Resumen

En este proyecto se diseñó el sistema de electrónica de control para robots con trabajo colaborativo, el sentido del giro de los motores (llantas) esto se logró con un puente-H full-Bridge L298n, se realizó el diseño, la elaboración y la implementación de este circuito, los sensores ultrasónicos, estos nos sirve para medir distancias, el sensor utilizado es el SRF02, que usa una comunicación I2C, por lo tanto se tuvo que diseñar una placa de comunicación I2C, para poder conectar todos los sensores y una brújula electrónica que maneja el mismo protocolo de comunicación pero con diferente voltaje, los sensores biométricos, nos sirven para medir gas(butano, metano, etc..), estos se conectan de manera analógica, todo es controlado por un arduino, en este caso se realizaron las pruebas con un arduino duemilanove.

Después de contar con todos los dispositivos anteriormente mencionados el móvil debe de ser capaz de esquivar obstáculo si lo requiere, esto quiere decir que ya tiene la inteligencia de avanzar, detenerse, retroceder, girar hacia la derecha o a la izquierda, según le convenga.

Estos datos obtenidos por los sensores, se mandan a MATLAB, de manera que se puede llevar un registro para los otros robots.

Palabras Claves: Robot, trabajo colaborativo, diseño, sensores, protocolos de comunicación, móvil.

1.- Introducción

En la actualidad los robots ya están de forma continua trabajando en ellos con mucho esmero y esfuerzo, trabajé en un móvil de trabajo colaborativo, esto quiere decir que trabajó en forma de enjambre, en vez de realizar un robot demasiado complejo que realiza varias actividades, se realizan robots más sencillos que realizan tareas determinadas.

Estos robots se comportaran de manera colectiva derivada de las interacciones entre robots con su entorno, inspirada en el comportamiento emergente en los insectos sociales.

El robot en el que se está trabajando debe de ser capaz de llegar a una posición que se le proporcione por medio de la programación, este debe de ser inteligente y autónomo, por ello cuenta con sensores que le permitirán tomar las decisiones.

1.1.- Justificación.

Actualmente, en el campo de la robótica, se está produciendo un gran desarrollo en el campo de los robots cooperativos (colaborativos). Algunos problemas son demasiado difíciles de resolver para un único robot: (empujar una caja, explorar un campo…).

La robótica colectiva (colaborativa) busca diseñar sistemas compuestos de varios robots capaces de resolver problemas conjuntamente. Los robots que forman parte de un sistema multirobot son simples en términos de diseño y control, y menos costosos que los sistemas de un solo robot especializado. Los sistemas multirobot están orientados a resolver problemas en los cuales la participación de un solo robot no es suficiente o resulta ser muy costosa, en términos de diseño y tiempo, como por ejemplo el transporte de objetos voluminosos, el manejo de material peligroso, la exploración y cobertura de terreno.

1.2.- Objetivo.

Diseñar un circuito electrónico que sea capaz de controlar el movimiento de un robot colaborativo (móvil), por medio de 4 motores microspeed, los movimientos hacia adelante, atrás, derecha e izquierda, cuando el robot lo requiera.

1.3.- Objetivos específicos.

*Diseñar y elaborar el driver necesario para los motores.

*Familiarización con la placa arduino y su entorno.

*Estudio, diseño y realización de la comunicación I2C, para los sensores ultrasónicos y la brújula electrónica.

*Caracterización de los sensores ultrasónicos.

*Montaje de los dispositivos electrónicos en el móvil.

1.4.- Alcances y limitaciones.

1.4.1.- Alcances.

Los alcances de esta residencia son de hacer funcionar los dispositivos electrónicos de control del móvil, estos son los motores con un par de puentes H lm298n, los sensores ultrasónicos estos por medio de la comunicación I2C, sensores biométricos, estos sensores miden gas (butano, metano, etc…), y con estos parámetros realizar rutinas para que avance, retroceda, gire ala derecha o a la izquierda. Todo es con respecto a la estructura que se tiene. Estos datos obtenidos de los sensores, tanto ultrasónicos como biométricos se capturan en MATLAB, para llevar un registro. La alimentación de todos los dispositivos, se realizó un circuito para alimentar los mismos, porque no todos requieren el mismo voltaje, se diseñó la placa, en base a un batería Li-PO de 11.1v a 2200 mAh.

1.4.2.- Limitaciones.

Las limitaciones del proyecto, fue hasta que se contó con el funcionamiento correcto de los sensores: ultrasónicos, biométricos; los actuadores en este caso son los motores, el funcionamiento de la brújula electrónica, realizando sus circuitos necesarios para el funcionamiento de cada uno.

1.5.- Hipótesis.

El diseñar la electrónica de control para robots de trabajo colaborativo nos facilitaría el control de los movimientos del móvil.

2.- Marco teórico.

2.1.- Motor Eléctrico.

Un motor eléctrico es una máquina eléctrica que transforma energía eléctrica en energía mecánica por medio de campos magnéticos variables electromagnéticas. Algunos de los motores eléctricos son reversibles, pueden transformar energía mecánica en energía eléctrica funcionando como generadores.

Principio de funcionamiento.

Los motores de corriente alterna y los de corriente continua se basan en el mismo principio de funcionamiento, el cual establece que si un conductor por el que circula una corriente eléctrica se encuentra dentro de la acción de un campo magnético, éste tiende a desplazarse perpendicularmente a las líneas de acción del campo magnético.

El conductor tiende a funcionar como un electroimán debido a la corriente eléctrica que circula por el mismo adquiriendo de esta manera propiedades magnéticas, que provocan, debido a la interacción con los polos ubicados en el estátor, el movimiento circular que se observa en el rotor del motor. Aprovechando el estator y rotor ambos de acero laminado al silicio se produce un campo magnético uniforme en el motor.

Partiendo del hecho de que cuando pasa corriente por un conductor produce un campo magnético, además si lo ponemos dentro de la acción de un campo magnético potente, el producto de la interacción de ambos campos magnéticos hace que el conductor tienda a desplazarse produciendo así la energía mecánica. Dicha energía es comunicada al exterior mediante un dispositivo llamado flecha.

Ventajas

En diversas circunstancias presenta muchas ventajas respecto a los motores de combustión:

- A igual potencia, su tamaño y peso son más reducidos.
- Se pueden construir de cualquier tamaño.
- Tiene un par de giro elevado y, según el tipo de motor, prácticamente constante.

- Su rendimiento es muy elevado (típicamente en torno al 75%, aumentando el mismo a medida que se incrementa la potencia de la máquina).
- Este tipo de motores no emite contaminantes, aunque en la generación de energía eléctrica de la mayoría de las redes de suministro sí emiten contaminantes.

Motor de corriente continua

En la figura número 1 se pueden observar tres motores de diferentes tamaños.

FIGURA No. 1) Motores de CC de varios tamaños.

El motor de corriente continua es una máquina que convierte la energía eléctrica continua en mecánica, provocando un movimiento rotatorio. En la actualidad existen nuevas aplicaciones con motores eléctricos que no producen movimiento rotatorio, sino que con algunas modificaciones, ejercen tracción sobre un riel. Estos motores se conocen como motores lineales.

Esta máquina de corriente continua es una de las más versátiles en la industria. Su fácil control de posición, paro y velocidad la han convertido en una de las mejores opciones en aplicaciones de control y automatización de procesos. Pero con la llegada de la electrónica su uso ha disminuido en gran medida, pues los motores de corriente alterna, del tipo asíncrono, pueden ser controlados de igual forma a precios más accesibles para el consumidor medio de la industria. A pesar de esto los motores de corriente continua se siguen utilizando en muchas aplicaciones de potencia (trenes y tranvías) o de precisión (máquinas, micromotores, etc.)

La principal característica del motor de corriente continua es la posibilidad de regular la velocidad desde vacío a plena carga.

Su principal inconveniente, el mantenimiento, muy caro.

Una máquina de corriente continua (generador o motor) se compone principalmente de dos partes, un estator que da soporte mecánico al aparato y tiene un hueco en el centro generalmente de forma cilíndrica. En el estator además se encuentran los polos, que pueden ser de imanes permanentes o devanados con hilo de cobre sobre núcleo de hierro. El rotor es generalmente de forma cilíndrica, también devanado y con núcleo, al que llega la corriente mediante dos escobillas.

También se construyen motores de CC con el rotor de imanes permanentes para aplicaciones especiales.

Sentido de giro.

El sentido de giro de un motor de corriente continua depende del sentido relativo de las corrientes circulantes por los devanados inductor e inducido.

La inversión del sentido de giro del motor de corriente continua se consigue invirtiendo el sentido del campo magnético o de la corriente del inducido.

Si se permuta la polaridad en ambos bobinados, el eje del motor gira en el mismo sentido.

Los cambios de polaridad de los bobinados, tanto en el inductor como en el inducido se realizarán en la caja de bornes de la máquina, y además el ciclo combinado producido por el rotor produce la fmm (fuerza magnetomotriz).

El sentido de giro lo podemos determinar con la regla de la mano derecha, la cual nos va a mostrar el sentido de la fuerza. La regla de la mano derecha es de la siguiente manera: el pulgar nos muestra hacia donde va la corriente, el dedo índice apunta en la dirección en la cual se dirige el flujo del campo magnético, y el dedo medio hacia donde va dirigida la fuerza resultante y por lo tanto el sentido de giro.

2.2.- Sistema arduino.

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (boot loader) que corre en la placa, en la figura No. 2 se observa un arduino duemilanove.

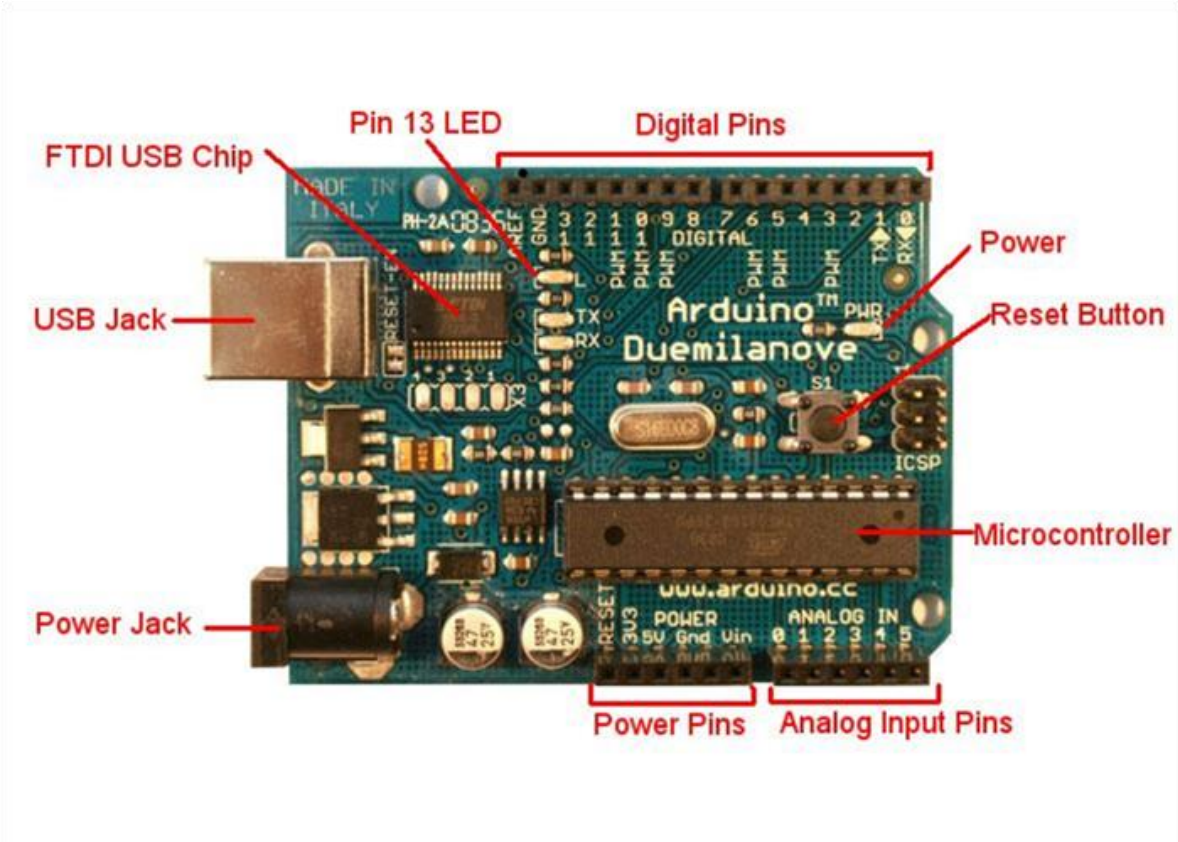


FIGURA No. 2) Arduino duemilanove.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

Al ser open-hardware, tanto su diseño como su distribución es libre. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.

Entradas y salidas

Consta de 14 entradas digitales configurables entrada i/o salidas que operan a 5 voltios. Cada pin puede proporcionar o recibir como máximo 40 mA. Los pines 3, 5, 6, 8, 10 y 11 pueden proporcionar una salida PWM (Pulse Width Modulation). Si se conecta cualquier cosa a los pines 0 y 1, eso interferirá con la comunicación USB. Diecimila también tiene 6 entradas analógicas que proporcionan una resolución de 10 bits. Por defecto miden de 0 voltios (masa) hasta 5 voltios, aunque es posible cambiar el nivel más alto, utilizando el pin Aref y algún código de bajo nivel.

Especificaciones

Los microcontroladores Arduino Diecimila, Arduino Duemilanove y Arduino Mega están basados en Atmega168, Atmega 328 y Atmega1280, como se muestra en la tabla número 1.

	Atmega168	Atmega328	Atmega1280
Voltaje operativo	5 V	5 V	5 V
Voltaje de entrada recomendado	7-12 V	7-12 V	7-12 V
Voltaje de entrada límite	6-20 V	6-20 V	6-20 V
Pines de entrada y salida digital	14 (6 proporcionan PWM)	14 (6 proporcionan PWM)	54 (14 proporcionan PWM)
Pines de entrada analógica	6	6	16

Intensidad de corriente	40 mA	40 mA	40 mA
Memoria Flash	16KB (2KB reservados para el bootloader)	32KB (2KB reservados para el bootloader)	128KB (4KB reservados para el bootloader)
SRAM	1 KB	2 KB	8 KB
EEPROM	512 bytes	1 KB	4 KB
Frecuencia de reloj	16 MHz	16 MHz	16 MHz

TABLA No.1) Especificaciones de arduinos.

Lenguaje de programación Arduino

La plataforma Arduino se programa mediante el uso de un lenguaje propio basado en el popular lenguaje de programación de alto nivel Processing. Sin embargo, es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino. Algunos ejemplos son:

- Java
- Flash (mediante ActionScript)
- Processing
- Pure Data
- MaxMSP (entorno gráfico de programación para aplicaciones musicales, de audio y multimedia)
- VVVV (síntesis de vídeo en tiempo real)
- Adobe Director
- Python
- Ruby
- C
- C++ (mediante libSerial o en Windows)
- C#
- Cocoa/Objective-C (para Mac OS X)
- Linux TTY (terminales de Linux)

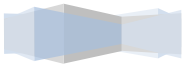
- 3DVIA Virtools (aplicaciones interactivas y de tiempo real)
- SuperCollider (síntesis de audio en tiempo real)
- Instant Reality (X3D)
- Liberlab (software de medición y experimentación)
- BlitzMax (con acceso restringido)
- Squeak (implementación libre de Smalltalk)
- Mathematica
- Matlab
- Minibloq (Entorno gráfico de programación, corre también en OLPC)
- Isadora (Interactividad audiovisual en tiempo real)
- Perl
- Visual Basic .NET
- VBScript
- Gambas

Esto es posible debido a que Arduino se comunica mediante la transmisión de datos en formato serie que es algo que la mayoría de los lenguajes anteriormente citados soportan. Para los que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida. Es bastante interesante tener la posibilidad de interactuar Arduino mediante esta gran variedad de sistemas y lenguajes puesto que dependiendo de cuales sean las necesidades del problema que vamos a resolver podremos aprovecharnos de la gran compatibilidad de comunicación que ofrece.

2.3.- Comunicación I2C.

El bus I2C, un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos con cierto nivel de "inteligencia", sólo requiere de dos líneas de señal y un común o masa. Fue diseñado a este efecto por Philips y permite el intercambio de información entre muchos dispositivos a una velocidad aceptable, de unos 100 Kbits por segundo, aunque hay casos especiales en los que el reloj llega hasta los 3,4 MHz.

La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca el tiempo (pulsos de reloj) y la otra se utiliza para intercambiar datos.



Descripción de las señales

SCL (System Clock) es la línea de los pulsos de reloj que sincronizan el sistema.

SDA (System Data) es la línea por la que se mueven los datos entre los dispositivos.

GND (Masa) común de la interconexión entre todos los dispositivos "enganchados" al bus.

Las líneas SDA y SCL son del tipo drenaje abierto, es decir, un estado similar al de colector abierto, pero asociadas a un transistor de efecto de campo (o FET). Se deben polarizar en estado alto (conectando a la alimentación por medio de resistores "pull-up") lo que define una estructura de bus que permite conectar en paralelo múltiples entradas y salidas, como se observa en la figura No. 3.

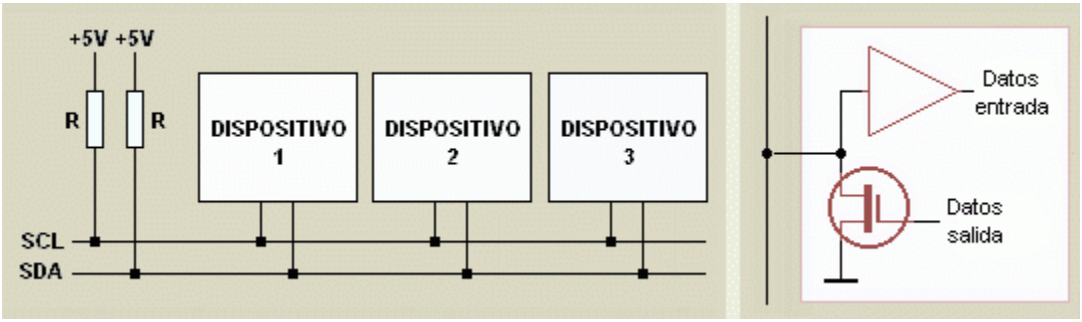


FIGURA No. 3) Conexión de la comunicación I2C.

El diagrama es suficientemente auto explicativo. Las dos líneas del bus están en un nivel lógico alto cuando están inactivas. En principio, el número de dispositivos que se puede conectar al bus no tiene límites, aunque hay que observar que la capacidad máxima sumada de todos los dispositivos no supere los 400 pF. El valor de los resistores de polarización no es muy crítico, y puede ir desde 1K8 (1.800 ohms) a 47K (47.000 ohms). Un valor menor de resistencia incrementa el consumo de los integrados pero disminuye la sensibilidad al ruido y mejora el tiempo de los flancos de subida y bajada de las señales. Los valores más comunes en uso son entre 1K8 y 10K.

