ROBOT QUE RESUELVE LABERINTOS "SHOCOBOT"

Domínguez Ramírez, Sergio Osbaldo; Tovilla Blanco, Kenya; Yamasaki Aguilar Miriam
Lenguajes de Interfaz.
Instituto Tecnológico de Tuxtla Gutiérrez, Carretera Panamericana Km. 1080, C.P. 29050, Apartado Postal: 599, Tuxtla Gutiérrez, Chiapas.

osbaldo93dominguez@gmail.com; miri-xion@hotmail.com ; ktb1205@hotmail.com

Resumen— El presente proyecto consiste en la construcción de un robot el cual tiene como propósito resolver laberintos; este robot está programado en arduino UNO, por consiguiente, se tiene incluido en los materiales de dicho robot la placa arduino UNO, así como también un Shield L293D, con tres sensores ultrasónicos HC_SR04, cuatro motores DC para el funcionamiento de las cuatro ruedas; también se incluye una batería de 12 Volts para el suministro de energía que necesita el robot para su adecuado funcionamiento y conteniendo un sensor de color TCS3200 el cual ayuda a que el robot se detenga cuando detecte un color en específico (en este caso es amarillo), avance en su camino y posteriormente se detiene.

Palabras clave: Robot, laberinto, sensores ultrasónicos, sensor de color, arduino UNO.

Abstract— This project involves the construction of a robot which aims to solve mazes; this robot is programmed in Arduino UNO, therefore, it has included in the robot's materials arduino UNO, as well as, a Shield L293D, with three HC_SR04 ultrasonic sensors, four DC motors to the operation of the four wheels; also includes a 12 Volts battery to the energy supply by the robot for its proper functioning and containing a color sensor TCS3200 which helps the robot stops when it detects a specific color (in this case is yellow), progress on its way and then stops.

Keywords: Robot, maze, ultrasonic sensors, color sensor, arduino UNO.

Introducción

En vista de que en los últimos tiempos se ha presentado la necesidad de

desarrollar tecnología que reemplace al ser humano principalmente cuando se

trabaja en áreas de alto riesgo o poco accesibles para las personas como son

encontrar averías en tuberías de oleoductos o en derrumbes, se ha visto la

necesidad de la utilización de robots y es aquí donde ingresa el manejo de

algoritmos de búsqueda o desplazamiento en diferentes tipos de áreas, así como

el manejo de sensores para reconocimiento del entorno en el cual el robot se

desplazará.

Una de las diferentes formas a las que se puede dar aplicación es con un robot que es capaz de resolver laberintos, siendo un robot autónomo que por medio de sensores detecta cualquier obstáculo o pared que un laberinto tenga. Es así como este proyecto trata acerca de "Shocobot" un robot que soluciona laberintos, así como la descripción de sus componentes físicos y de la lógica que dicho robot contiene.

Materiales

Este robot capaz de solucionar un laberinto, el cual está compuesto por los siguientes materiales:

- ⇨ 3 sensores ultrasónicos HC_SR04.
- ⇨ 1 sensor de color TCS3200.
- ⇨ 1 Motor Shield L293D.
- ⇨ 1 Placa de arduino UNO.
- ⇨ 4 Motores DC.
- ⇨ 1 Batería de 12 Volts.
- ⇨ 1 Base de madera de 17 cm x 17 cm
- ⇨ 4 ruedas
- ⇨ 1 Mini Protoboard
- ⇨ 1 switch

Desarrollo y diseño

Durante el diseño del prototipo se tuvieron en cuenta varias características que fueron detectadas en la implementación exitosa del mismo. Entre las cuestiones que se analizaron en las etapas de diseño se pueden mencionar: la base (chasis), los movimientos a realizar por el robot y los materiales electrónicos (que se mencionarán más adelante).

A continuación, se realiza una descripción acerca de los componentes electrónicos que contiene el robot nombrado “Shocobot”:

Componentes Electrónicos:

- ***Sensor ultrasónico HC_SR04***

  El HC-SR04 es un sensor ultrasónico de bajo costo que no sólo puede detectar si un objeto se presenta, como un sensor PIR (Pasiva Infra red Sensor), sino que también puede sentir y transmitir la distancia al objeto.

  Tienen dos transductores, básicamente, un altavoz y un micrófono.

  Ofrece una excelente detección sin contacto (remoto) con elevada precisión y lecturas estables en un formato fácil de usar.

  El funcionamiento no se ve afectado por la luz solar o el material negro como telémetros ópticos (aunque acústicamente materiales suaves como telas pueden ser difíciles de detectar).

  La velocidad del sonido en el aire (a una temperatura de 20 °C) es de

  343 m/s. (por cada grado centígrado que sube la temperatura, la velocidad del sonido aumenta en 0,6 m/s)

Características técnicas:

Los módulos incluyen transmisores ultrasónicos, el receptor y el circuito de control.

