HARDWARE

8.4. COSTE DE LA MANO DE OBRA

Concepto	Precio unitario (hora)	Cantidad horas	Precio total
Diseño del proyecto	20 h	96	1920 €
Programación	20 h	60	1200 €
Ensayos	20 h	27	540 €
Montaje Robot	20 h	3	60 €
Documentación	20 h	90	1800 €
Total			**5520 €**

TABLA 19 : PRESUPUESTO MANO DE OBRA

8.5. PRECIO TOTAL PROYECTO

Concepto	Coste
Coste parte mecánica	**21 €**
Coste parte hardware	**73.53 €**
Coste mano de obra	**5520 €**
TOTAL	**5614.53 €**

TABLA 20 : COSTE TOTAL PROYECTO

El coste total del proyecto es: 5614.53 €

9. ANEJO 1: ÍNDICE IMÁGENES

10. ANEJO 2: ÍNDICE ILUSTRACIONES

11. ANEJO 3: ÍNDICE TABLAS

12. ANEJO 4: INDICE FIGURAS

13. ANEJO 5: INDICE DIAGRAMAS DE BLOQUE

14. ANEJO 6: Adjuntos Digitales

En el CD adjunto con el proyecto se puede encontrar una carpeta llamada Anejos, en la que podemos encontrar los siguientes archivos y documentos:

- ✓ Manual de usuario Arduino
- ✓ Datasheet Micro-Atmel
- ✓ Manual de montaje del chasis del Robot
- ✓ Datasheet sensor TCRT500
- ✓ Datasheet transistor BC559B
- ✓ Datasheet integrado L293D
- ✓ Datasheet integrado L298N
- ✓ Datasheet sensor HC-SR04
- ✓ Datasheet modulo Bluetooth HC-06
- ✓ Diagrama de Gantt

15. Bibliografías

Periódicos y revistas

eldiario.es,Cristina Sanchez,2017

Noarduino,Wodpress, Abril26,2012

Manual Arduino

Manual App Inventor

Links internet

http://www.areatecnologia.com/electricidad/resistencia-electrica.html

https://hipertextual.com/2011/02/open-hardware, Geraldine Juárez 11/02/11

http://www.cortoc.com/2011/12/introduccion-arduino.html, Julio Roberto Letrán cardona

http://arduinodhtics.weebly.com/iquestqueacute-es.html

https://es.wikipedia.org/wiki/Conjunto_de_comandos_Hayes

https://forum.arduino.cc/

Descarga de software

http://Processing.org.co/

http://wiring.org.co/

https://www.arduino.cc/en/Main/Software

www.fritzing.org/download/

software online

https://create.arduino.cc/editor

www.ai2.appinventor.mit.edu.

Libros

Practical Arduino, Oxer Jonathan, Blemings Hugh,2010

Arduino programing notebook, ARDUMANIA,2011