Habiendo varios dispositivos conectados sobre el bus, es lógico que para establecer una comunicación a través de él se deba respetar un protocolo. Digamos, en primer lugar, lo más importante: existen dispositivos maestros y dispositivos esclavos. Sólo los dispositivos maestros pueden iniciar una comunicación.

La condición inicial, de bus libre, es cuando ambas señales están en estado lógico alto. En este estado cualquier dispositivo maestro puede ocuparlo, estableciendo la condición de inicio (start). Esta condición se presenta cuando un dispositivo maestro pone en estado bajo la línea de datos (SDA), pero dejando en alto la línea de reloj (SCL), como se observa en la figura No. 4.

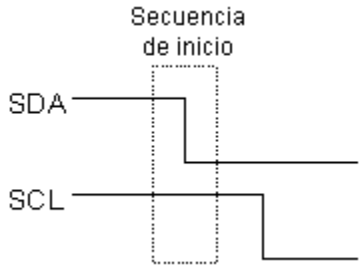


FIGURA No. 4) Secuencia de inicio de las líneas SDA y SCL.

El primer byte que se transmite luego de la condición de inicio contiene siete bits que componen la dirección del dispositivo que se desea seleccionar, y un octavo bit que corresponde a la operación que se quiere realizar con él (lectura o escritura).

Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits (A0-A6) está presente en el bus, éste contesta con un bit en bajo, ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro. Este bit de reconocimiento (ACK) en bajo le indica al dispositivo maestro que el esclavo reconoce la solicitud y está en condiciones de comunicarse. Aquí la comunicación se establece en firme y comienza el intercambio de información entre los dispositivos, como se observa en la figura No. 5.

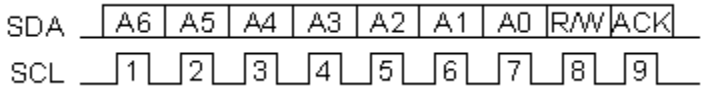


FIGURA No. 5) Intercambio de información entre dispositivos.

Si el bit de lectura/escritura (R/W) fue puesto en esta comunicación a nivel lógico bajo (escritura), el dispositivo maestro envía datos al dispositivo esclavo. Esto se mantiene

mientras continúe recibiendo señales de reconocimiento, y el contacto concluye cuando se hayan transmitido todos los datos, como se muestra en la figura No. 6.

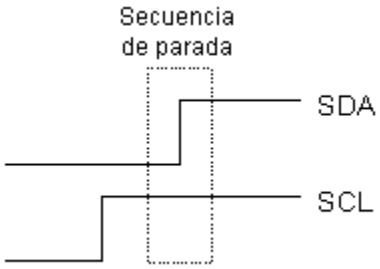


FIGURA No. 6) Finalización de la comunicación.

En el caso contrario, cuando el bit de lectura/escritura estaba a nivel lógico alto (lectura), el dispositivo maestro genera pulsos de reloj para que el dispositivo esclavo pueda enviar los datos. Luego de cada byte recibido el dispositivo maestro (quien está recibiendo los datos) genera un pulso de reconocimiento.

El dispositivo maestro puede dejar libre el bus generando una condición de parada (o detención; stop en inglés).

Si se desea seguir transmitiendo, el dispositivo maestro puede generar otra condición de inicio en lugar de una condición de parada. Esta nueva condición de inicio se denomina "inicio reiterado" y se puede emplear para direccionar un dispositivo esclavo diferente o para alterar el estado del bit de lectura/escritura.

Direccionamiento de dispositivos en el bus I2C

Lo más común en los dispositivos para el bus I2C es que utilicen direcciones de 7 bits, aunque existen dispositivos de 10 bits. Este último caso es raro.

Una dirección de 7 bits implica que se pueden poner hasta 128 dispositivos sobre un bus I2C, ya que un número de 7 bits puede ir desde 0 a 127. Cuando se envían las direcciones de 7 bit, de cualquier modo la transmisión es de 8 bits. El bit extra se utiliza para

informarle al dispositivo esclavo si el dispositivo maestro va a escribir o va a leer datos desde él. Si el bit de lectura/escritura (R/W) es cero, el dispositivo maestro está escribiendo en el esclavo. Si el bit es 1 el maestro está leyendo desde el esclavo. La dirección de 7 bit se coloca en los 7 bits más significativos del byte y el bit de lectura/escritura es el bit menos significativo.

El hecho de colocar la dirección de 7 bits en los 7 bits más significativos del byte produce confusiones entre quienes comienzan a trabajar con este bus. Si, por ejemplo, se desea escribir en la dirección 21 (hexadecimal), en realidad se debe enviar un 42, que es un 21 desplazado un bit hacia arriba. También se pueden tomar las direcciones del bus I2C como direcciones de 8 bit, en las que las pares son de sólo escritura y las impares son de sólo lectura. Para dar un ejemplo, el integrado de brújula magnética CMPS03 es fijado en fábrica en la dirección 0xC0 ($C0). La dirección 0xC0 se utiliza para escribir en él y la dirección 0xC1 es para leer de él.

2.4.- Sensores.

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, pH, etc. Una magnitud eléctrica puede ser una resistencia eléctrica (como en una RTD), una capacidad eléctrica (como en un sensor de humedad), una Tensión eléctrica (como en un termopar), una corriente eléctrica (como en un fototransistor), etc.

Un sensor diferencia de un transductor en que el sensor está siempre en contacto con la variable de instrumentación con lo que puede decirse también que es un dispositivo que aprovecha una de sus propiedades con el fin de adaptar la señal que mide para que la pueda interpretar otro dispositivo. Como por ejemplo el termómetro de mercurio que aprovecha la propiedad que posee el mercurio de dilatarse o contraerse por la acción de la temperatura. Un sensor también puede decirse que es un dispositivo que convierte una forma de energía en otra.

Características de un sensor

- Rango de medida: dominio en la magnitud medida en el que puede aplicarse el sensor.
- Precisión: es el error de medida máximo esperado.
- Offset o desviación de cero: valor de la variable de salida cuando la variable de entrada es nula. Si el rango de medida no llega a valores nulos de la variable de entrada, habitualmente se establece otro punto de referencia para definir el offset.
- Linealidad o correlación lineal.
- Sensibilidad de un sensor: suponiendo que es de entrada a salida y la variación de la magnitud de entrada.
- Resolución: mínima variación de la magnitud de entrada que puede apreciarse a la salida.
- Rapidez de respuesta: puede ser un tiempo fijo o depender de cuánto varíe la magnitud a medir. Depende de la capacidad del sistema para seguir las variaciones de la magnitud de entrada.
- Derivas: son otras magnitudes, aparte de la medida como magnitud de entrada, que influyen en la variable de salida. Por ejemplo, pueden ser condiciones ambientales, como la humedad, la temperatura u otras como el envejecimiento (oxidación, desgaste, etc.) del sensor.
- Repetibilidad: error esperado al repetir varias veces la misma medida.

Un sensor es un tipo de transductor que transforma la magnitud que se quiere medir o controlar, en otra, que facilita su medida. Pueden ser de indicación directa (e.g. un termómetro de mercurio) o pueden estar conectados a un indicador (posiblemente a través de un convertidor analógico a digital, un computador y un display) de modo que los valores detectados puedan ser leídos por un humano.

Por lo general, la señal de salida de estos sensores no es apta para su lectura directa y a veces tampoco para su procesado, por lo que se usa un circuito de acondicionamiento, como por ejemplo un puente de Wheatstone, amplificadores y filtros electrónicos que adaptan la señal a los niveles apropiados para el resto de los circuitos.

Resolución y precisión

La resolución de un sensor es el menor cambio en la magnitud de entrada que se aprecia en la magnitud de salida. Sin embargo, la precisión es el máximo error esperado en la medida.

La resolución puede ser de menor valor que la precisión. Por ejemplo, si al medir una distancia la resolución es de 0,01 mm, pero la precisión es de 1 mm, entonces pueden

apreciarse variaciones en la distancia medida de 0,01 mm, pero no puede asegurarse que haya un error de medición menor a 1 mm. En la mayoría de los casos este exceso de resolución conlleva a un exceso innecesario en el coste del sistema. No obstante, en estos sistemas, si el error en la medida sigue una distribución normal o similar, lo cual es frecuente en errores accidentales, es decir, no sistemáticos, la repetitividad podría ser de un valor inferior a la precisión.

Sin embargo, la precisión no puede ser de un valor inferior a la resolución, pues no puede asegurarse que el error en la medida sea menor a la mínima variación en la magnitud de entrada que puede observarse en la magnitud de salida.

2.4.1.- Sensores de gas (Químicos).

La función de estos sensores es dar lugar a una magnitud física (conductancia, resistencia,...) la cual pueda ser capturada por el hardware de adquisición. Dicha magnitud debería reflejar en menor o mayor la exposición de los sensores a la muestra olorosa.

El funcionamiento de estos sensores es básicamente el siguiente: tras ser expuestos los sensores a un determinado gas o mezcla de ellos la magnitud física antes mencionada se ve alterada en una manera teóricamente diferente según la sustancia a la que se expone. En el caso más simplificado en el que sólo se emplee un sensor, éste debería sufrir una variación de magnitud tal que ésta fuese característica de la sustancia a la que se expone.

A continuación se describirán los principales tipos de sensores químicos utilizados en estos dispositivos.

Tipos de sensores químicos.

Los tipos de sensores más ampliamente utilizados son cuatro: basados en semiconductor de óxido metálico (Metal-Oxide Semiconductor), basados en onda acústica de superficie (Surface Acoustic Wave, SAW), ópticos, basados en fotoionización y los basados en resistencia (Chemiresistors).

* Basados en **semiconductor de óxido metálico**, estos sensores están formados por una fina lámina de semiconductor de cierto óxido metálico. Tras la exposición tiene lugar un cambio en la conductancia del material y esto es el lo que se utiliza para caracterizar la sustancia olorosa. Estos sensores son comercialmente accesibles y tienen buena sensibilidad pero para su correcto funcionamiento deben operar a temperaturas entre

100 °C y 600 °C lo cual hace que consuman más potencia que aquellos que pueden funcionar a temperatura ambiente siendo difícilmente adaptables a dispositivos portátiles por razones obvias.

* Basados en **onda acústica de superficie**, estos sensores hacen uso de las ondas acústicas conocidas como ondas Rayleigh en honor de su descubridor. El funcionamiento es el siguiente: estos sensores están formados por un material piezoeléctrico (normalmente un cuarzo) el cual se recubre con una delgada capa de un material (en la mayoría de los casos se usa un polímero) que reacciona en contacto con ciertos gases, dicha estructura es excitada mediante señales de radiofrecuencia las cuales varían su frecuencia inicial de excitación tras la aparición de las mencionadas ondas de superficie las cuales se inducen en la estructura cuando ésta entra en contacto con la sustancia olorosa objetivo. Las ventajas de este tipo de sensores son su alta sensibilidad y que pueden ser producidos en masa con alta reproducibilidad (es decir, se puede fabricar una cantidad elevada de los mismos y su comportamiento es parecido con cierta tolerancia). Sin embargo, dado que han de excitarse con radiofrecuencia el aumento de la miniaturización puede ser un problema a la hora de aplicar dicha excitación.

Sensores de óxido metálico.

Los sensores fabricados con materiales semiconductores adquirieron una considerable popularidad a finales de los 80. Su aparición permitió ofrecer la posibilidad de adquirir un detector de gas universal y de bajo coste. De igual forma que los sensores catalíticos, funcionan gracias a la absorción de gas en la superficie de un óxido caliente. De hecho, es una fina película de óxido de metal (generalmente los óxidos de los metales de transición o metales pesados, como el estaño) que se deposita en un trozo de silicona, un proceso muy parecido al usado en la fabricación de los “chips” de ordenadores. La absorción de la muestra de gas en la superficie con óxido, y su posterior oxidación catalítica, da como resultado un cambio de la resistencia eléctrica del material con óxido y se puede relacionar con la concentración de la muestra de gas. La superficie del sensor se calienta a una temperatura constante de alrededor de 200-250°C, para acelerar el ritmo de reacción y para reducir los efectos de los cambios de la temperatura ambiente.

El principio de operación de los sensores oxido metálico está basado en el cambio de en la conducción del óxido en interacción con un gas y el cambio es usualmente proporcional a la concentración de gas. Hay dos tipos de sensores oxido metálico; el tipo n (óxido de zinc, dióxido de estaño, dióxido de titanio y óxido de hierro) con respuesta a la reducción de gases y el tipo p (oxido de níquel, oxido de cobalto) con respuesta a oxidación de gases. El sensor tipo n, su operación es la siguiente: oxígeno en el aire reacciona con la superficie del sensor y atrapa cualquier electrón libre en la superficie o en los límites de la lámina de óxido metálico. Esto produce una resistencia alta en estas áreas debido a la falta de

transportadores, y el resultado de la diferencia de potencial entre la inhibición (bloqueo) de la lámina y la movilidad de los transportadores. Si el sensor es introducido a una reducción de gas como H_2,CH_4, CO, C_2H_5, o H_2S la resistencia baja porque el gas reacciona con el oxígeno y libera un electrón.

Esto disminuye la barrera de potencial y permite que los electrones fluyan, así incrementa la conductividad. Los sensores de tipo p responden a la oxidación de gases como O_2, NO_2 y Cl_2 como estos gases remueven electrones y crean hoyos.

Produciendo portadores de carga. La ecuación 1 y 2 describe la reacción ocurrida en la superficie.

$\frac{1}{2}O_2 + e^- \rightarrow O^-(s)$ Ecuación (1)

$R(g) + O^-(s) \rightarrow RO(g) + e$ Ecuación (2)

Donde e es un electrón del óxido. R(g) es la reducción de gas, y g y s son la superficie y el gas, respectivamente.

La sensibilidad del sensor de gas de óxido (ΔR/R_b)/c (gas), es calculado ΔR = R - R_b para la oxidación de gases y ΔR = R_b – R para reducción de gases, donde R_b es la resistencia base y R es la resistencia expuesta a el gas/olor y c (gas) es la concentración del gas.

La sensibilidad del sensor óxido metálico depende del espesor de la película y la temperatura ala que el sensor está operando, si la película es más delgada llega a ser más sensible a los gases.

La sensibilidad de los sensores puede ser mejorada agregándole un metal catalítico al óxido; un dopaje en exceso puede reducir la sensibilidad.

El efecto de dopar el sensor de SnO_2 con Cu, por ejemplo, demuestra significantemente el incremento de la sensibilidad. El tamaño de la lámina del óxido también afecta la sensibilidad y selección particular del gas como los límites de la lámina actúa como distribución centrales de los electrones.

El radio del tamaño de la lámina (D) a la merma del electrón por el espesor de la lámina (L), se muestra a continuación D/2L = 1

2.4.2.- Sensor ultrasónico SRF02.

El sensor ultrasónico SRF02 (Figura No. 7) proporciona un rango largo dentro de los sensores ultrasónicos, de 15 cm hasta los 600 cm. Es de uso sencillo, ya que cuenta con dos tipos protocolos de comunicación, Serial e I2C.

Figura No. 7) Sensor ultrasónico SRF02

El sensor trabaja transmitiendo un sonar ultrasónico (por encima del rango audible para los humanos), y proporciona una salida en tres medidas diferentes, que son:

- Pulgadas.
- Centímetros.
- Microsegundos.

Las primeras dos medidas (Pulgadas y Centímetros) es la distancia que existe entre el objeto y el sensor.

La tercer medida (Microsegundos) es el tiempo en el que regresa el sonar (Eco producido por el tren de pulso enviado) a el sensor.

En la tabla No. 2 se puede observar las características.

Alimentación.	5 V	Comunicación.	Serial/ I2C
Corriente.	4 mA	Angulo.	45 grados
Rango.	15 cm a 600 cm*	Tiempo que oscila.	70 mS

TABLA No. 2) Características del sensor ultrasónico SRF02.

* El rango mínimo varía según la temperatura, en un día caluroso el rango mínimo puede ser 17- 18 cm que es caso de la ciudad de Tuxtla Gutiérrez, en un clima frío es de 15- 16 cm.

En la figura No. 8 se observa las dimensiones con las que cuenta el sensor SRF02.

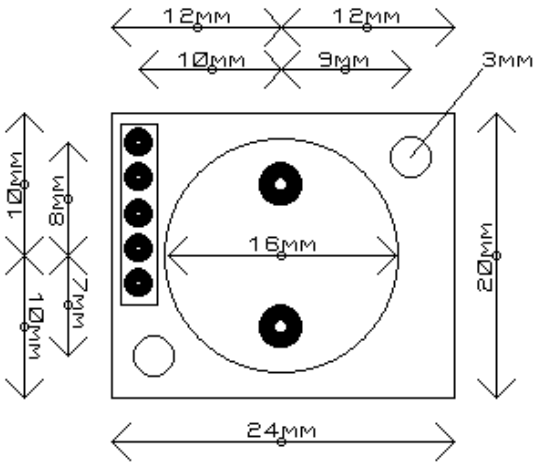


TABLA No. 8) Dimensiones del sensor ultrasónico SRF02

Teoría de operación.

El sensor detecta objetos por la emisión de una ráfaga por pequeños sonares ultrasónicos y luego escucha el eco.

Bajo el control de un microcontrolador anfitrión (Pulso de disparo), el sensor emite una pequeña ráfaga (sonar ultrasónico) de 40 KHZ. Esta ráfaga viaja a través del aire cerca de 1130 ft por segundo, golpea un objeto y luego rebota hacia el sensor .E sensor proporciona un pulso de salida al microcontrolador anfitrión que terminara cuando el eco es detectado, por lo tanto el ancho de este pulso corresponde a la distancia a la que se encuentra el objeto.

Los sensores de proximidad ultrasónicos son dispositivos fácilmente aplicables que permiten medir cualquier objeto que se encuentre perpendicular a su haz. Aunque su precisión puede ser afectada por muchos factores, en algunos casos se trata de la elección

de la construcción. La razón principal es que, aparte de los sensores ópticos los sensores ultrasónicos pueden funcionar en diversos entornos, independientemente de la absorción de la luz del objeto medido.

La principal ventaja de metros de distancia por ultrasonidos son los siguientes:

*Medición sin contacto en entornos ruidosos, con polvo, niebla.

* El rendimiento no depende del color o tipo del material.

Principio de medición del sensor ultrasónico.

En la figura No. 9 se observa el ancho del haz del sensor.

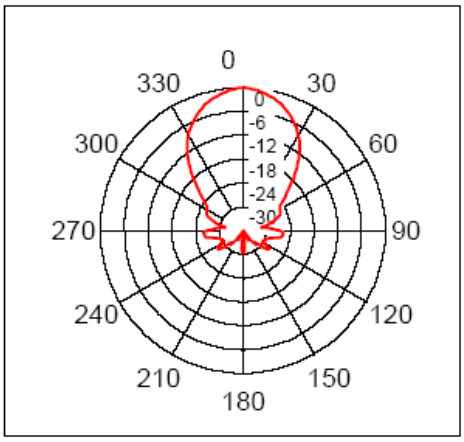


FIGURA No. 9) Espectro de detección.