Número de pines:

- VCC: Alimentación +5V (4.5V min – 5.5V max)
- TRIG: Trigger entrada (input) del sensor (TTL)
- ECHO: Echo salida (output) del Sensor (TTL)
- GND

>Corriente de reposo: < 2mA

>Corriente de trabajo: 15mA

>Ángulo de medición: 30º

>Ángulo de medición efectivo: < 15º

>Detección de 2cm a 400cm o 1" a 13 pies (Sirve a más de 4m, pero el fabricante no garantiza una buena medición).

> "Resolución" La precisión puede variar entre los 3mm o 0.3cm.

>Dimensiones: 45mm x 20mm x 15mm

>Frecuencia de trabajo: 40KHz

Funcionamiento:

1. Enviar un Pulso "1" de al menos de 10uS por el Pin Trigger (Disparador).
2. El sensor enviará 8 Pulsos de 40KHz (Ultrasonido) y coloca su salida Echo a alto (seteo), se debe detectar este evento e iniciar un conteo de tiempo.
3. La salida Echo se mantendrá en alto hasta recibir el eco reflejado por el obstáculo a lo cual el sensor pondrá su pin Echo a bajo, es decir, terminar de contar el tiempo.
4. Se recomienda dar un tiempo de aproximadamente 50ms de espera después de terminar la cuenta.
5. La distancia es proporcional a la duración del pulso y puedes calcularla con las siguiente formula (Utilizando la velocidad del sonido = 340m/s):

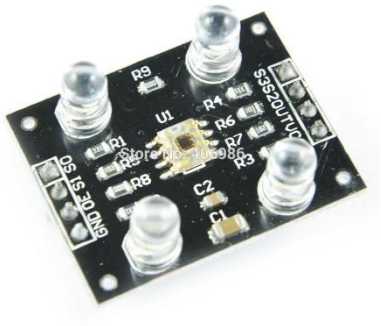
Cómo Funciona el Sensor Ultrasónico PING

Distancia en cm (centímetros) = Tiempo medido en us x 0.017

- ***Sensor de color TCR3200:***

El sensor de color TCR3200 es un detector de color completo, incluye un chip sensor TAOS TCR3200 RGB y 4 LEDs blancos de ángulo amplio. El TCR3200 puede detectar y medir una gama casi ilimitada de colores visibles. Las aplicaciones incluyen clasificación por color, sensor de luz ambiente, calibración y coincidencia de colores, para nombrar algunas pocas.

El TCR3200 tiene una matriz de detectores, cada uno de ellos con filtros rojo, verde y azul o sin filtro (clear). Los filtros de cada color se distribuyen uniformemente a lo largo de la matriz para eliminar el sesgo de ubicación entre los colores. El dispositivo tiene un oscilador que produce una salida de onda cuadrada cuya frecuencia es proporcional a la intensidad del color elegido.



Especificación

-Alimentación: 2.7V a 5.5V

-Interfaz: Digital TTL

-Conversión de alta resolución, de intensidad de luz a frecuencia

-Color y rango de frecuencia de salida programables

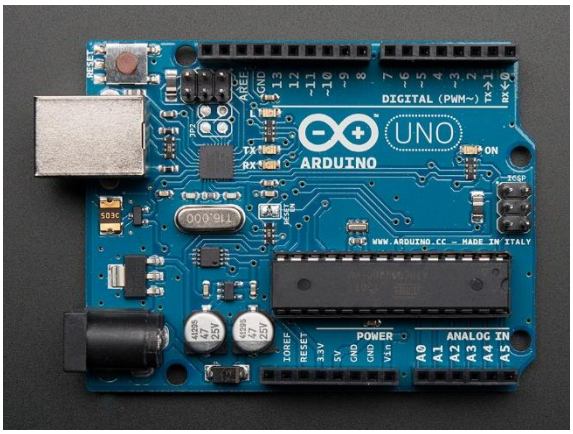
-Función de apagado

-Se comunica directamente al microcontrolador

-Tamaño: 28.4x28.4mm

- ***Arduino UNO:***

Arduino es una placa con un microcontrolador de la marca Atmel y con toda la circuitería de soporte, que incluye, reguladores de tensión, un puerto USB (En los últimos modelos, aunque el original utilizaba un puerto serie) conectado a un módulo adaptador USB-Serie que permite programar el microcontrolador desde cualquier PC de manera cómoda y también hacer pruebas de comunicación con el propio chip.

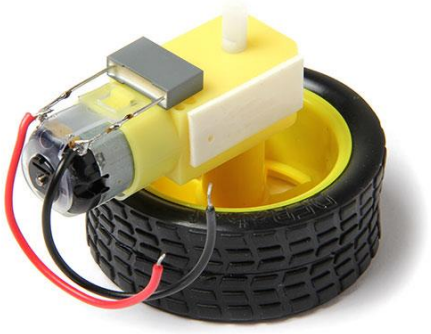


Un arduino dispone de 14 pines que pueden configurarse como entrada o salida y a los que puede conectarse cualquier dispositivo que sea capaz de transmitir o recibir señales digitales de 0 y 5 V. También dispone de entradas y salidas analógicas. Mediante las entradas analógicas podemos obtener datos de sensores en forma de variaciones continuas de un voltaje. Las salidas analógicas suelen utilizarse para enviar señales de control en forma de señales PWM.

Arduino UNO es la última versión de la placa, existen dos variantes, la Arduino UNO convencional y la Arduino UNO SMD. La única diferencia entre ambas es el tipo de microcontrolador que montan.

- ***Motores DC:***

Un motor de corriente directa, o motor CD, convierte la energía eléctrica en energía mecánica. Es uno de los dos tipos de motores básicos: el otro tipo es el de corriente alterna, o motor CA. Entre los motores CD, está el derivado, el de serie, el compuesto y el de imán permanente.



Función

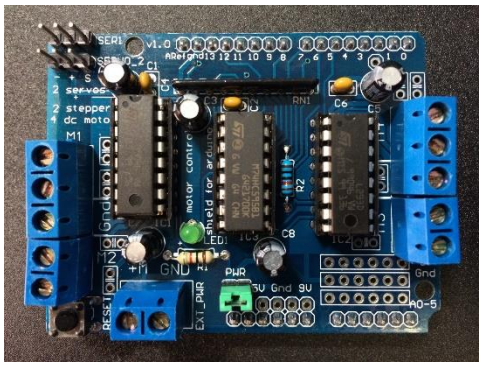
Un motor CD consiste en un estator, una armadura, un rotor y un colector con escobillas. La polaridad opuesta entre dos campos magnéticos dentro del motor hace que gire. Los motores CD son el tipo más simple de motor y se utilizan en electrodomésticos, como las máquinas de afeitar eléctricas, y en ventanas eléctricas de automóviles.

Operación básica de un motor CD

Un motor CD está equipado con imanes, ya sean permanentes o bobinas electromagnéticas, que producen un campo magnético. Cuando la corriente pasa a través de la armadura, también conocida como bobina o alambre, ubicada entre los polos norte y sur del imán, el campo generado por la armadura interactúa con el campo del imán y genera torsión. En un motor CD, el imán forma el estator, la armadura se ubica en el rotor y el colector alterna la corriente entre una bobina y la otra. El colector conecta la fuente de energía estacionaria a través del uso de escobillas o varas conductoras. Además, los motores CD operan con una velocidad fija o un voltaje fijo y no existe división.

- ***Motor Shield L293D:***

  El motor shield con dos L293D permite controlar 4 cargas inductivas + 2 servos conectados al timer dedicado de alta resolución de Arduino para un control sin bíter. Para el caso de motores, en cada canal se puede controlar tanto la velocidad como la dirección de giro independientemente. Posibilita el control desde proyectos sencillos hasta de media complejidad.



  Características:

  - Incorpora 2 circuitos integrados L293D proporcionando 4 puentes-H completos
  - Protección contra sobre carga térmica
  - Diodos de protección contra voltajes inversos generados por las cargas inductivas
  - 4 canales (M1, M2, M3 y M4) para controlar igual número de cargas inductivas como motores DC, solenoides, relés, ó 2 motores paso a paso unipolares o bipolares de entre 4.5 V y 25 V
  - En el caso de motores es posible controlar tanto su velocidad como su dirección de giro
  - Control de velocidad de motores DC con resolución de 8 bits
  - Control de motores paso a paso con 1 y 2 bobinas en modo sencillo, doble (mayor torque), intercalado (medio paso) y micro pasos
  - 2 conectores independientes para controlar 2 servos pequeños de 5 V (Pines digitales 9 y 10)
  - Corriente máxima continua en cada una de las salidas M1, M2, M3 y M4: 600 mA (0.6 A)
  - Corriente máxima pico no repetitivo en cada una de las salidas M1, M2, M3 y M4: 1.2 A
  - Resistores de pull-down que mantienen los motores deshabilitados durante el encendido
  - Requiere fuente externa para la alimentación de las cargas en las salidas M1, M2, M3 y M4. Puede empleada la fuente externa conectada a la placa Arduino o a la shield con selección mediante el jumper "PWR"
  - Bloques de terminales de tornillo para calibres 18 a 26 AWG
  - Botón de reset para la placa Arduino
  - Compatibilidad probada con Arduino Diecimila, Duemilanove, UNO y Mega 1280 & 2560

Componentes Físicos (Diseño del robot):

Como todo prototipo de un robot necesita de un diseño, es decir, el material por el que estará compuesto. Es así como este robot está hecho de madera la cual se realizaron dos bases: la primera es el chasis del robot y la segunda es una parte donde están ubicados demás componentes electrónicos de dicho robot.