Los fabricantes muestran el patrón del haz, mostrando la sensibilidad del transductor en db.

En la figura No. 10 se observa el patrón del haz para el SRF02, mostrando la máxima detección de un encapsulado de 55mm de diámetro.

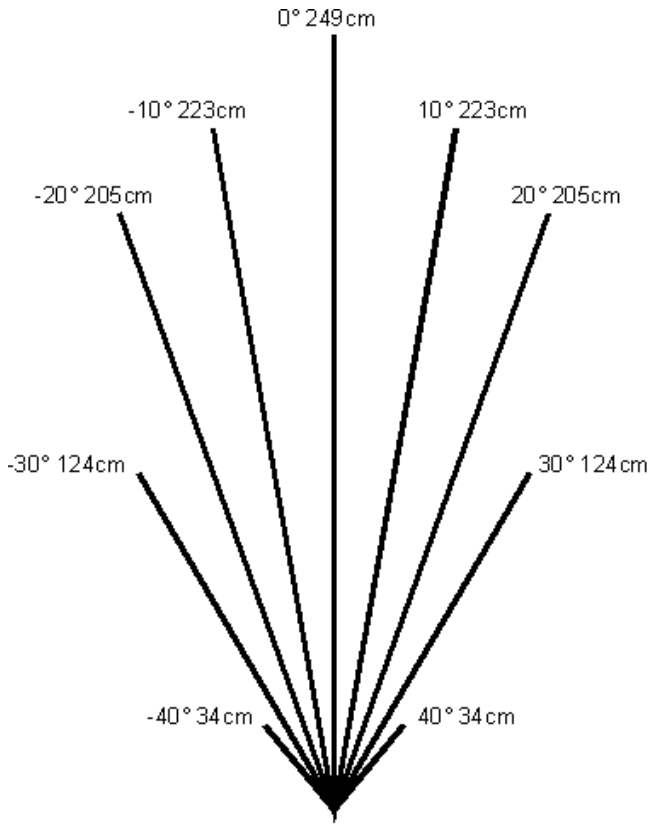


FIGURA No. 10) Espectro de detección.

Ecuación del sensor.

La ecuación del sensor esta en relación a tres variables, que son:

- t, es el tiempo en el que tarda en regresar el sonar emitido por el pulso.
- d, es la distancia a la que se encuentra el objeto, con el que se produce el eco.
- v, es la velocidad a la que viaja la el sonar.

El sensor SRF02 nos proporciona el tiempo y la distancia, con estos podemos calcular a la velocidad aproximada a la que viaja el sonar.

La ecuación para poder calcular la velocidad del sonar es la siguiente:

$$v = \frac{2*d}{t} \qquad \text{Ecuación (3)}$$

La distancia se debe multiplicar por dos ya que el tiempo que tenemos es el que tarda en ir y regresar del sonar.

Después de realizar múltiples lecturas, nos da que la velocidad a la que viaja el sonar es de 1130 ft/s.

Para poder calcular la distancia es la siguiente ecuación:

$$d = \frac{v*t*\cos\beta}{2} \qquad \text{Ecuación (4)}$$

Dónde:

d = distancia final.

v = velocidad del sonar.

t = tiempo en el que transcurre el sonar.

β = Angulo de incidencia.

Conexión I2C.

Para seleccionar el modo I2C deberá dejar sin conectar el Pin Modo del SRF02. El Pin SDA corresponde a la señal de datos y el Pin SCL a la señal de reloj. Ambas señales se deben polarizar a +5Vcc a través de dos resistencias de polarización positiva que normalmente se encuentran en el circuito maestro del bus I2C que controla los dispositivos I2C esclavos. Esto quiere decir que solo son necesarias dos resistencias en todo el bus, no dos por cada circuito.

La dirección I2C del medidor SRF02 por defecto es 0xE0 pero puede elegir cualquiera de las otras 16 siguientes para conectar otros sensores: 0xE0, 0xE2, 0xE4, 0xE6, 0xE8, 0xEA, 0xECC, 0xEE, 0xF0, 0xF4, 0xF6, 0xF8, 0xFA, 0xFC, 0xFE.

En la figura 11 se observa el aspecto físico que tiene el sensor SRF02, y los pines con lo que cuenta.

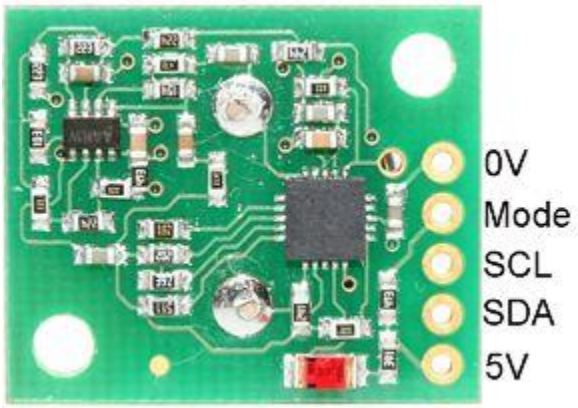


FIGURA No. 11) Pines del sensor SRF02.

Registros.

El SRF02 está compuesto por un juego de 6 registros, como se puede observar en la tabla No. 3:

Registros Nº	Modo de lectura	Modo de Escritura
0	Revisión de software interno	Registros de comandos
1	No usado (se lee 0x80)	No disponible
2	Byte alto de la medida realizada	No disponible
3	Byte bajo de la medida realizada	No disponible
4	Byte alto del valor mínimo de distancia	No disponible
5	Byte bajo del valor mínimo de distancia	No disponible

TABLA No. 3) Registros del sensor SRF02.

El único registro que se puede escribir es el 0, ya que este es que el que se utiliza para empezar un nuevo cálculo. Las medidas tardan unos 65mS en llevarse a cabo y mientras se realizan no responden a ninguna otra operación que se realice mediante el bus I2C. Si se intenta leer este registro se obtiene la versión del firmware interno del sonar SRF02.

Los registros 2 y 3 son el resultado de la última medida realizada en un valor de 16 bits medidos en pulgadas, centímetros o microsegundos según el comando que se haya utilizado. Un valor de 0 indica que el sensor no ha detectado ningún objeto. Recuerde no

iniciar ningún cálculo de medición antes de 65 mS del anterior para dar tiempo a desvanecerse a la primera ráfaga de ultrasonidos.

Los registros 4 y 5 son el resultado de el valor aproximado de la distancia mínima que el sonar puede medir en un valor de 16 bits.

Comandos.

Los 3 primeros comandos del (80 al 82) se utilizan para iniciar una nueva medición en pulgadas, centímetros o microsegundos.

Los otros 3 siguientes del (86 al 88) son parecidos pero no transmiten ninguna ráfaga por lo que no miden la distancia a ningún objeto. Estos comandos le pueden servir para detectar las ráfagas de otros medidores ultrasónicos.

El comando 92 se utiliza para transmitir una ráfaga pero no realiza ninguna medida.

El comando 96 reinicia el SRF02 realizando un ciclo de auto ajuste. Es como si se conectara la alimentación.

Los tres últimos (160,165 y 170) son los encargados de cambiar la dirección I2C de los SRF02. Los comandos se puede observar en la tabla No. 4.

Comandos		Descripción
Decimal	Hexadecimal	
80	0x50	Iniciar una nueva medición real. Resultado en pulgadas
81	0x51	Iniciar una nueva medición real. Resultado en centímetros
82	0x52	Iniciar una nueva medición real. Resultado en microsegundos
86	0x56	Iniciar una nueva medida falsa. Resultado en pulgadas
87	0x57	Iniciar una nueva medida falsa. Resultado en centímetros
88	0x58	Iniciar una nueva medida falsa. Resultado en microsegundos
92	0x5C	Transmite una ráfaga de 8 ciclos de 40khz- no hace cálculos de medición
96	0x60	Fuerza un reinicio del sonar SRF02 realizando un

		ciclo de autoajuste.
160	0xA0	1º comando de la secuencia para cambiar la dirección I2C
165	0xA5	3º comando de la secuencia para cambiar la dirección I2C
170	0xAA	2º comando de la secuencia para cambiar la dirección I2C

TABLA No. 4) Comandos del sensor SRF02.

Realizando un cálculo de medición.

Para realizar un cálculo de medición se debe escribir un comando de los de arriba sobre el registro de comandos y esperar un tiempo (aproximadamente 66 mS) para poder leer los registros 2 y 3 y así obtener el cálculo completo.

Comprobación para cálculos de medición completos.

Mientras el SRF02 está en mitad de un cálculo no responderá a ninguna orden sobre el bus I2C. Tan pronto como se completa el cálculo de medición el SRF02 responderá de nuevo al bus I2C pudiendo leer el resultado. Deberá esperar alrededor de unos 70mS para poder volver a realizar una nueva medición. Si no quiere esperar puede intentar leer el registro 0x00. Si el SRF02 está ocupado, el bus I2C estará inactivo con la línea de datos a (1) y un valor de 0xFF. Así pues si lee un valor distinto de 0xFF, es porque el SRF02 no está ya haciendo el cálculo y ha respondido devolviendo el nº de la versión del firmware interno del propio SRF02.

Cambiando las dirección del bus I2C.

Para cambiar la dirección I2C del SRF02 es necesario que tenga conectado solamente un circuito en el bus. Se debe escribir la secuencia de los 3 comandos en el orden correcto seguido de la nueva dirección que se le quiere poner. Por ejemplo para cambiar la dirección por defecto de (0xE0) a la (0xF2) debe escribir la siguiente dirección 0xE0;

(0xA0, 0xAA, 0xA5, 0xF2).

Estos comandos deben ser enviados en la secuencia correcta evitando además mandar ningún otro comando en mitad de la secuencia. Esta secuencia debe ser enviada a el registro de comandos 0 en cuatro ciclos de escrituras diferentes y consecutivos. Pasando esta a ser la dirección actual mientras que no se cambie a otra nueva. Deberá etiquetar el

sonar con su dirección pero si se le olvida este solo encienda este sin enviar ningún comando ya que el SRF02 emitirá su dirección en unos destellos en el led.

Dirección		Destello largo	Destello corto
Decimal	Hexadecimal		
224	E0	1	0
226	E2	1	1
228	E4	1	2
230	E6	1	3
232	E8	1	4
234	EA	1	5
236	EC	1	6
238	EE	1	7
240	F0	1	8
242	F2	1	9
244	F4	1	10
246	F6	1	11
248	F8	1	12
250	FA	1	13
252	FC	1	14
254	FE	1	15

TABLA No. 5) Direcciones del sensor SRF02.

Nota- Hay solo una dirección de modulo almacenada en el SRF02. Si la cambia, la dirección equivalente del modo serie cambiará:

2.5.- Matlab.

MATLAB es un entorno de programación para el desarrollo de algoritmos, análisis de datos, visualización y cálculo numérico. Usando MATLAB, puede resolver los problemas técnicos de computación más rápido que con los lenguajes de programación tradicionales, tales como C, C + + y Fortran, en la figura No. 12 se observa el logo del programa MATLAB.

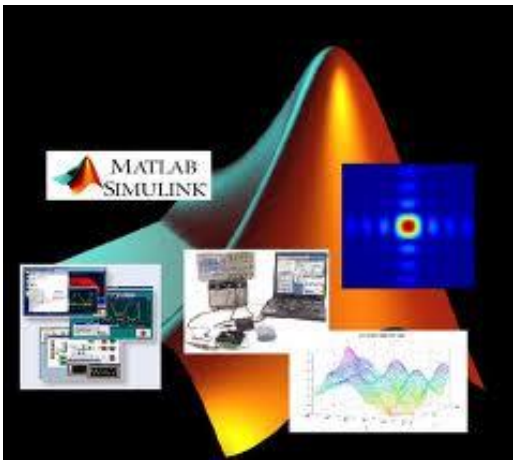


FIGURA No. 12) Logotipo de matlab.

Puede usar MATLAB en una amplia gama de aplicaciones, incluyendo la señal y el procesamiento de imágenes, comunicaciones, diseño de control, prueba y medición, modelado y análisis financiero y biología computacional.

2.5.1.- Interfaz gráfica Matlab (GUI).

En la figura No. 13 se muestrea la ventana para entrar ala GUI de MATLAB.

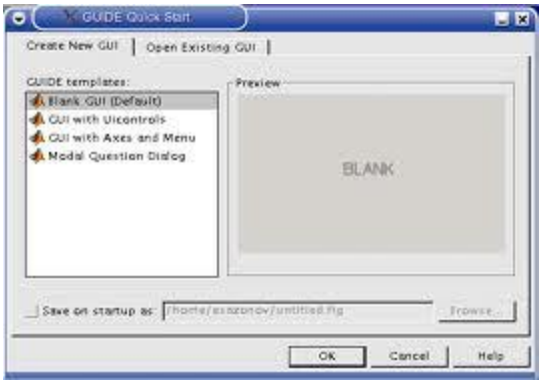


FIGURA No. 13) GUI de MATLAB.

La interfaz gráfica de usuario, conocida también como GUI (del inglés graphical user interface) es un programa Informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones

disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

Habitualmente las acciones se realizan mediante manipulación directa, para facilitar la interacción del usuario con la computadora. Surge como evolución de las interfaces de línea de comandos que se usaban para operar los primeros sistemas operativos y es pieza fundamental en un entorno gráfico. Como ejemplos de interfaz gráfica de usuario, cabe citar los entornos de escritorio Windows, el X-Window de GNU/Linux o el de Mac OS X, Aqua.

En el contexto del proceso de interacción persona-ordenador, la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

GUIDE es un entorno de programación visual disponible en MATLAB para realizar y ejecutar programas que necesiten ingreso continuo de datos.

Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++.

3.- Metodología.

Actividades realizadas.

1.- **Se investigó y recopiló información de robots de trabajo colaborativo:** se recabo información de diferentes sistemas de robots de trabajo colaborativo verificando trabajos similares.

2.- **Caracterización de los sensores:** se recopiló la información y se clasificó, separando la de utilidad, posteriormente se realizaron pruebas, hasta obtener el correcto funcionamiento de los sensores.

3.- **Diseño del control de los motores:** se recabó información acerca de los drivers que podrían ser útiles, se seleccionó uno por las características que nos proporcionó, posteriormente se realizaron pruebas para poder realizar el diseño final.

4.- **Alimentación del sistema:** se calculó el consumo de corriente y voltaje que requería el sistema para poder realizar el sistema de alimentación del sistema del móvil.

5.- **Interface:** se diseñó un interface en MATLAB que nos sirve para poder guardar los datos obtenidos de los sensores.

6.- **Diseño de placas de sensores:** se realizaron las placas correspondientes a los sensores, tanto a los de comunicación analógica como a los de comunicación I2C.

7.- **Pruebas:** se probaron todas las partes anteriormente mencionadas de manera conjunta, para observar el funcionamiento del sistema.

3.1.- Desarrollo del diseño de electrónica de control.

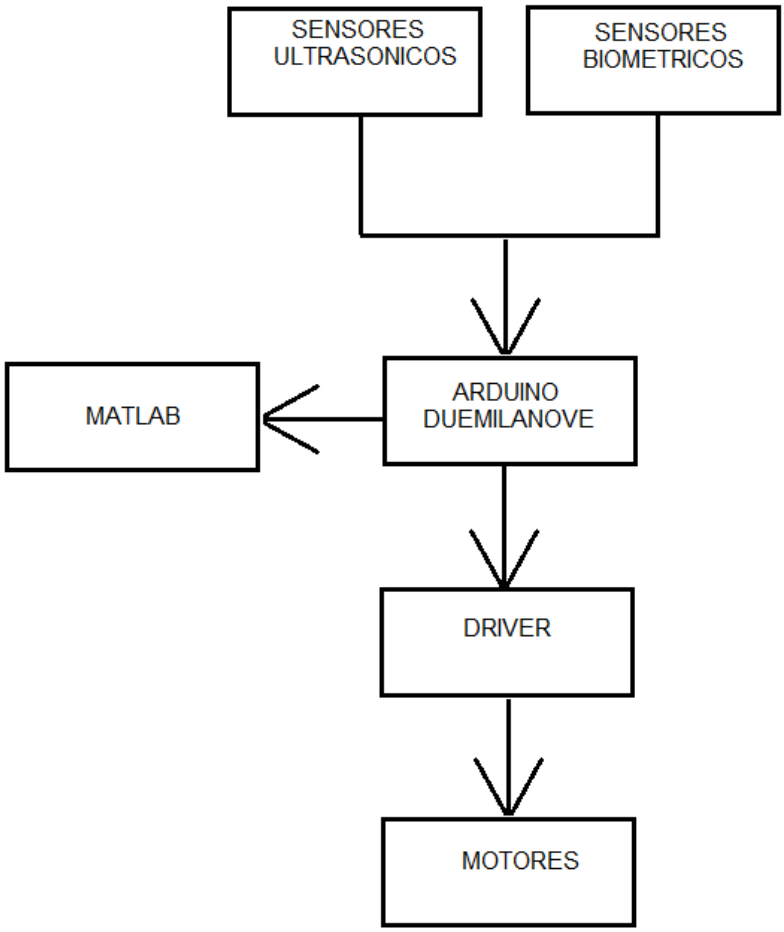


FIGURA No. 14) Diagrama a bloques del funcionamiento del sistema.

Motores.

En la figura No. 15 se observa el móvil y los motores para el que se realizó el diseño.

FIGURA No. 15) Imagen del móvil y los motores.

Estos motores nos permiten el movimiento del móvil, el móvil cuenta con 4 motores, uno por llanta, se realizaron pruebas de los motores, para ver el consumo de corriente y de voltaje ya que no contamos con las especificaciones de los mismos.

En las pruebas nos arrojaron los siguientes resultados, que se observan en la tabla No. 6:

Voltaje	Estado	Consumo de corriente
5v	Sin esfuerzo	300 mA
5v	Con esfuerzo	700 mA

TABLA No. 6) Resultado de pruebas.

Los motores se colocaran de tal manera que nos permita los giros hacia la derecha, izquierda, adelante y atrás, para eso se los agrupó en dos pares, los del lado y los del lado izquierdo, los de lado derecho van a ser controlados por una señal y los de lado izquierdo por otra, entonces sí quiero que gire:

A) Gire hacia adelante.
   Los cuatro motores deben de girar hacia adelante.

B) Gire hacia la derecha.
Los motores del lado izquierdo deben de girar hacia adelante y los del lado izquierdo deben de estar detenidos.

C) Gire hacia la izquierda.
Los motores del lado derecho deben de girar hacia adelante y los del lado derecho deben de estar detenidos.

D) Gire hacia atrás.
Los cuatro motores deben de girar hacia atrás.

Con este arreglo de los motores se puede obtener los movimientos necesarios para que el carro sea capaz de realizar los esquivos de obstáculos.

Driver, Puente h.

El control de los motores se realizó con un punteh-l298b, es el que se observa en la figura No. 16.