Chasís

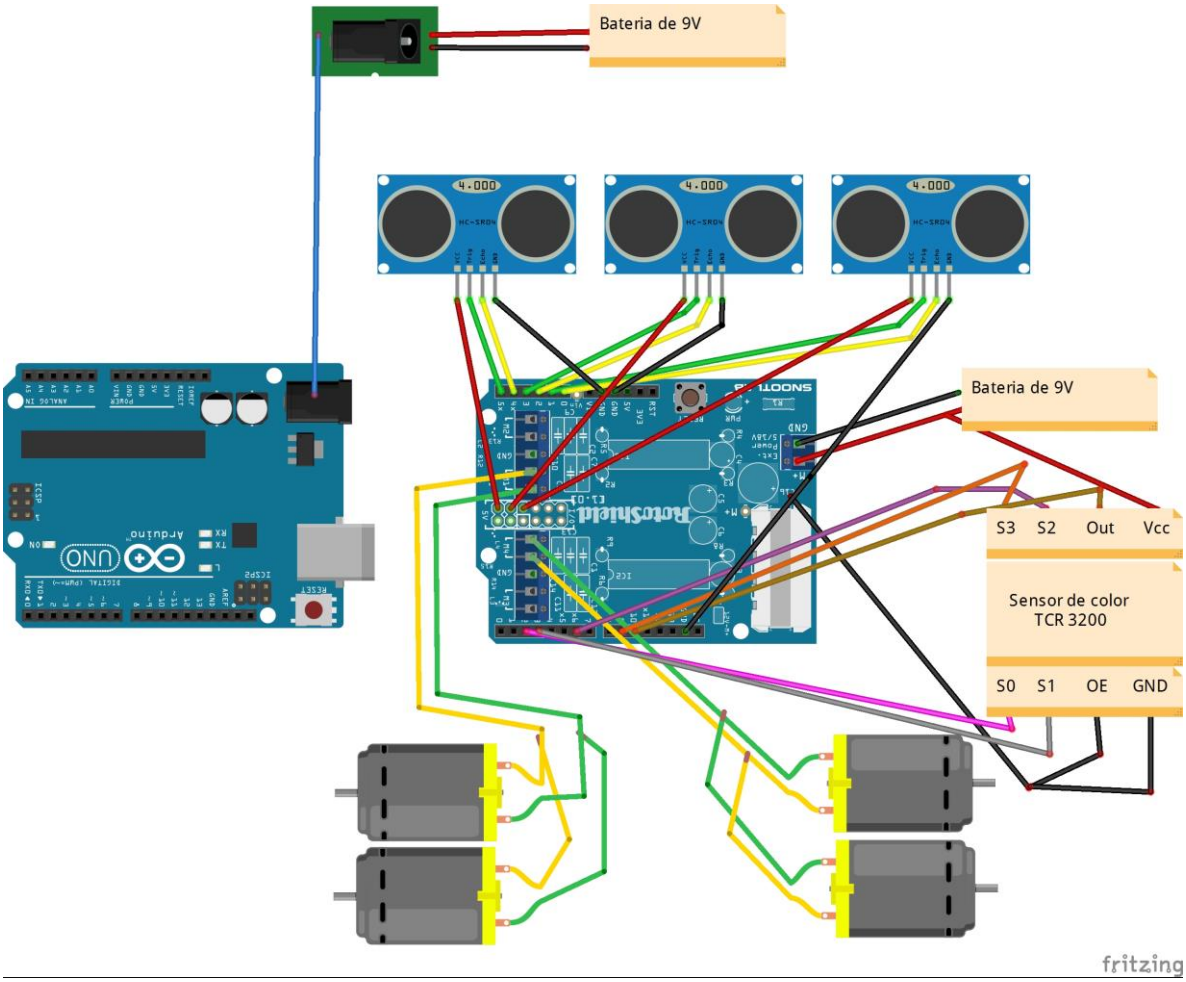
El diseño del chasís es un factor importante en el éxito o no de la implementación del robot, dado que según su forma permitirá realizar desplazamientos con mayor o menor exactitud al recorrer el laberinto.

El robot está compuesto de un chasís de madera de forma rectangular con medidas de 15 cm de ancho por 17 cm de largo, medida sumamente adecuada para el buen desplazamiento y/o movimiento del mismo; en este chasís incluye las uniones de los motores DC con las ruedas, así como también se tiene incrustado el sensor de color TCS3200 por la parte de abajo para que pueda detectar el color que se tenga al inicio del recorrido del laberinto. En la parte de arriba, se tienen la placa arduino UNO insertado con el motor Shield, un mini protoboard, una batería de 12 Volts y por consiguiente las conexiones hechas entre el motor shield, la batería con el mini protoboard, para que dicho robot funcione y comience a realizar los movimientos correspondientes.

Para la siguiente base o chasís (como se le quiera nombrar) está hecha igualmente de madera solamente con el detalle de tener la forma de una mesa, se realizó de esta forma para darle altura a los demás componentes que hacen falta colocar en el robot, además de colocar los sensores ultrasónicos en una altura adecuada y así detecte los obstáculos que haya en el momento de hacer el recorrido del laberinto.

Conexiones:

Después de conseguir los materiales necesarios para la construcción del robot, se procede a realizar las conexiones que tendrá cada elemento electrónico ya mencionados anteriormente, este esquema fue realizado en el programa fritzing:



Algoritmo:

Una vez conseguido el armado de las conexiones de “Shocobot” ahora se realiza la parte del entorno de la programación en Arduino, ya que es la parte lógica donde el robot tiene que ir haciendo los movimientos necesarios para que pueda resolver correctamente el laberinto.

Para ello, se propuso el siguiente algoritmo:

Para que el robot pueda solucionar el camino adecuado de un laberinto, se tienen tres direcciones que debe seguir: Frente, izquierda y derecha, por esta razón se colocaron los tres sensores ultrasónicos. Después de haber definido las tres direcciones, ahora se tiene como punto de referencia la medición de la distancia, siendo esta distancia la forma en que los sensores detectan qué tan cerca están de un obstáculo o pared del laberinto y así poder realizar los movimientos adecuados para evitarlos y finalmente llegar a terminar dicho laberinto.

Cada sensor obtiene la medida de la distancia en que esté, tanto de frente o adelante, izquierda y derecha. Una vez medida la distancia, ahora el robot elige el camino a seguir el cual está dado por un rango de medidas definidas; si la distancia medida a la derecha está demasiado cerca, es decir, si una pared del lado derecho se encuentra muy cerca del robot, entonces se alinea hacia la izquierda y se centra, en caso contrario, si está cercano al lado izquierdo, entonces se alinea a la derecha e igualmente se centra. Si no se encuentra muy cerca de los dos lados, entonces el robot sigue su camino.

Al igual para detectar los lados que están muy cercanos al robot, también se realiza la cuestión de los giros con rangos específicos de las distancias de cada lado y frente, ya que al cumplir con cada rango definido de distancias el robot retrocede, hace un giro ya sea del lado izquierdo o derecho y sigue hacia adelante. El robot se detendrá hasta detectar el color asignado (en este caso es el negro); al igual desde el inicio del recorrido, al detectar el color (amarillo) el robot queda inmóvil hasta los cinco segundos y comienza con la medición de las distancias en cada lado para comenzar a recorrer el laberinto para así llegar finalmente a la meta.

Código:

A continuación, se muestra la parte lógica del robot, la cual es la codificación que se realizó acorde el algoritmo mencionado anteriormente y en el entorno de programación Arduino 1.6.7:

```
//declaración de librerías

#Include <AFMotor.h> //librería para control de los motorreductores

#include <NewPing.h> //librería para el control del SR04 sensor ultrasónico


//declaración de pines y medidas

#define ECHO_PIN_ADELANTE A5 //el pin echo (el que escucha y mide: micrófono) del sensor delantero se define con el pin A5 del motor shield

#define TRIG_PIN_ADELANTE A4 //el pin trigger (el que comienza la medición y dispara el pulso: altavoz) del sensor delantero se define con el pin A4 del motor shield

#define ECHO_PIN_DERECHA A3 //el pin echo del sensor derecho se define con el pin A3 del motor shield

#define TRIG_PIN_DERECHA A2 //el pin trigger del sensor derecho se define con el pin A2 del motor shield

#define ECHO_PIN_IZQUIERDA A1 //el pin echo del sensor izquierdo se define con el pin A1 del motor shield

#define TRIG_PIN_IZQUIERDA A0 //el pin trigger del sensor izquierdo se define con el pin A0 del motor shield

#define DISTANCIA_MAXIMA 400 //se define la distancia máxima de medición de los sensores ultrasónicos

#define MAXIMA_VELOCIDAD 90 //se define la velocidad máxima que pueden obtener los motorreductores

//inicializaciones de funcionamiento de los sensores

//con estas sentencias, con estos constructores, se establece la configuración de trabajo de cada sensor

NewPing sonar_adelante (TRIG_PIN_ADELANTE, ECHO_PIN_ADELANTE, DISTANCIA_MAXIMA); //se inicializa la operación "sonar_adelante" usando TRIG_PIN_ADELANTE como el ping trigger del sensor delantero, ECHO_PIN_ADELANTE como el ping echo de ese sensor y DISTANCIA_MAXIMA como la distancia máxima al operar

NewPing sonar_derecha (TRIG_PIN_DERECHA, ECHO_PIN_DERECHA, DISTANCIA_MAXIMA); //se inicializa la operación "sonar_derecha" usando TRIG_PIN_DERECHA como el ping trigger del sensor derecho, ECHO_PIN_DERECHA como el ping echo de ese sensor y DISTANCIA_MAXIMA como la distancia máxima al operar

NewPing sonar_izquierda (TRIG_PIN_IZQUIERDA, ECHO_PIN_IZQUIERDA, DISTANCIA_MAXIMA); //se inicializa la operación "sonar_izquierda" usando TRIG_PIN_IZQUIERDA como el ping trigger del sensor izquierdo, ECHO_PIN_IZQUIERDA como el ping echo de ese sensor y DISTANCIA_MAXIMA como la distancia máxima al operar


//inicialización de funcionamiento de los motorreductores

//con estos constructores, se establece la configuración de trabajo de los motorreductores, los cuales fueron
```

conectados en paralelo, evitando las 2 sentencias más

AF_DCMotor motores1(1, MOTOR12_1KHZ); //se declaran los motores1, trabajando en el canal 1 del motor shield, usando una frecuencia de 1KHz

AF_DCMotor motores2(4, MOTOR34_1KHZ); //se declaran los motores2, trabajando en el canal 2 del motor shield, usando una frecuencia de 1KHz