FIGURA No. 16) Circuito integrado L298N (Puente H).

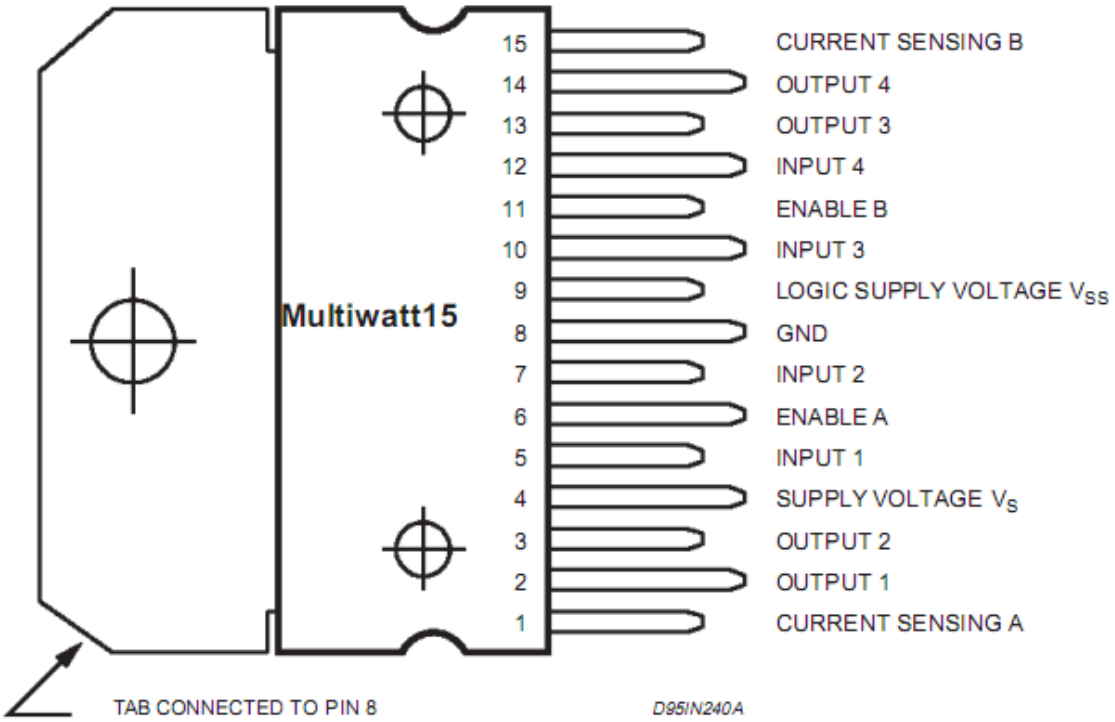


FIGURA No. 16) Pines del puente h l298.

Como se puede observar en la figura No. 16 el puente h-l298b cuenta con 4 entradas, 4 salidas, 2 habilitaciones, dos censados de corriente, la salida total e corriente es de 4 amperes, esto quiere decir que nos puede entregar hasta 1 ampere por salida. El puente h cuenta con dos configuraciones, half-bridge y full-bridge, si la configuración es full-bridge se cuenta con dos puentes y si es half-bridge se cuenta con cuatro, la que usamos en este caso es la de full-bridge, que es la configuración que se muestra en la figura VIII.II.II.III.

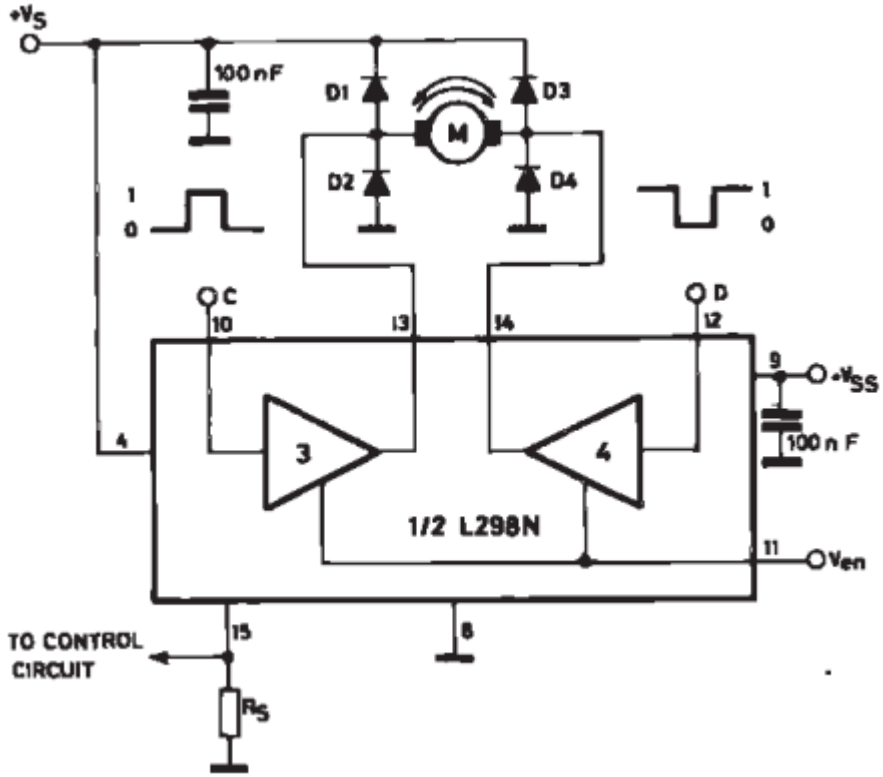


FIGURA No. 17) Configuración para el control bidireccional de un motor de cd.

El funcionamiento de la configuración de la figura No. 17 es para un motor de cd para que tenga movimientos en ambos sentidos, donde vs y vss están conectados a 5v.

El funcionamiento del driver se presenta en la tabla No. 6 de verdad:

Entrada 1	Entrada 2	Salida 1	Salida 2	Dirección
0	0	0	0	Paro
0	1	0	1	Atrás
1	0	1	0	Adelante
1	1	0	0	Paro

TABLA No. 7) Funcionamiento del driver.

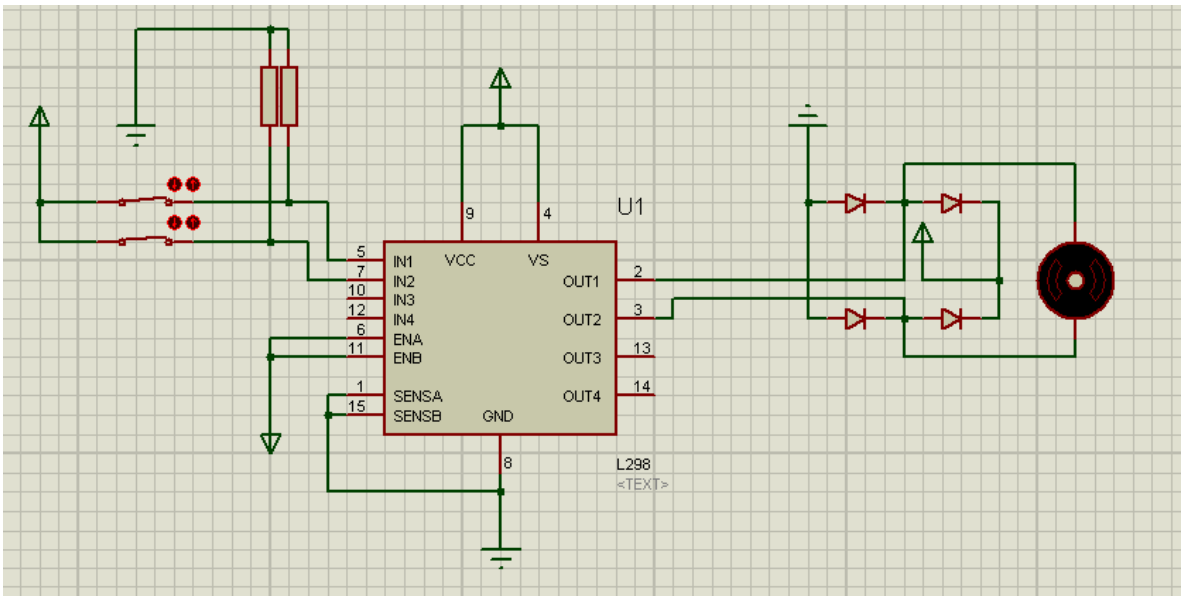


FIGURA No. 18) simulación en ISIS 7 Profesional.

En la figura No. 18 se observa la simulación del puente h-l298b, con sus entradas (1 y 2) en 0 lógico (Tierra), se observa que el motor no gira en ninguna dirección.

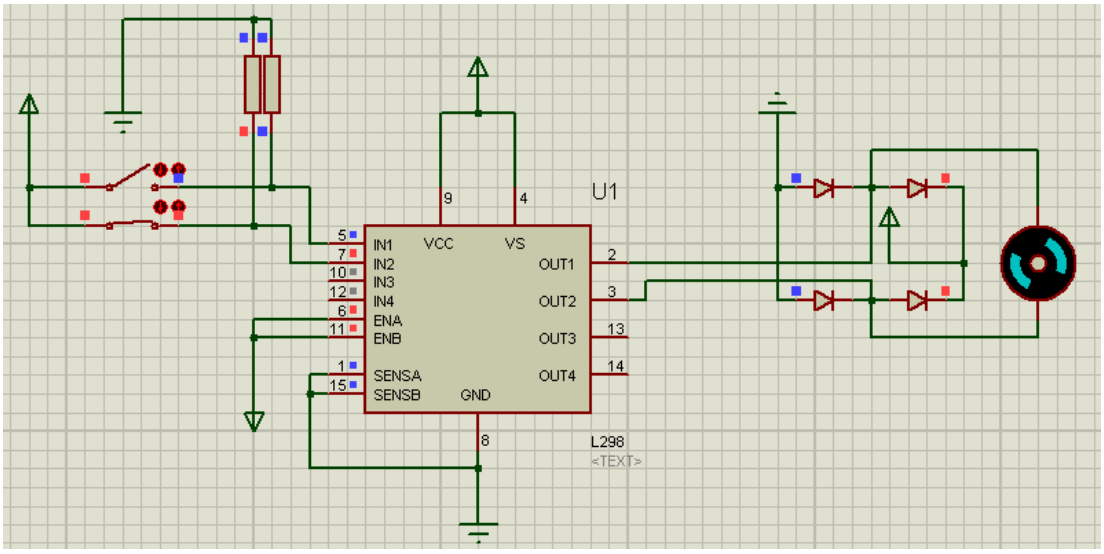


FIGURA No. 19) simulación en ISIS 7 Profesional.

En la figura No. 19 se observa que la entrada 1 esta en nivel alto, el giro del motor es hacia adelante.

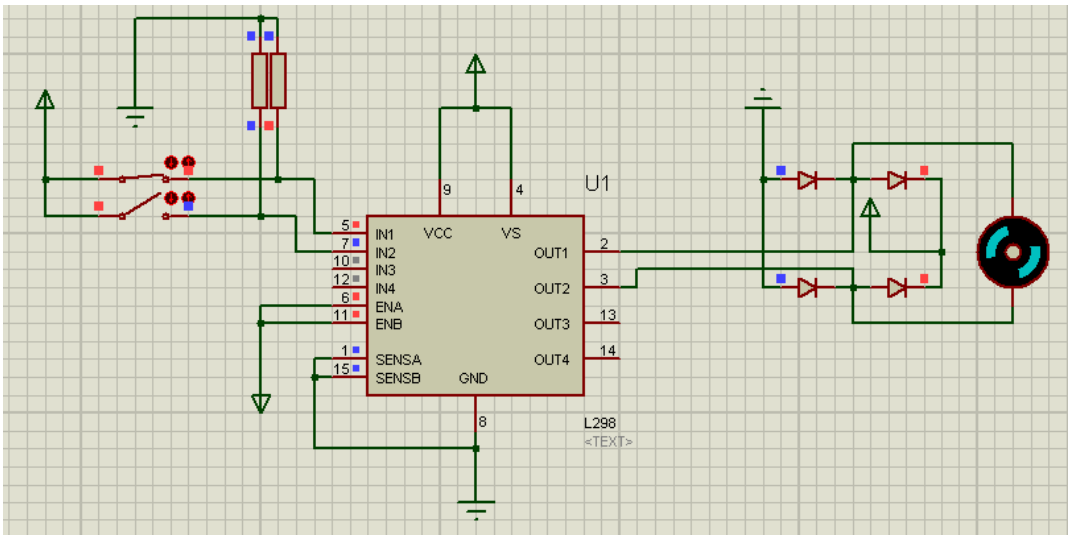


FIGURA No. 20) simulación en ISIS 7 Profesional.

En la figura No. 20 se observa que la entrada 1 esta en nivel bajo y la entrada 2 a nivel alto, por lo tanto el giro es hacia atrás.

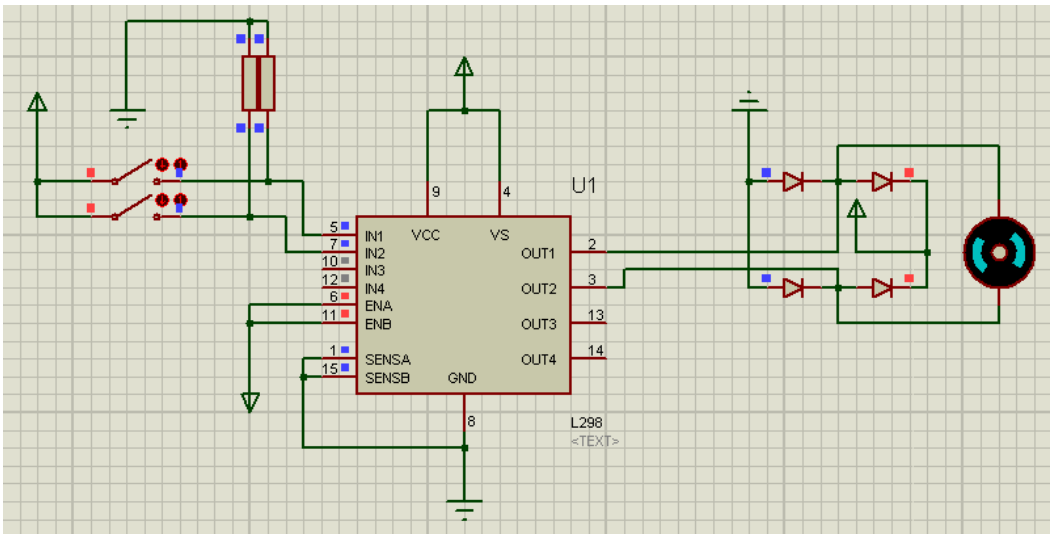


FIGURA No. 21) simulación en ISIS 7 Profesional.

En la figura No. 21 se observa que la entrada 1 y 2 están en nivel alto, por lo tanto el motor no gira.

Cuando las entradas del driver son iguales, el motor no gira.

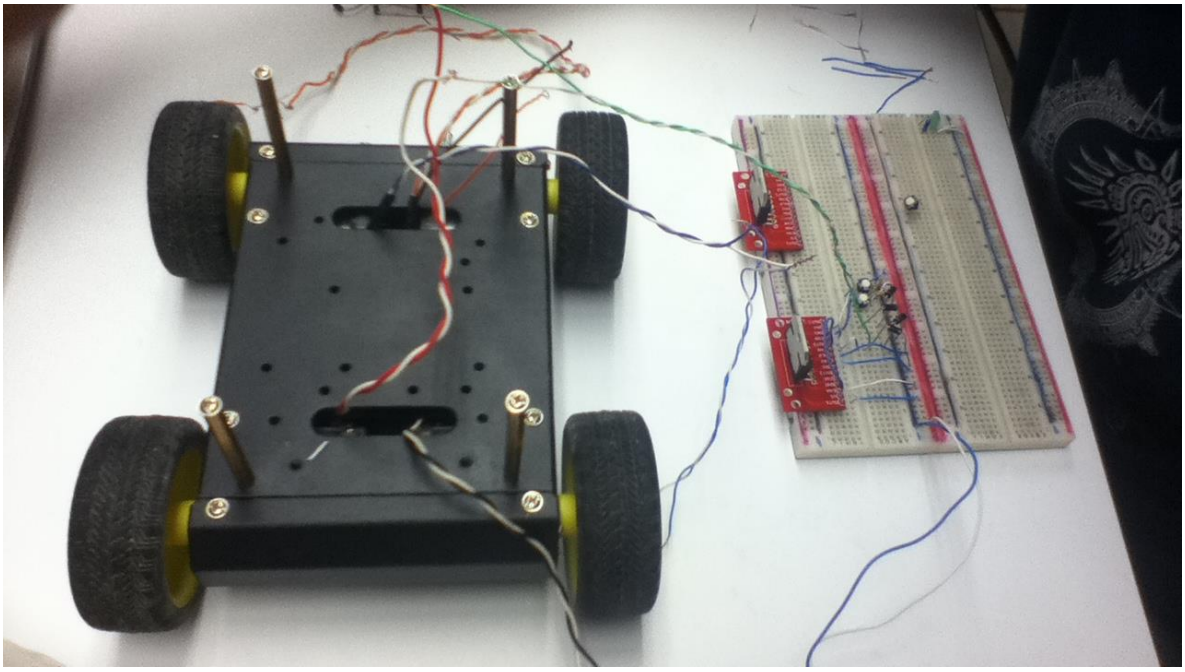


FIGURA No. 22) Primera prueba del driver.

Para el control de los motores, se colocaron en pares, esto quiere decir que los motores del lado derecho están siendo controlados por la misma señal de entrada, mas no están siendo controlados por la misma salida, y de la misma manera los motores del lado izquierdo, como se muestra en figura No. 22.

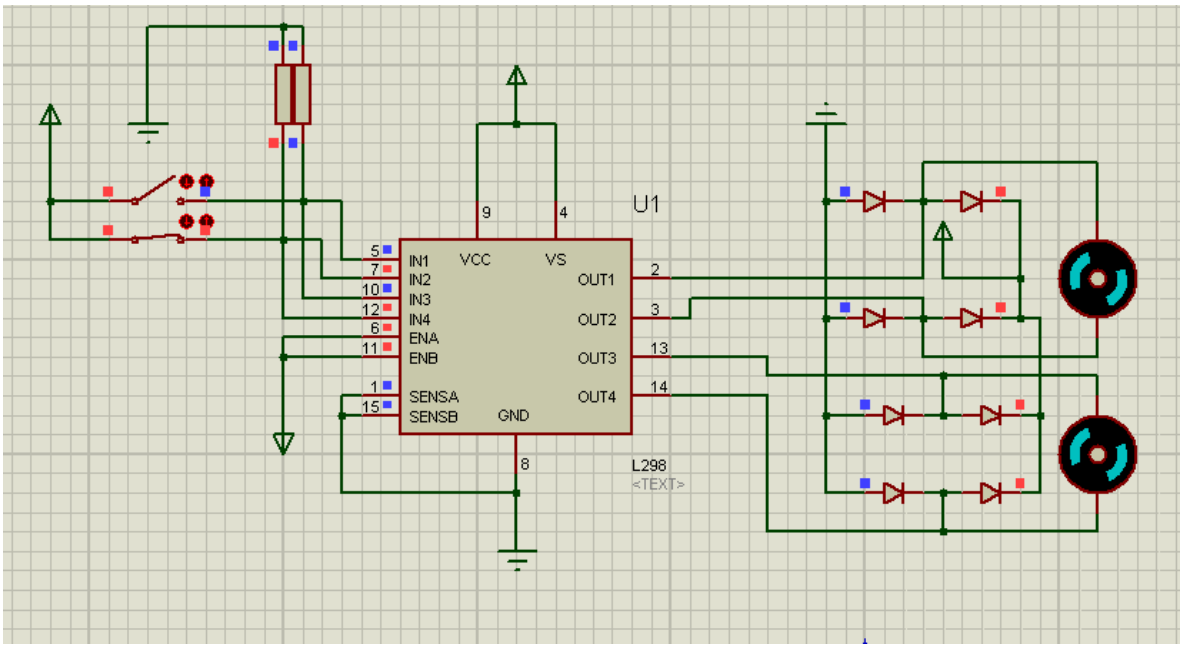


FIGURA No. 23) Simulación en ISIS 7 Profesional.

En la figura No. 23 se observa que la entrada 1 esta puenteada con la entrada 3 y la entrada 2 con la 4, y la salida son de manera independiente. Esto son los motores del lado derecho, están controlados por la misma señal, pero no por la misma salida. En esta imagen el giro de los motores es hacia adelante.

Después de pruebas realizadas se procedió ha realizar el rutado y posteriormente las placas, el diseño de las placas se realizaron en PCB Wizard, realizando dos placas una para los motores del lado derecho y otra para los motores del lado izquierdo, como se muestra en la figura No. 24.

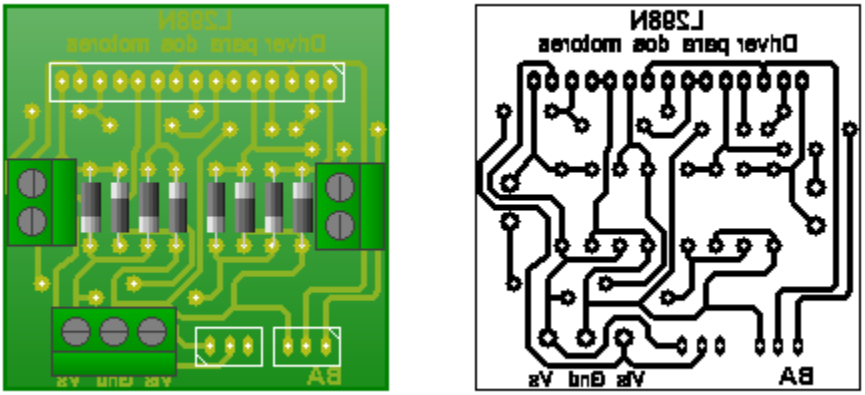


FIGURA No. 24) Rutado del puente h-l298b, PCB Wizard.

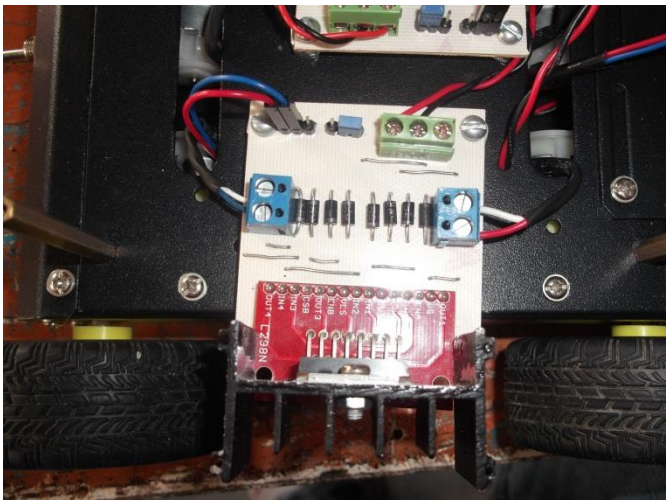


FIGURA No. 25) El driver quedo como se observa en esta figura.

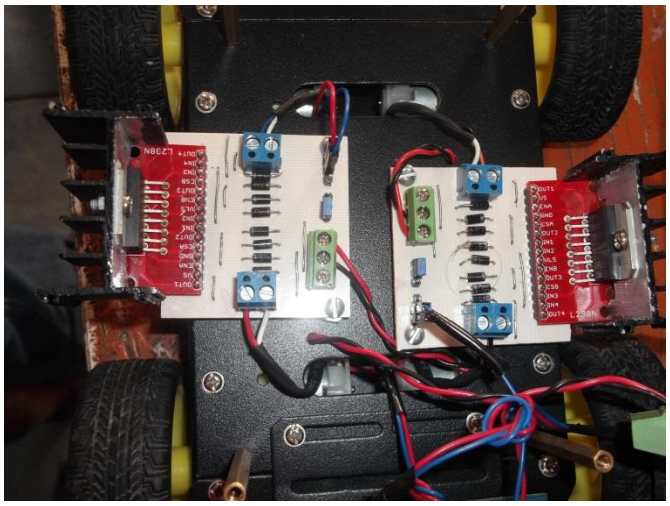


FIGURA No. 26) En esta imagen se puede observar las dos placas de los motores, tanto de la derecha como de la izquierda, las placas cuentan con una entrada de dos bits, una alimentación de 5v y dos salidas, una para cada motor del lado que le correspondan.

Sensores ultrasónicos.

Los sensores ultrasónicos utilizados fueron los SRF02 (Figura No. 27), estos sensores ultrasónicos cuentan con dos modos de comunicación, la comunicación serial y la comunicación I2C, que es la utilizamos (comunicación I2C).

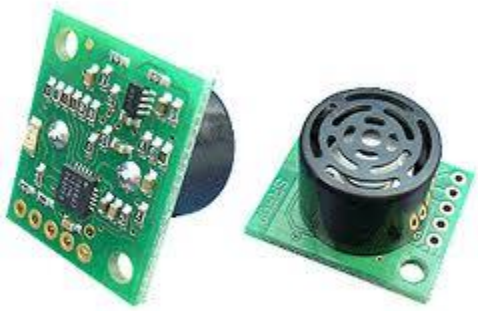


FIGURA No. 27) Sensores ultrasónicos SRF02.

Los sensores ultrasónicos nos permiten calcular la distancia a la que se encuentra un objeto, con esto nosotros podemos saber a que distancia se encuentra el objeto y poder tomar la decisión de hacia donde ir.

Para poder utilizar estos sensores primero debimos de estudiar la comunicación I2C.

I2C por sus siglas en ingles “Inter Integrated Circuito” o lo que es lo mismo en español “Entre circuitos integrados”, este tipo de comunicación nos permite conectar hasta 16 dispositivos de forma paralela, para poder identificar un dispositivo de otro se les asignan direcciones a cada uno de ellos, de manera que lo primero que se manda es la dirección del dispositivo y luego la instrucción a realizar, luego este dispositivo coloca en sus registros los datos para que los pueda enviar a el maestro. En la comunicación I2C hay un maestro y los demás dispositivos son esclavos, esto quiere decir que solo el maestro es el que tiene la facultad de dar las ordenes a los demás dispositivos, en nuestro caso es un arduino.

La comunicación I2C son dos líneas:

A) SDA: Es la línea de los datos.

B) SCL: Es la línea del reloj.

Como las dos líneas son de drenador abierto necesitan resistencias de pull-up.

El sensor SRF02 tiene 5 pines, como se muestra en la tabla No. 8:

Vcc	5v
SDA	Señal de datos
SCL	Señal de reloj
MODE	Serial/I2C
Tierra	0v

TABLA No. 8) Conexión de los pines del sensor SRF02.

Como se observa en l tabla Vcc a 5v, SDA a la línea de datos, SCL a la línea del pulso de reloj, MODE no se conecta, se deja flotando por que es la manera de configurarlo para que este en la comunicación I2C, y la tierra, que va a 0v.

La conexión se puede observar en la figura No. 28, donde las resistencias son de 10KΩ.

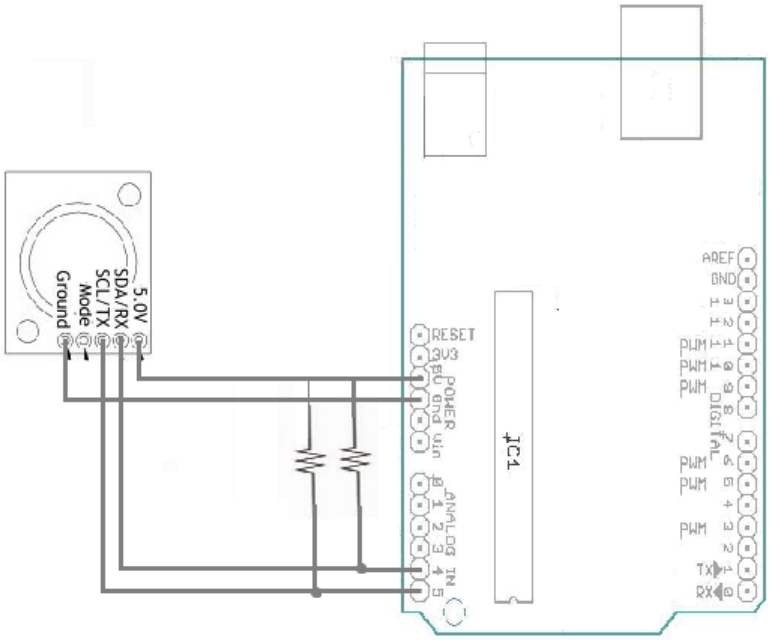


FIGURA No. 28) Conexión del sensor SRF02.

El sensor SRF02 trae por default la dirección 0xe0, pero si solo se utiliza un sensor no hay problema pero en nuestro caso utilizamos 3 sensores por lo tanto se le deben de cambiar la dirección, la dirección de nuestros sensores son las siguientes:

1) 113(decimal), 71(hexadecimal).
2) 114(decimal), 72(hexadecimal).
3) 115(decimal), 74(hexadecimal).

Para poder establecer la comunicación con el sensor se deben de llevar los siguientes pasos:

1) Abrir la comunicación. Esto nos quiere decir que debemos de abrir el bus para transmitir y recibir.
2) Colocamos en el bus la dirección del dispositivo con el que queremos establecer la comunicación.
3) En este caso, después de dar la dirección debemos de poner en 0 todos los registros del sensor para realizar un nuevo cálculo de la distancia.
4) Le damos la orden al sensor que calcule la distancia, el sensor nos da las opciones de poder calcular las distancias en pulgadas, centímetros y microsegundos. Los comandos son 0x50, 0x51 y 0x52 respectivamente.
5) Cerrar la comunicación, esto es para darle tiempo al sensor de calcular la distancia, el tiempo que tara en calcular la distancia el sensor es de 65 microsegundos.

6) Re-abrir la comunicación, volvemos a abrir el bus.
7) Volvemos a colocar la dirección del dispositivo.
8) Escribimos en el bus que nos queremos colocar en el registro donde se encuentra el cálculo realizado por el sensor (distancia), la distancia esta guardado en dos bytes, un byte alto y un byte alto.
9) Después de tener los bytes, tanto alto como bajo, los unimos para poder tener la distancia.
10) Cerramos la comunicación, para que se cierre el bus.

Se diseño la placa para la comunicación I2C, en PCB Wizard, quedando de la forma en que se muestra en la figura No. 29I:

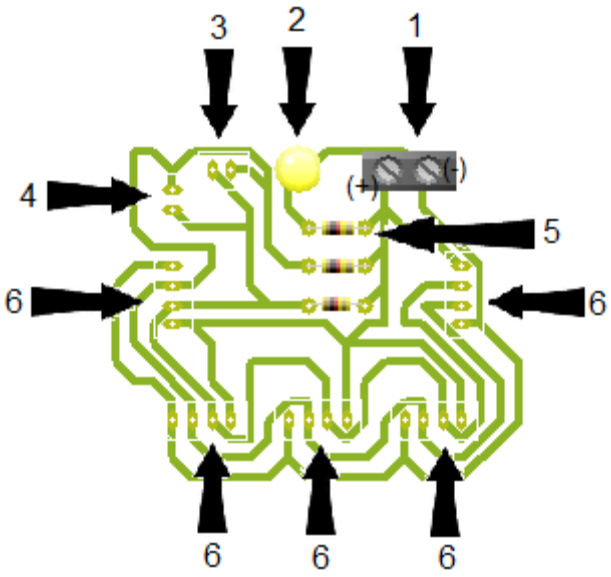


FIGURA No. 29) Rutado de la placa de comunicacionI2C, PCB Wizard.

En la tabla No. 9 se muestra las conexiones con la que cuenta la placa que se diseño.

1	Alimentación 5v.
2	Led indicador de alimentación.
3	Salida de la comunicación I2C. SDA y SCL.
4	Salida para conectar más dispositivos.
5	Resistencias de pull-up.

6	Salidas para los sensores SRF02.

TABLA No. 9) Conexiones de la placa.

FIGURA No. 30) La placa después de planchar y meter en el cloruro férrico.

FIGURA No. 31) La placa ddespués de colocar y soldar los componentes quedo de la forma que se ve en la imagen.

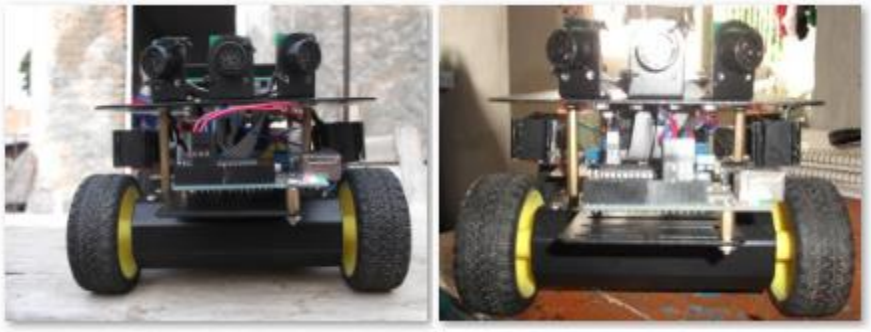


FIGURA No. 32) En estas imágenes se pueden observar los sensores montados en el móvil, la placa de la comunicación I2C no se puede observar ya que se encuentra oculta.

Sensores Analógicos.

Un sensor analógico es aquel que, como salida, emite una señal comprendida por un campo de valores instantáneos que varían en el tiempo, y son proporcionales a los efectos que se están midiendo, en la figura No. 33 se observan los sensores analógicos que utilizamos.

FIGURA No. 33) Sensores analógicos MQ-4, MQ-6 y MQ-7.

Esto nos quiere decir que dependiendo la entrada que tenga es el valor de la salida, en este caso los sensores que utilizamos son de un rango de 0v a 5v, esta es nuestra señal que nos brindan los sensores.

Sensor analógico MQ-4.

Este sensor nos permite detectar el gas metano, este es un sensor sencillo de usar para gas natural comprimido (CNG), ideal para detectar concentraciones de gas natural (compuesto en su mayoría por metano [CH4]) en el aire. El MQ-4 puede detectar concentraciones desde 200 hasta 100000 ppm (Figura No. 34).

FIGURA No. 34) Sensor MQ-4.

Este sensor es de alta sensibilidad y con un tiempo de respuesta alto, su salida es una resistencia analógica. El circuito para operarlo es bastante sencillo, lo único que se necesita es alimentar el devanado calefactor con 5v, añadir una resistencia de carga y conectar la salida a un ADC, como se observa en la figura No. 35.

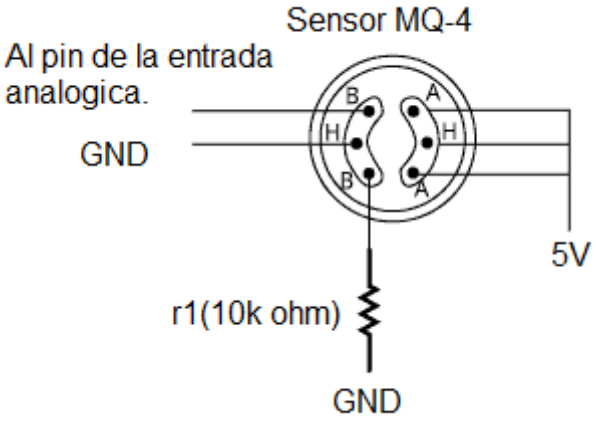


FIGURA No. 35) En esta imagen se observa la configuración del sensor.

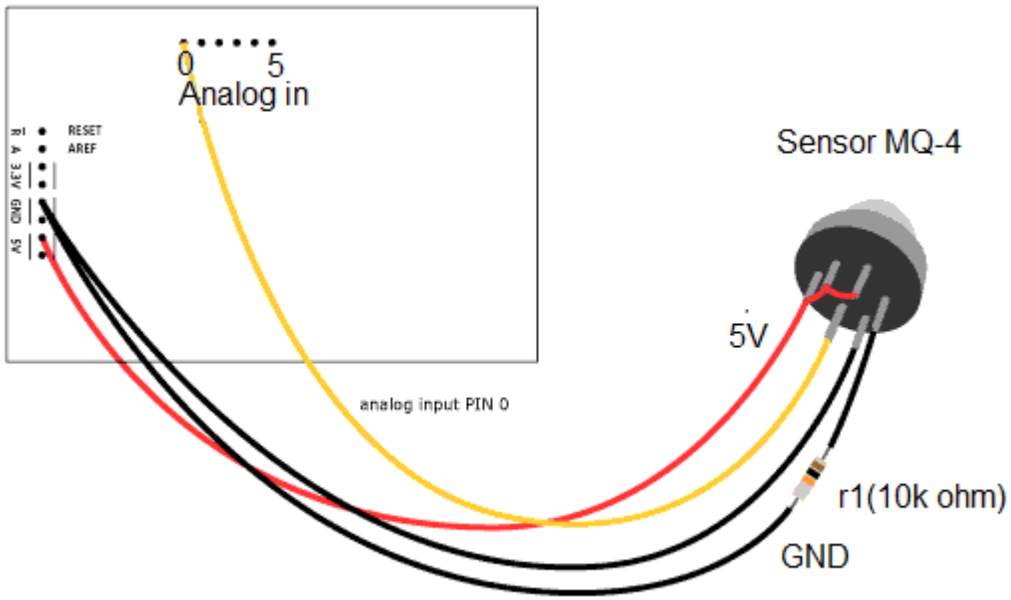


FIGURA No. 36) En esta imagen se observa la manera de conectarla al sensor, como ya se había mencionado es muy sencillo, la salida solo se debe de conectar a una entrada analógica del arduino.

Sensor analógico MQ-6.