//declaración de variables

int distancia_adelante=0; //variable que guarda la medida de distancia del sensor delantero

int pared_adelante=0; //variable que guarda la medida de distancia del sensor delantero una vez se comprueba que no haya errores en la lectura

int distancia_derecha=0; //variable que guarda la medida de distancia del sensor derecho

int pared_derecha=0; //variable que guarda la medida de distancia del sensor derecho una vez se comprueba que no haya errores en la lectura

int distancia_izquierda=0; //variable que guarda la medida de distancia del sensor izquierdo

int pared_izquierda=0; //variable que guarda la medida de distancia del sensor izquierdo una vez se comprueba que no haya errores en la lectura

boolean adelante=false; //variable auxiliar para verificar si el robot ya comenzó a avanzar hacia adelante

int velocidad=0; //variable para aumentar la velocidad

boolean sigue_trabajando=true; //variable auxiliar para que el robot siga trabajando o no

boolean mas_adelante=false; //variable auxiliar para avanzar un poco más antes de realizar un cambio de dirección

enum Colores {RED, GREEN, BLUE, CLEAR}; //variable que guarda todos los estados de Colores

#define s2Pin 6 //s2Pin se define como el pin 6 del Arduino

#define s3Pin 9 //s3Pin se define como el pin 9 del Arduino

#define outPin 10 //outPin se define como el pin 10 del Arduino

int rojo=0, verde=0, azul=0, limpio=0; //variables que guardan las lecturas para cada componente rojo=RED, verde=GREEN, azul=BLUE, CLEAR=limpio

//configuración inicial del programa

void setup () {

pinMode (s2Pin, OUTPUT); //s2Pin queda en modo de salida

pinMode (s3Pin, OUTPUT); //s3Pin queda en modo de salida

pinMode (outPin, INPUT); //outPin queda en modo de entrada

Serial.begin(9600); //Se inicia el serial a esa velocidad, es para ayudar a verificar las mediciones

delay (1600); //una espera de 1.6 segundos, ya que, más los segundos que tarda para iniciar todo el programa, se da un total de espera de 5 segundos, el tiempo de espera reglamentario en el concurso