Este sensor nos permite detectar el gas butano y propano, este es un sensor sencillo de usar para gas licuado de petróleo (LPG), ideal para detectar concentraciones LPG (compuesto en su mayoría por propano y butano) en el aire. El MQ-6 puede detectar concentraciones desde 200 hasta 100000 ppm (figura No. 37).

FIGURA No. 37) Sensor MQ-6.

Este sensor es de alta sensibilidad y con un tiempo de respuesta alto, su salida es una resistencia analógica. El circuito para operarlo es bastante sencillo, lo único que se necesita es alimentar el devanado calefactor con 5v, añadir una resistencia de carga y conectar la salida a un ADC como se observa en la Figura No. 38.

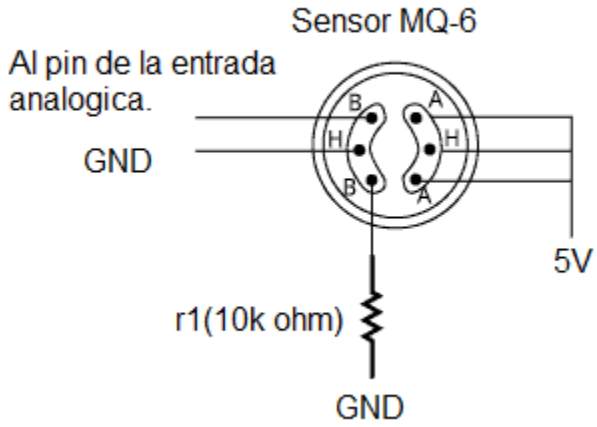


FIGURA No. 38) En esta imagen se observa la configuración del sensor.

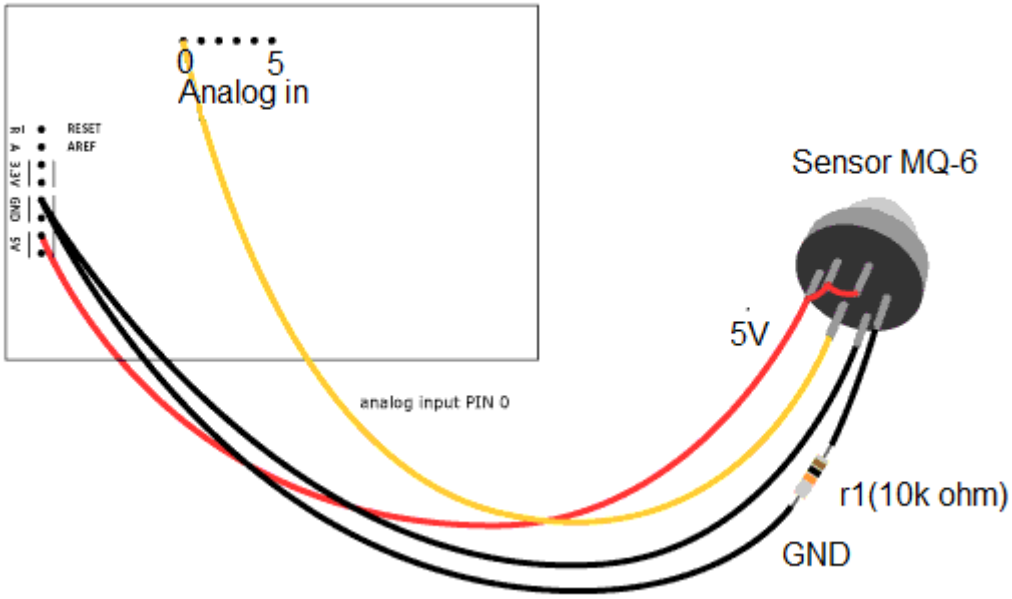


FIGURA No. 39) En esta imagen se observa la manera de conectarla al sensor, como ya se había mencionado es muy sencillo, la salida solo se debe de conectar a una entrada analógica del arduino.

Sensor analógico MQ-7.

Este sensor nos permite detectar el gas de monóxido de carbono, este es un sensor sencillo de usar para gas de monóxido de carbono (CO), ideal para detectar concentraciones CO en el aire. El MQ-7 puede detectar concentraciones desde 20 hasta 2000 ppm (FIGURA No. 40).

FIGURA No. 40) Sensor MQ-6.

Este sensor es de alta sensibilidad y con un tiempo de respuesta alto, su salida es una resistencia analógica. El circuito para operarlo es bastante sencillo, lo único que se necesita es alimentar el devanado calefactor con 5v, añadir una resistencia de carga y conectar la salida a un ADC, como se observa en la figura No. 41.

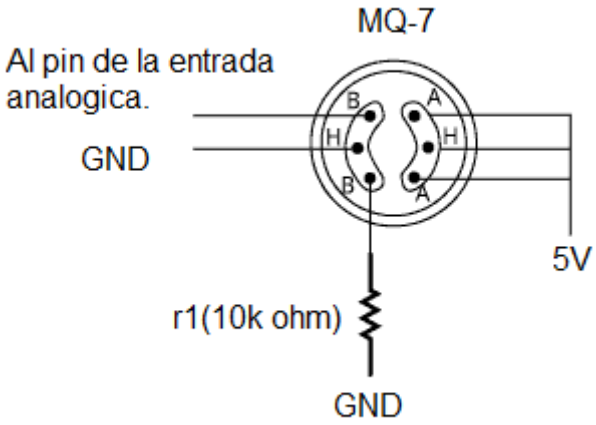


FIGURA No. 41) En esta imagen se observa la configuración del sensor.

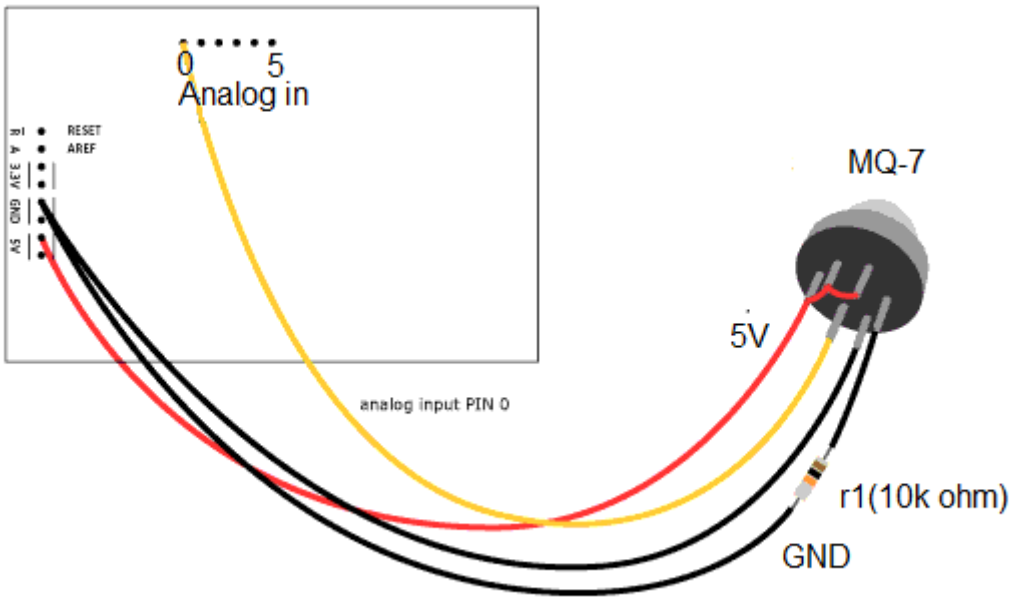


FIGURA No. 42) En esta imagen se observa la manera de conectarla al sensor, como ya se había mencionado es muy sencillo, la salida solo se debe de conectar a una entrada analógica del arduino.

Brújula electrónica.

Se describe un sistema de brújula electrónica que incluye un circuito sensor magnético que tiene por lo menos dos elementos sensores para detectar componentes perpendiculares del vector del campo magnético de la Tierra. Un circuito de procesamiento se conecta al circuito sensor para filtrar, procesar y calcular una orientación. El circuito de procesamiento además selecciona un patrón geométrico de aproximación, tal como una esfera, elipsoide, elipse o círculo, determina un error métrico de los puntos de datos con relación al patrón de aproximación, ajusta el patrón para minimizar el error, por lo cual se obtiene un patrón de mejor ajuste. El patrón de mejor ajuste entonces se utiliza para calcular la orientación para cada lectura sucesiva del sensor con la condición de que el nivel de ruido no sea ruidoso y hasta que se identifique un nuevo patrón de mejor ajuste.

En este caso la brújula que se utilizo fue una brújula HMC6352 (Figura No. 43) con interface I2C, ideal para aplicaciones que requieren posición absoluta con el polo magnético, es decir que se pueden tomar mediciones para aplicaciones como odométria, en robótica móvil para posicionamiento, que es nuestro caso.

FIGURA No. 43) Brújula electrónica HMC6352.

Algunas de las características con las que cuenta la brújula HMc6352 son las siguientes:

*Interfaz I2C por lo que solo necesitamos 2 pines para obtener los datos.

*Alimentación 2.7 - 5.2V. Lo recomendable es utilizar el mínimo por consumo, en este caso lo alimentaremos con el pin de 3v del Arduino.

*Refresco ajustable de 1 a 20Hz.

*Resolución 0.5 grados. Con pedir datos a nivel de grado tendremos una buena precisión.

*Consumo: 1mA (3V).

*Dimensiones: 15x15mm.

*Permite calibrado.

La calibración es algo muy importante ya que como se cuenta con motores en el móvil, si no es calibrado esto afectar, para calibrarlo utilizaremos un comando de la librería Libcompass, el comando es GetHeadingSerial, nos devolverá por el puerto serie el ángulo del sensor con respecto al norte y CalibrateCompass nos permite calibrar el sensor.

Esto último es muy importante. Debemos subir este sketch al arduino y nos indicará a través del puerto serie que giremos el sensor 720º (dos vueltas). De esta manera el sensor mide los campos magnéticos de alrededor y se calibra para obviarlos respecto al terrestre. Ya que incluimos este sensor en un móvil, este tendrá motores, que tienen imanes. Por lo que debemos calibrarlo ya instalado en el móvil para que nos apunte al campo magnético terrestre y no incorrectamente a los motores u otra perturbación magnética.

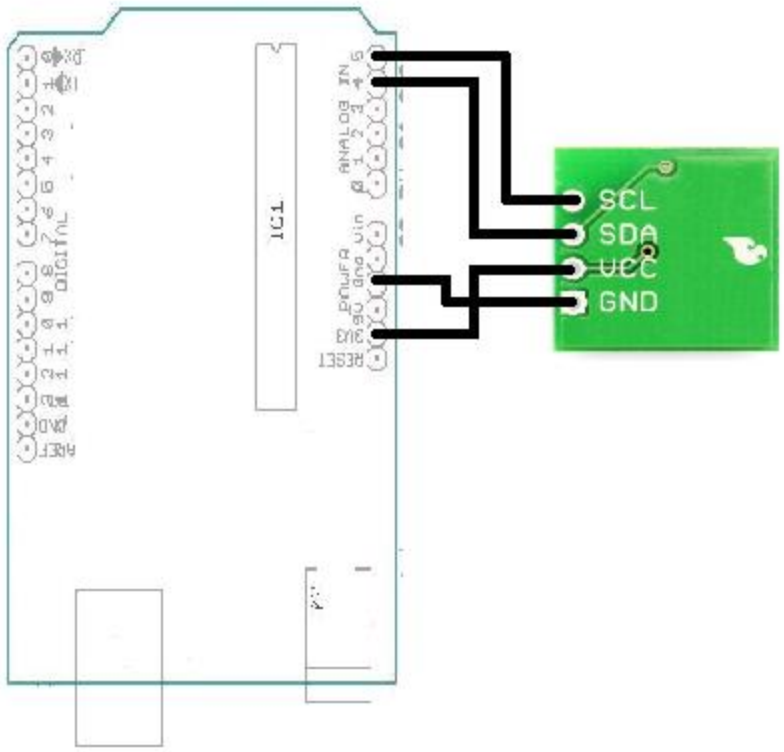


FIGURA No. 44) Conexión de la brújula electrónica.

En la figura No. 44 se puede observar que el sensor se alimenta con 3.3V, lo podemos tomar del arduino, o de otra fuente externa, los pines de la comunicación I2C, SDA y SCL van conectado a los pines analógicos A4 yA5 respectivamente

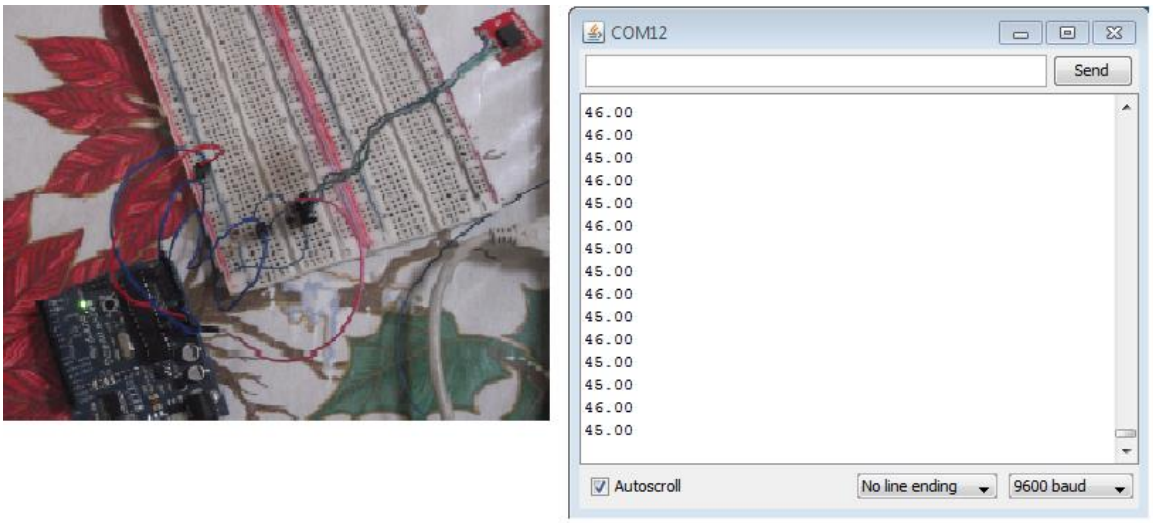


FIGURA No. 45) Pruebas con la brújula electrónica, mostrado en la terminal virtual del compilador de arduino.

En la figura No. 45 se puede observar la brújula conectada, como se había mencionado anteriormente, la brújula nos marca los grados al norte que nos encontramos, en este caso nos encontramos a 45º del norte, de forma serial, en una terminal virtual del programa de arduino.

Matlab.

Este es un programa que se realizó para poder observar en la computadora las distancias de los sensores, los podemos guardar en 3 vectores, y con ellos podemos graficar o guardarlos, figura No. 46.

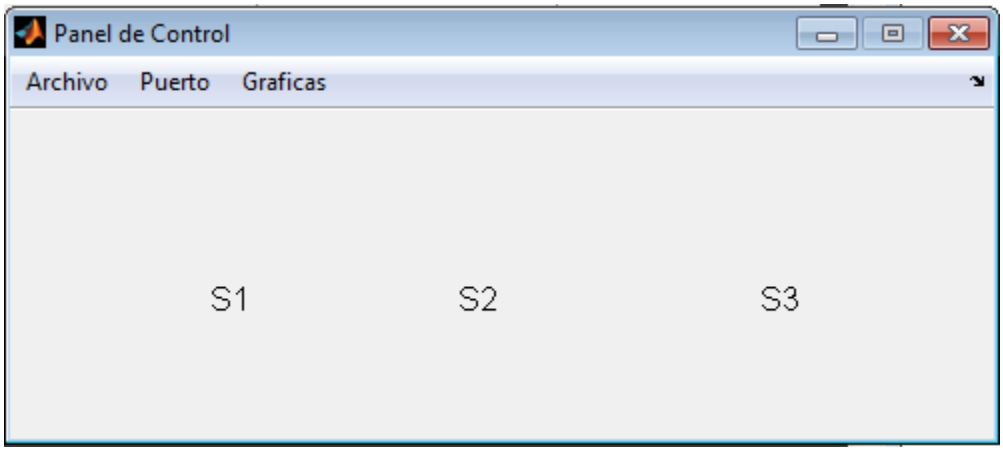


FIGURA No. 46) Programa que se realizó para la visualización de datos.

La conexión es con el arduino, es por medio de un max232, que es un convertidor de comunicación USB a serial.

En el programa diseñado se cuenta con tres pestañas en la parte superior que son:

*Archivo.

*Puerto.

*Graficas.

En la pestaña “Archivo” contamos con las opciones

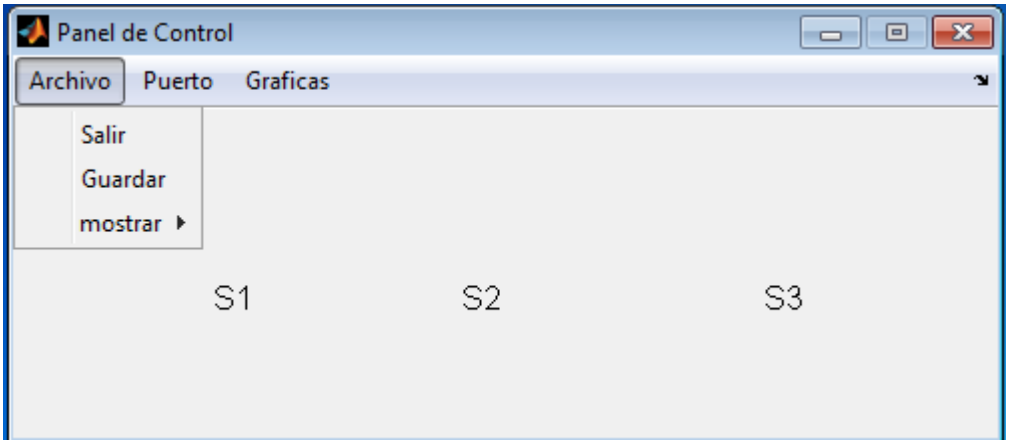


FIGURA No. 47) Opciones de la pestaña archivo.

*Salir, nos sirve para salir del programa, y para cerrar el puerto.

*Guardar, nos sirve para guardar las distancias, que son tres vectores de 100 datos cada uno, denominados v1, v2 y v3.