}

```
//función de ciclo infinito

void loop () {

  medir (); //se llama a dicha función

  elegirCamino (); //se llama a la función respectiva

}


//declaración de funciones

//función para medir la distancia de los 3 sensores y guardarlos en variables para su posterior uso y comparación

void medir () {

  distancia_adelante=sonar_adelante.
ping_cm (); //se obtiene la distancia en cm del sensor delantero

  if (¡distancia_adelante! =0) {//se coloca esta condición para evitar errores, ya que, si ocurre un error en la medición de un sensor, devuelve un valor 0; de esta forma, solo se pasará de esta condición si se mide exitosamente

    pared_adelante=distancia_adelante; //si no hay errores, la variable pared_adelante obtiene la medida almacenada en distancia_adelante

    Serial.print("\adelante:"); //las siguientes dos sentencias son para imprimir en el serial la medida guardada en pared_adelante

    Serial.print(pared_adelante);

  }

  distancia_derecha=sonar_derecha. ping_cm (); //se obtiene la distancia en cm del sensor derecho

  if (¡distancia_derecha! =0) {//condición para evitar errores de medición

    pared_derecha=distancia_derecha; //si no hay errores, la variable pared_derecha obtiene la medida almacenada en distancia_derecha

    Serial.print("\nDerecha:"); //las siguientes dos sentencias son para imprimir en el serial la medida guardada en pared_derecha

    Serial.print(pared_derecha);

  }

  distancia_izquierda=sonar_izquierda.
ping_cm (); //se obtiene la distancia en cm del sensor izquierdo

  if (¡distancia_izquierda! =0) {//condición para evitar errores de medición

    pared_izquierda=distancia_izquierda; //si no hay errores, la variable pared_izquierda obtiene la medida almacenada en distancia_izquierda

    Serial.print("\nIzquierda:"); //las siguientes dos sentencias son para imprimir en el serial la medida guardada en pared_izquierda

    Serial.print(pared_izquierda);

  }

}


//función para elegir el camino al cual ir después de obtener las medidas de los 3 sensores

void elegirCamino () {

  if(pared_adelante>10) {//si la distancia en pared_adelante es mayor a 10, el robot avanza hacia adelante

    mas_adelante=true; //se vuelve verdadera esta variable para que cuando elija un giro el robot, avance un poco más antes de realizar el cambio de dirección

    if(adelante==false) {//si adelante es igual a false, significa que el robot está quieto y ahora se va a mover, acelerando y moviéndose hacia adelante con la función moverAdelante


moverAdelante (); //se llama a dicha función
```

```
    //aquí se cumple una condición o la otra, jamás ambas

    if(pared_derecha>0 && pared_derecha<=7) {//si la distancia en pared_derecha es mayor a 0 y menor o igual a 7, significa que el robot está muy pegado a la pared derecha, por lo que se llama a la función alinearIzquierda para alejarlo y centrarlo

     alinearIzquierda (); //se llama a la función respectiva

     Serial.print("\n Alinear izquierda"); //se imprime esta oración en el serial para saber qué hace el robot

    }

    else if(pared_izquierda>0 && pared_izquierda<=7) {//si la distancia en pared_izquierda es mayor a 0 y menor o igual a 7, significa que el robot está muy pegado a la pared izquierda, por lo que se llama a la función alinearDerecha para alejarlo y centrarlo

     alinearDerecha (); //se llama a dicha función

     Serial.print("\n Alinear derecha"); //se imprime esta oración en el serial para saber qué hace el robot

    }

    giros_recto (); //se llama a esta función después de los ifs

    Serial.print("\nmoverAdelante"); //se imprime esta oración en el serial para saber se entró en la condición de adelante igual a false

   }

   else {//si adelante es igual a true, significa que el robot ya se está moviendo

    motores1.run(FORWARD); //los motores solo siguen avanzando, debido a que ya cuentan con velocidad


motores2.run(FORWARD);
```

```
    //aquí se cumple una condición o la otra, jamás ambas

    if(pared_derecha>0 && pared_derecha<=7) {//se realizan los mismos alineamientos y acciones vistos en el código superior

      alinearIzquierda ();

      Serial.print("\n Alinear izquierda");

    }

    else if(pared_izquierda>0 && pared_izquierda<=7) {

      alinearDerecha ();

      Serial.print("\n Alinear derecha");

    }

    giros_recto ();

    Serial.print("\nsigueAdelante"); //se imprime esta oración en el serial para saber se entró en la condición de adelante igual a true

   }

  }


  if (pared_adelante<=10 && pared_adelante>0) {//si pared_adelante es menor o igual a 10 y mayor a 0, se entra en esta condición

   //aquí se cumple una condición o más

   if(pared_derecha>0 && pared_derecha<=8) {//se realizan alineamientos iguales a los vistos anteriormente, con la diferencia que ahora pared_derecha y pared_izquierda sean iguales o menores a 8

      alinearIzquierda ();

      Serial.print("\n Alinear izquierda");

   }
```

```
 if(pared_izquierda>0 && pared_izquierda<=8) {
      alinearDerecha ();
      Serial.print("\n Alinear derecha");
   }

   //si pared_adelante es mayor a 0 y menor a 9, y pared_derecha es mayor a 0 y menor a 13 y pared_izquierda es mayor a 0 y menor a 13, se entra en esta condición
   if((pared_adelante>0 && pared_adelante<9) && (pared_derecha>0 && pared_derecha<13) && (pared_izquierda>0 && pared_derecha<13)) {
     motores1.run(BACKWARD); //el robot se mueve hacia atrás
     motores2.run(BACKWARD);
     delay (400); //mantiene ese movimiento por 0.4 segundos
     motores1.run(BACKWARD); //el robot gira hacia la izquierda
     motores2.run(FORWARD);
     delay (1160); //mantiene el giro por 1.16 segundos
     motores1.run(FORWARD); //se mueve hacia adelante
     motores2.run(FORWARD);
     delay (200); //sigue con ese movimiento por 0.2 segundos
     motores1.run(RELEASE); //se detiene el robot
     motores2.run(RELEASE);
     color (); //se llama a dicha función
     compararColor (); //se llama a la función respectiva
     if(sigue_trabajando==false) {//si sigue_trabajando es igual a false, se entra en esta condición
       while (1) {//se accede a un ciclo infinito
        motores1.run(RELEASE); //el robot se detiene hasta que se reinicie el programa
        motores2.run(RELEASE);
       }
     }
     else {//si sigue_trabajando es igual a true, se entra en esta condición
       pared_adelante=0; //estas variables toman el valor de 0 y false, respectivamente