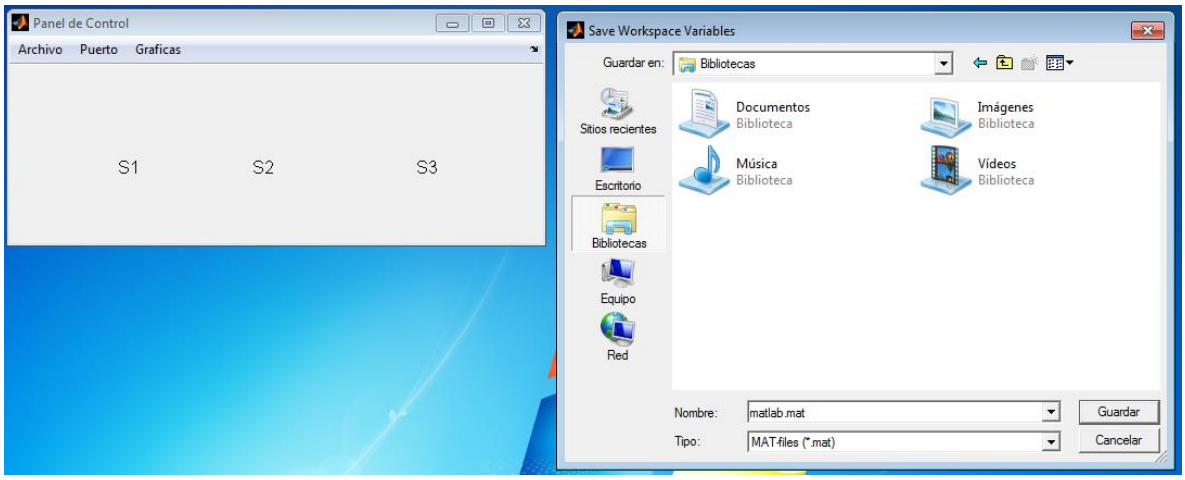


FIGURA No. 48) Como guardar los datos.

*Mostrar, este nos permite observar los vectores, el valor de las distancias.

En la pestaña “Puerto” tenemos las opciones:

*Conectar, este nos permite conectarnos a el puerto, en este caso yo tengo el COM10.

*Desconectar, este nos permite desconectarnos del puerto, lo cierra para que no vaya a existir algún tipo de conflicto.

En la pestaña “Graficas” nos da la opción de graficar alguna de las distancias.

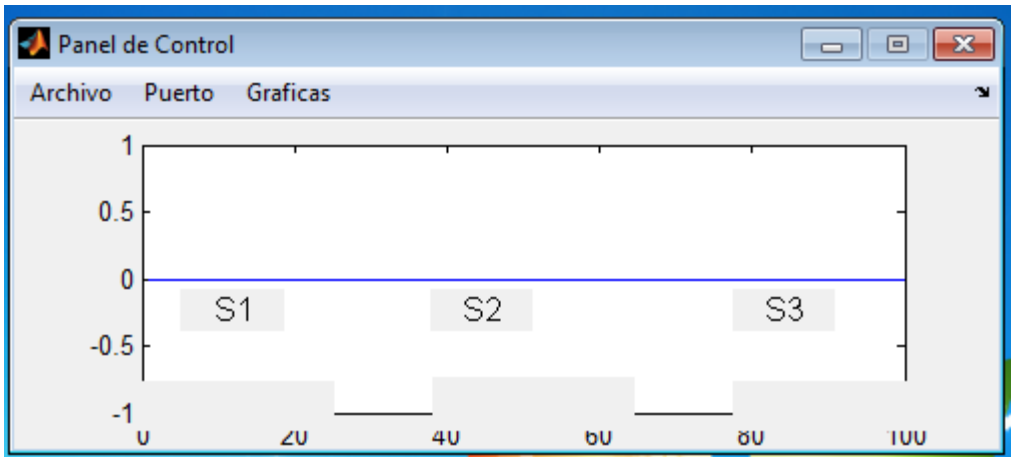


FIGURA No. 49) Grafica del valor del sensor V1.

3.2.- Programas.

La programación se realizó en arduino-1, y en arduino-19, la programación se realizo por separado, para las pruebas, se realizó en arduino duemilanove.

Sensores analógicos, estos son los MQ-4, MQ-6 y MQ-7.

El programa realiza la lectura de los puertos analógicos, hacemos la equivalencia de las partes por millón y los mostramos en la lcd de 2X16.

```
#include <LiquidCrystal.h>
int mq4,mq41,mq7,ppm,ppm1,ppm2;
LiquidCrystal lcd(8,9,10,11,12,13);

void setup()
{
lcd.begin(16,2);
lcd.print("Iniciando");
}
void loop()
{
 mq4 = analogRead(A0);
 mq41 = analogRead(A1);
 mq7 = analogRead(A2);
 ppm=mq4 * 0.100;
 ppm1=mq41 * 0.100;
 ppm2=mq7 * 0.506;
 lcd.clear();
```

```
 lcd.print("CH4: ");
 lcd.print(ppm);
 lcd.print("ppm ");
 lcd.print(ppm1);
 lcd.print("ppm");
 lcd.setCursor(0,1);
 lcd.print("CO: ");
 lcd.print(ppm2);
 lcd.print(" ppm");
 delay(1000);
}
```

La brújula, este programa realiza la lectura de la brújula y lo muestra en una lcd y de forma serial.

```
#include <Wire.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(8,9,10,11,12,13);
void setup()
{
 Wire.begin();
 Serial.begin(9600);
 lcd.begin(16,2);
 lcd.print("Iniciando");
}
void loop()
{
 Wire.beginTransmission(0x21);
 Wire.write("A");
 delay(100);
 Wire.requestFrom(0x21,2);
 byte MSB = Wire.read();
 byte LSB = Wire.read();
 Wire.endTransmission();
 float myres = ((MSB << 8) + LSB)/10;
 lcd.clear();

 lcd.print("Grados:");
 lcd.print(myres);
 Serial.println(myres);
 delay(1000);
 }
```

Comunicación I2C (sensores ultrasónicos), este programa realiza la lectura de los sensores y la muestra en un lcd de 2x16.

```
#include <Wire.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(8,9,10,11,12,13);
void setup()
{
 Wire.begin();
 lcd.begin(16,2);
}
int reading = 0,reading1 = 0,reading2 = 0;
void loop()
{
  Wire.beginTransmission(113);
  Wire.write(byte(0x00));
  Wire.write(byte(0x51));
  Wire.endTransmission();
  delay(100);
  Wire.beginTransmission(113);
  Wire.write(byte(0x02));
```

```
Wire.endTransmission();
Wire.requestFrom(113,2);
if(2 <= Wire.available())
 {
  reading = Wire.read();
  reading = reading << 8;
  reading |=Wire.read();
 }
delay(100);
Wire.beginTransmission(114);
Wire.write(byte(0x00));
Wire.write(byte(0x51));
Wire.endTransmission();
delay(100);
Wire.beginTransmission(114);
Wire.write(byte(0x02));
Wire.endTransmission();
Wire.requestFrom(114,2);
if(2 <= Wire.available())
 {
  reading1 = Wire.read();
  reading1 = reading1 << 8;
  reading1 |=Wire.read();
 }
delay(100);
Wire.beginTransmission(115);
Wire.write(byte(0x00));
Wire.write(byte(0x51));
Wire.endTransmission();
delay(100);
Wire.beginTransmission(115);
Wire.write(byte(0x02));
Wire.endTransmission();
Wire.requestFrom(115,2);
if(2 <= Wire.available())
 {
  reading2 = Wire.read();
  reading2 = reading2 << 8;
  reading2 |=Wire.read();
 }
delay(100);
Wire.beginTransmission(0x21);
Wire.write("A");
delay(100);
Wire.requestFrom(0x21,2);
byte MSB = Wire.read();
byte LSB = Wire.read();
Wire.endTransmission();
float myres = ((MSB << 8) + LSB)/10;
delay(100);
lcd.clear();
lcd.print("D.1:");
lcd.print(reading);
lcd.setCursor(8,0);
lcd.print("D.2:");
lcd.print(reading1);
lcd.setCursor(0,1);
lcd.print("D.3:");
lcd.print(reading2);
lcd.setCursor(8,1);
lcd.print("G.:");
lcd.print(myres);
}
```

Prueba, este programa es la prueba de los sensores ultrasónicos, para que avance y se detenga.

```
 #include <Wire.h>
int reading, reading2,reading3;
int ledPin =13;
void setup()
{
  Wire.begin();
  pinMode(ledPin,OUTPUT);
  pinMode(12,INPUT);
  pinMode(11,INPUT);
  pinMode(10,INPUT);
}
void loop()
{
 Wire.beginTransmission(114);
  Wire.write(byte(0x00));
  Wire.write(byte(0x51));
  Wire.endTransmission();
  delay(100);
  Wire.beginTransmission(114);
  Wire.write(byte(0x02));
  Wire.endTransmission();
  Wire.requestFrom(114,2);
   if(2 <= Wire.available())
   {
    reading = Wire.read();
    reading = reading << 8;
    reading |=Wire.read();
   }
  Wire.beginTransmission(115);
  Wire.write(byte(0x00));
  Wire.write(byte(0x51));
  Wire.endTransmission();
  delay(100);
  else
  Wire.beginTransmission(115);
  Wire.write(byte(0x02));
  Wire.endTransmission();
  Wire.requestFrom(115,2);
   if(2 <= Wire.available())
   {
    reading2 = Wire.read();
    reading2 = reading << 8;
    reading2 |=Wire.read();
   }*/
    /*Wire.beginTransmission(116);
  Wire.write(byte(0x00));
  Wire.write(byte(0x51));
  Wire.endTransmission();
  delay(100);
  Wire.beginTransmission(116);
  Wire.write(byte(0x02));
  Wire.endTransmission();
  Wire.requestFrom(116,2);
   if(2 <= Wire.available())
   {
    reading3 = Wire.read();
    reading3 = reading3 << 8;
    reading3 |= Wire.read();
   }
  if(reading > 30)
   {
    digitalWrite(ledPin, HIGH);
    digitalWrite(12,LOW);
    digitalWrite(11,HIGH);
    digitalWrite(10,LOW);
   }
```

```
 {
   digitalWrite(ledPin, LOW);
   digitalWrite(12,LOW);
   digitalWrite(11,LOW);
   digitalWrite(10,LOW);
 }
```

MATLAB.

En el programa que se realizó en matlab, nos muestra los valores que se estan sensando con los sensores ultrasonicos, este programa obtiene la lectura en forma de string y se convierte en entero, se verifica de que sensor es el valor que acaba de recibir, los valores recibidos se guardan en tres vectores de tamaño de [100;1], esto nos sirve para poder guardar los valores al momento de salir.

El GUI se muestra en la figura No. 50, y el codigo se encuenttra en el anexo 1, en la pag. 65.

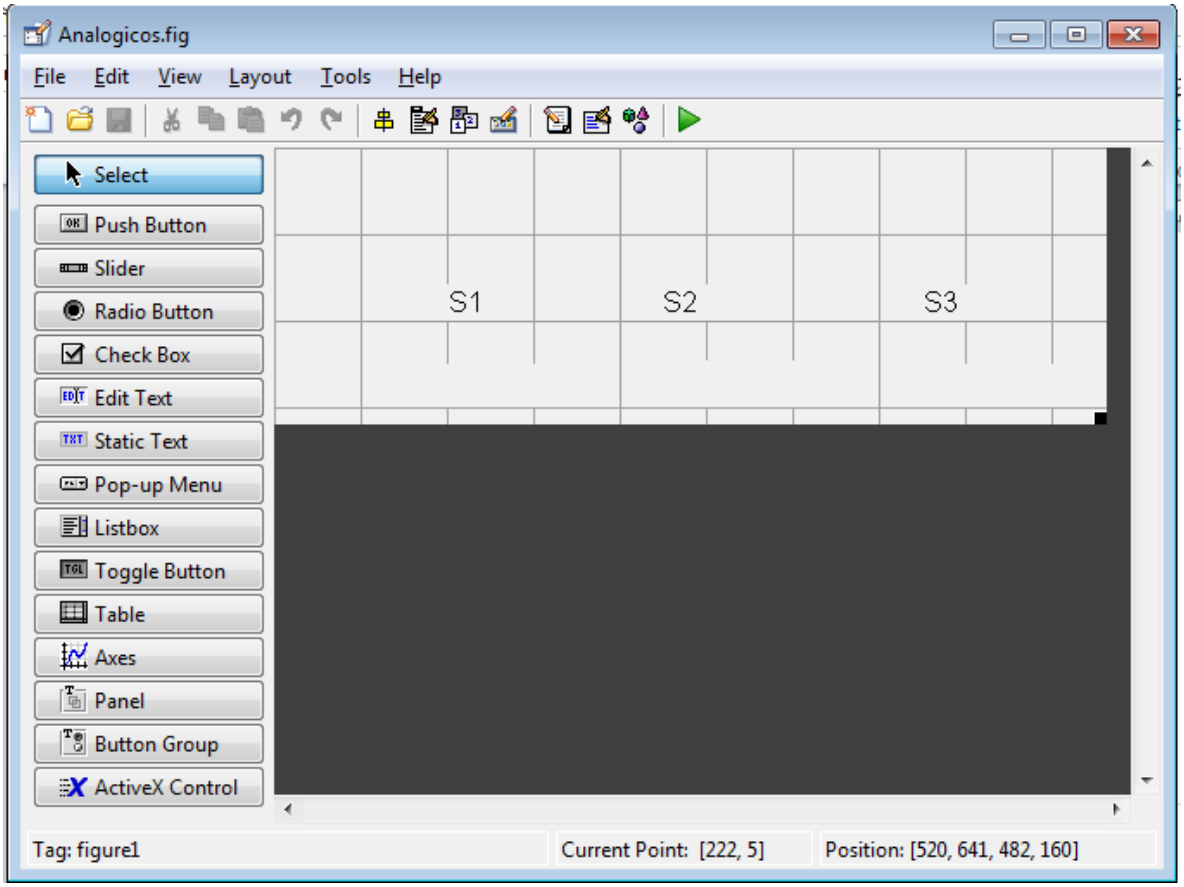


FIGURA No. 50) Figura creada en matlab, para el programa.

4.- Análisis de resultados.

Los resultados obtenidos fueron satisfactorios, ya que se consiguió el funcionamiento deseado, en todas las partes que integran el sistema del móvil, el funcionamiento de los motores, la lectura de los sensores ultrasónicos y biométricos, la brújula electrónica y la interface en Matlab.

El funcionamiento del sistema se vio reflejado en pruebas realizadas en un ambiente controlado, el sistema era capaz de detectar los obstáculos que tenía enfrente, para posteriormente poder esquivarlos y enviar los datos de los sensores ultrasónicos, que nos sirven para la detección de objetos.

Los sensores biométricos nos permiten realizar la detección de gases, y el sistema envía los datos a la computadora para poder ser procesados, y saber donde se encuentra la emanación de estos gases.

La interface en Matlab nos muestra los datos que el móvil detecta, y nos permite guardarlos para posteriormente poderlos procesarlos de la mejor manera.

Anexos.

Anexo 1.- Codigo de la interface de MATLAB

```
function varargout = Analogicos(varargin)
% ANALOGICOS M-file for Analogicos.fig
%      ANALOGICOS, by itself, creates a new ANALOGICOS or raises the existing
%      singleton*.
%
%      H = ANALOGICOS returns the handle to a new ANALOGICOS or the handle to
%      the existing singleton*.
%
%      ANALOGICOS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ANALOGICOS.M with the given input arguments.
%
%      ANALOGICOS('Property','Value',...) creates a new ANALOGICOS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Analogicos_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Analogicos_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Analogicos

% Last Modified by GUIDE v2.5 12-Mar-2012 19:33:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Analogicos_OpeningFcn, ...
                   'gui_OutputFcn',  @Analogicos_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Analogicos is made visible.
function Analogicos_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Analogicos (see VARARGIN)

% Choose default command line output for Analogicos
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Analogicos wait for user response (see UIRESUME)
% uiwait(handles.figure1);

set(gcf,'Name','Panel de Control')
global h s1 s2 s3 v1 v2 v3
v1=zeros(100,1);
v2=zeros(100,1);
v3=zeros(100,1);
h = handles;

function recibe(varargin)
global p h s1 s2 s3 v1 v2 v3
x = str2num(fgetl(p));
if(x>0 && x<=600)
  s1=x;
  v1(1:99) = v1(2:100)
  v1(100) = s1;
  set(h.s1,'String',num2str(s1));
end

if(x>600 && x<=1201)
```

```
    s2=x-601;
    v2(1:99) = v2(2:100)
    v2(100)=s2;
    set(h.s2,'String',num2str(s2));
end

if(x>1201 && x<=1802)
    s3=x-1201;
    v3(1:99) = v3(2:100)
    v3(100) = s3;
    set(h.s3,'String',num2str(s3));
end


% --- Outputs from this function are returned to the command line.
function varargout = Analogicos_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --------------------------------------------------------------------
function Archivo_Callback(hObject, eventdata, handles)
% hObject    handle to Archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Puerto_Callback(hObject, eventdata, handles)
% hObject    handle to Puerto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Puerto_Conectar_Callback(hObject, eventdata, handles)
% hObject    handle to Puerto_Conectar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

global p
delete(instrfind)
p = serial('COM10');
set(p,'BaudRate',9600,'BytesAvailableFcn',@recibe)
fopen(p)

% --------------------------------------------------------------------
function Puerto_Desconectar_Callback(hObject, eventdata, handles)
% hObject    handle to Puerto_Desconectar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global p
fclose(p)
delete(p)

% --------------------------------------------------------------------
function Archivo_Salir_Callback(hObject, eventdata, handles)
% hObject    handle to Archivo_Salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

delete(gcbf)


% --------------------------------------------------------------------
function guardar_Callback(hObject, eventdata, handles)
% hObject    handle to guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global v1 v2 v3
uisave({'v1', 'v2','v3'})



% --------------------------------------------------------------------
function Graficas_Callback(hObject, eventdata, handles)
% hObject    handle to Graficas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
function Graficas_v1_Callback(hObject, eventdata, handles)
% hObject    handle to Graficas_v1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global v1
plot(v1)


% --------------------------------------------------------------------
function Archivo_mostrar_Callback(hObject, eventdata, handles)
% hObject    handle to Archivo_mostrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Archivo_mostrar_v1_Callback(hObject, eventdata, handles)
% hObject    handle to Archivo_mostrar_v1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load('matriz','v1');
```

Anexo 2.- Fotografías y diagramas del proyecto.

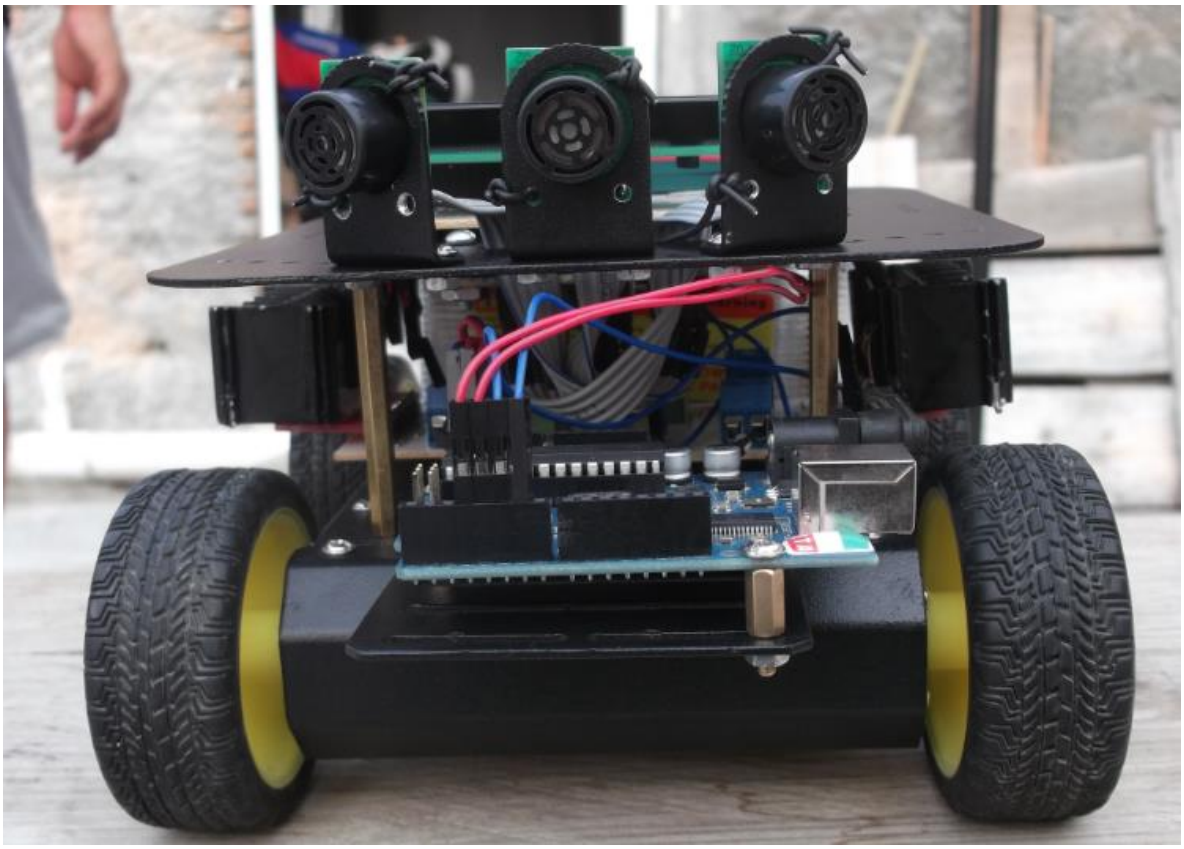


FIGURA No. 51) En esta imagen se puede observar el móvil con el arduino, sensores ultrasónicos, los driver, el circuito de alimentación se encuentra en la parte posterior del móvil, la batería de Li-PO se encuentra montada.

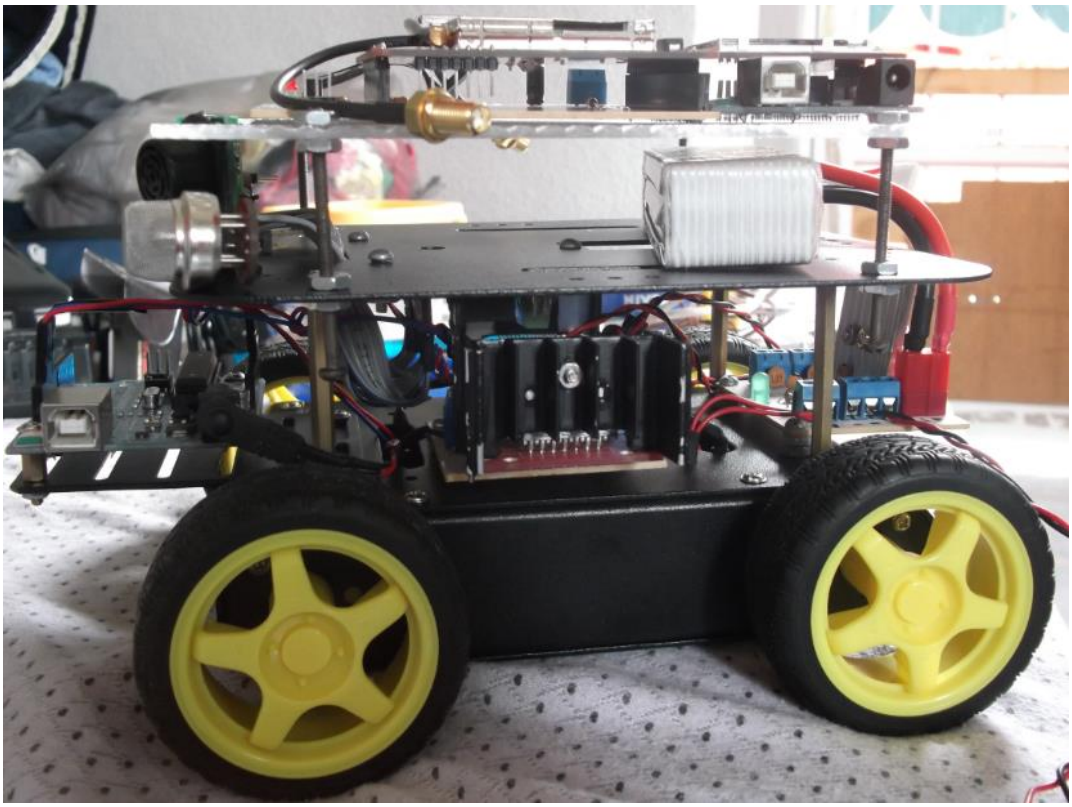


FIGURA No. 52) En esta imagen se puede observar el móvil, con las modificaciones que se le debieron de realizar, se le agregó un piso más, ya que teníamos que colocar el modulo GPS, los sensores biométricos ya se pueden observar, y la placa de alimentación del sistema se puede observar claramente en esta imagen.

Diagramas y placas

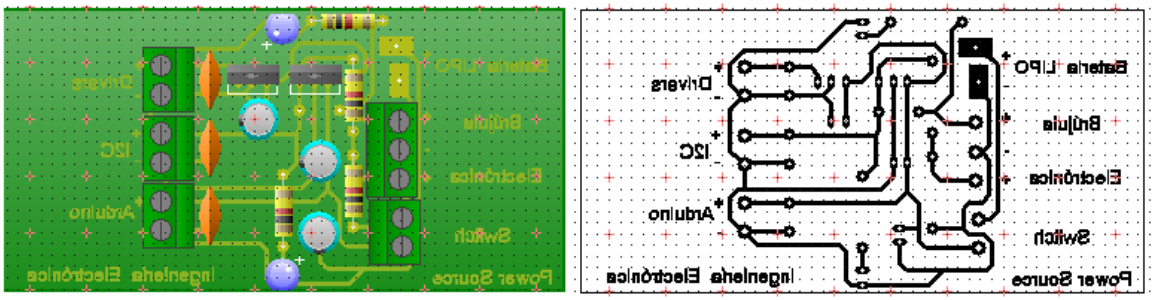


FIGURA No. 53) En la imagen se puede observar la conexión y el rutado de la placa que alimentara a todo el sistema del móvil, a partir de una batería de Li-PO, fue diseñado en el PCB Wizard.

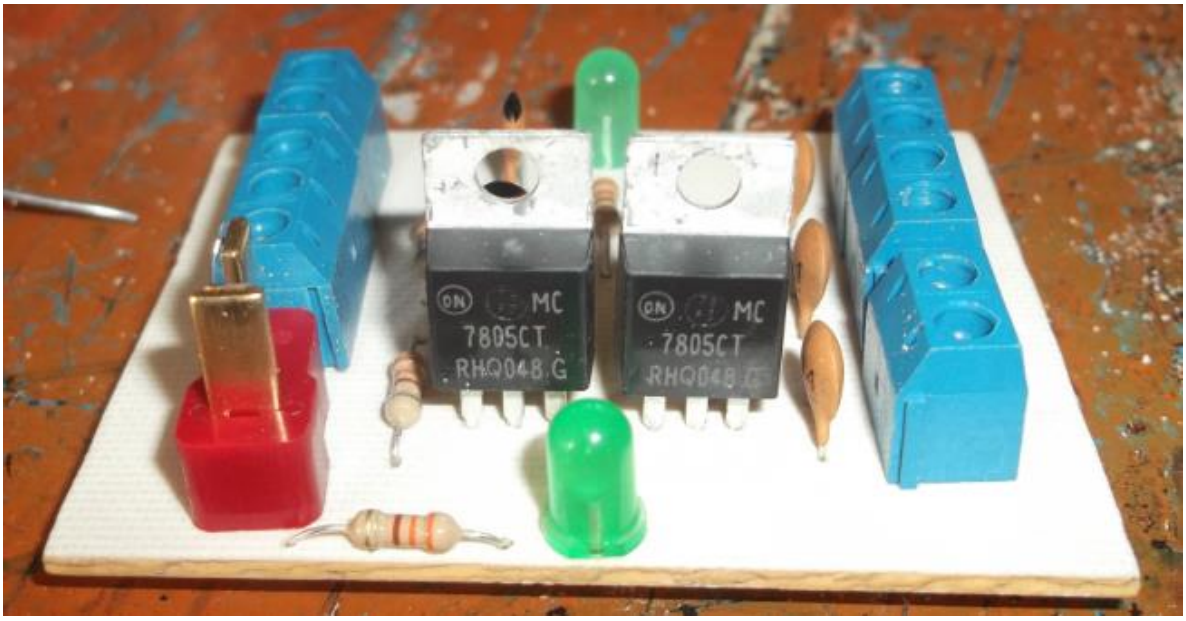


FIGURA No. 54) En esta imagen se puede observar la placa de alimentación del sistema terminada, donde se pueden apreciar dos reguladores 7805, la entrada de la batería Li-PO, las borneras de salidas, y los leds indicadores.

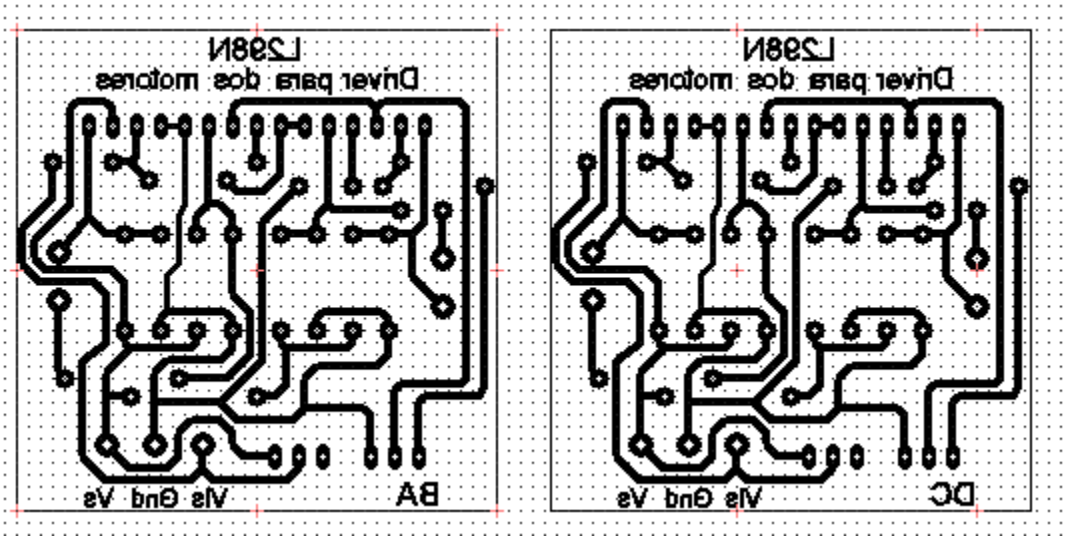


FIGURA No. 55) En esta imagen se pude observar los diagramas de los drivers puente h l-298, las pistas de la forma en que se conectaron y de la manera como debe de quedar, diseñado en PCB Wizard.

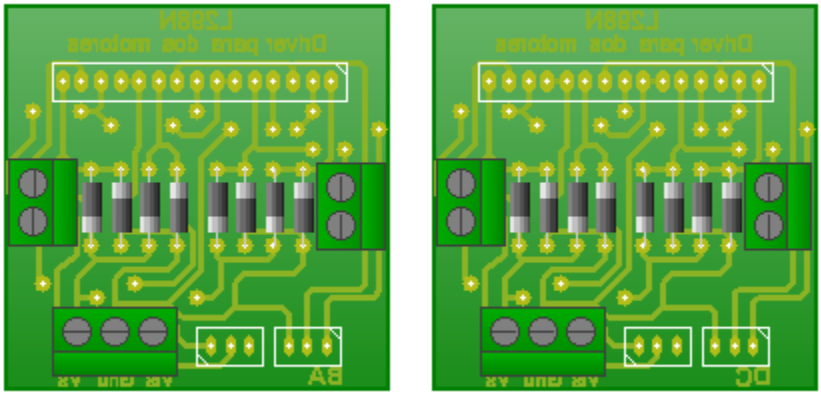


FIGURA No. 56) En esta figura se puede observar la forma en que quedarón los componentes.

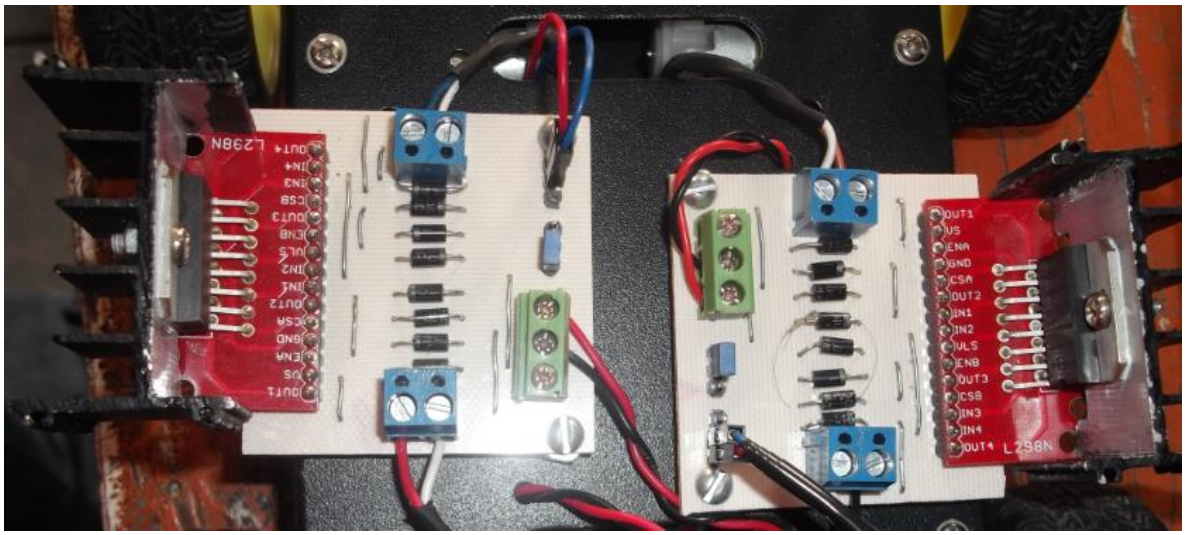


FIGURA No. 57) En esta imagen se puede observar la forma en que quedó el circuito, cuenta con una entrada de voltaje (borneras verdes), una señal de entrada (pines), dos salidas (borneras azules).

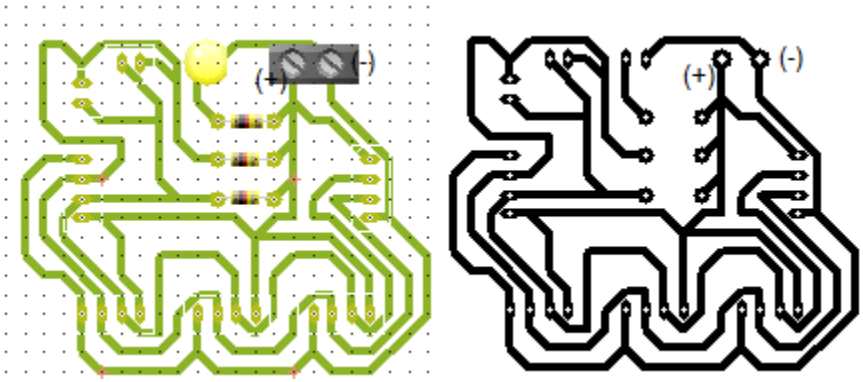


FIGURA No. 58) En esta imagen se puede observar el rutado de la placa para la comunicación I2C.

FIGURA No. 59) En esta imagen se puede observar la placa de la comunicación I2C, donde tenemos la alimentación (bornera azul), un led indicador, dos pines de la comunicación I2C (alado del led), otros dos pines para conectar la brújula digital, y 5 salidas para los sensores SRF02.

Conclusión.

Este es un proyecto muy complejo, por eso se descompuso en varias partes, para poder hacer varias tareas pero de menor complejidad, y al último poder unirlos.

Para cada una de las partes primero se debieron de realizar las pruebas, ya que debíamos de saber la conexión correcta de cada dispositivo, para poder posteriormente realizar las placas necesarias.

Se utilizó la plataforma arduino, que es una de las plataformas que se está utilizando con mucha frecuencia, es una plataforma diferente a la del microcontrolador de la microchip, ya que esta te ofrece algunas ventajas, las ventajas es que la placa ya está diseñada nada más para poder conectar las entradas y salidas, otra ventaja es que la programación de esta placa es muy sencilla, sin necesidad de un programador extra, las librerías son fáciles de entender y de utilizar.

Se aprendió a utilizar la comunicacionI2C, ya que no teníamos estos conocimientos, y que son muy interesantes e importantes, ya que muchos dispositivos usan este protocolo de comunicación.

Este proyecto nos hizo investigar mucho, ya que con la mayoría de los dispositivos electrónicos proporcionados no estábamos familiarizados, algunos sabíamos cómo funcionaban, pero no los habíamos utilizado, de esta manera obtuvimos más conocimientos.

Referencias bibliográficas y virtuales.

Aníbal Ollero Baturone. Robótica Manipuladores y robots móviles. Boixareu editors.

José Andrés Somolinos Sánchez. Avances en robótica y visión por computadora. Ediciones de la universidad de Castilla- La mancha. Ed III.

John J. Craig. Robotica. Editorial Pearson. 3ra. Edicion.

Jose Manuel Angulo Usategui, Susana Romero Yesa, Ignacio Angulo Martínez. Introducción a la robótica: principios teóricos, construcción y programación de un robot educativo. Editorial Paraninfo.

Rafael Lahoz-Beltrá. Bioinformática, simulación, vida artificial e inteligencia artificial. Ediciones Diaz de Santos, S.A.

Luis Álvarez Munarriz. Fundamentos de inteligencia artificial. Universidad de Murcia. Secretariado de publicaciones, Ed II.

http://robots-argentina.com.ar/Comunicacion_busI2C.htm

http://www.superrobotica.com/S320122.htm

http://www.matpic.com/esp/matlab/interfaz_grafica.html

http://www.mathworks.com/discovery/matlab-gui.html

http://www.personal.rdg.ac.uk/~shsmchlr/.../nnrob.pdf

http://www.scienceprog.com/neural-networks-and-artificial-intelligence-in-robotics/

http://www.nasa.gov/vision/universe/roboticexplorers/robots_like_people.html

http://www.worldscientific.com/worldscibooks/10.1142/3774