       pared_derecha=0;
       pared_izquierda=0;
       mas_adelante=false;
       adelante=false;
       medir (); //se obtienen nuevas medidas
     }
     Serial.print("\n Encerrado"); //se imprime esta oración en el serial para saber que el robot se encontró encerrado por 3 paredes
   }

   if(pared_adelante>0 && pared_adelante<=6) {//si pared_adelante es mayor a 0 y menor o igual a 6, significa que el robot está muy cerca de una pared adelante, por lo que para evitar colisionar se entra en esta condición
     moverAtras (); //se llama a dicha función: retrocede un poco
     detener (); //se llama a esta función: se detiene
     adelante=false; //adelante se vuelve false para indicar que el robot se detuvo y tiene que volver a agarrar velocidad al avanzar hacia adelante
```

```
    giros (); //se llama a la función correspondiente: compara las distancias para saber a dónde ir ahora

    Serial.print("\n No colision"); //se imprime esta oración para saber que se entró en esta condición

   }

 }


 //se reinician las variables, para evitar arrastrar valores

 pared_adelante=0;

 pared_derecha=0;

 pared_izquierda=0;

 mas_adelante=false;

}


//función para elegir a donde ir antes de colisionar, después de que el robot retrocedió y se detuvo

void giros () {

 //aquí se cumple una condición o más

 //si pared_adelante es mayor o igual a 14, y es mayor a pared_izquierda y es mayor a pared_derecha, se entra a esta condición

 if(pared_adelante>=14 && pared_adelante>pared_izquierda && pared_adelante>pared_derecha) {

   moverAdelante (); //se llama a esta función: el robot se mueve hacia adelante

   Serial.print("\nVa hacia adelante G"); //se imprime esta oración en el serial para saber que el robot se mueve hacia adelante, habiendo usado la función giros, por eso la G al final

 }

 //si pared_derecha es mayor o igual a 14, y es mayor a pared_izquierda y es mayor a pared_adelante, se entra a esta condición

 if(pared_derecha>=14 && pared_derecha>pared_izquierda && pared_derecha>pared_adelante) {

   girarDerecha (); //se llama a dicha función: el robot gira a la derecha y sigue de frente

   Serial.print("\nVa hacia la derecha G"); //se imprime la oración en el serial para saber que el robot gira a la derecha, habiendo usado la función giros, por eso la G al final

 }

 //si pared_izquierda es mayor o igual a 14, y es mayor a pared_derecha y es mayor a pared_adelante, se entra a esta condición

 if(pared_izquierda>=14 && pared_izquierda>pared_derecha && pared_izquierda>pared_adelante) {

   girarIzquierda (); //se llama la función respectiva: el robot gira a la izquierda y sigue de frente

   Serial.print("\nVa hacia la izquierda G"); //se imprime la oración en el serial para saber que el robot gira a la izquierda, habiendo usado la función giros, por eso la G al final

 }

 //si pared_adelante es mayor o igual a 14 y es igual a pared_derecha, se entra en esta condición

 if(pared_adelante>=14 && pared_adelante==pared_derecha) {

   moverAdelante (); //el robot se mueve hacia adelante

   Serial.print("\nVa hacia adelante G"); //se muestra la oración en el serial

 }

 //si pared_adelante es mayor o igual a 14 y es igual a pared_izquierda, se entra en esta condición

 if(pared_adelante>=14 && pared_adelante==pared_izquierda) {
```

```
  moverAdelante (); //el robot se mueve hacia adelante

  Serial.print("\nVa hacia adelante G"); //se muestra la oración en el serial

 }

 //si pared_izquierda es mayor o igual a 14 y es igual a pared_derecha, se entra en esta condición

 if(pared_izquierda>=14 && pared_izquierda==pared_derecha) {

  girarIzquierda (); //el robot gira a la izquierda y sigue de frente

  Serial.print("\nVa hacia la izquierda G"); //se muestra la oración en el serial

 }

}


//función para elegir a donde ir aún en movimiento

void giros_recto () {

 //aquí se cumple una condición o más

 if(mas_adelante==true) {//si mas_adelante es true, se entra en esta condición

  delay (300); //el robot se moverá hacia adelante durante 0.3 segundos más

  Serial.print("\nUn poco más"); //se imprime esta oración en el serial para saber si avanza un poco más

 }

 //si pared_adelante es mayor a 15, y es mayor a pared_izquierda y es mayor a pared_derecha, se entra a esta condición

 if(pared_adelante>15 && pared_adelante>pared_izquierda && pared_adelante>pared_derecha) {

  Serial.print("\nVa hacia adelante GR"); //se imprime esta oración en el serial para saber si se sigue hacia adelante, habiendo usado la función giros_recto, por eso la GR al final

  delay (800); //se avanza hacia adelante durante 0.8 segundos

 }

 //si pared_derecha es mayor a 15, y es mayor a pared_izquierda y es mayor a pared_adelante, se entra a esta condición

 if(pared_derecha>15 && pared_derecha>pared_izquierda && pared_derecha>pared_adelante) {

  girarDerecha (); //se llama a esta función: el robot gira a la derecha y sigue de frente

  Serial.print("\nVa hacia la derecha GR"); //se imprime esta oración en el serial para saber si gira hacia la derecha, habiendo usado la función giros_recto, por eso la GR al final

 }

 //si pared_izquierda es mayor a 15, y es mayor a pared_derecha y es mayor a pared_adelante, se entra a esta condición

 if(pared_izquierda>15 && pared_izquierda>pared_derecha && pared_izquierda>pared_adelante) {

  girarIzquierda (); //se llama a dicha función: el robot gira a la izquierda y sigue de frente

  Serial.print("\nVa hacia la izquierda GR"); //se imprime esta oración en el serial para saber si gira hacia la izquierda, habiendo usado la función giros_recto, por eso la GR al final

 }

 //si pared_adelante es mayor a 15 y es igual a pared_derecha, se entra en esta condición

 if(pared_adelante>15 && pared_adelante==pared_derecha) {

  delay (1200); //se sigue hacia adelante por 1.2 segundos

  Serial.print("\nVa hacia adelante GR"); //se imprime la oración en el serial

 }

 //si pared_adelante es mayor a 15 y es igual a pared_izquierda, se entra en esta condición
```

```
 if(pared_adelante>15 && pared_adelante==pared_izquierda) {

   delay (1200); //se sigue hacia adelante por 1.2 segundos

   Serial.print("\nVa hacia adelante GR"); //se imprime la oración en el serial

  }

  //si pared_izquierda es mayor a 15 y es igual a pared_derecha, se entra en esta condición

  if(pared_izquierda>15 && pared_izquierda==pared_derecha) {

   girarIzquierda (); //el robot gira a la izquierda y sigue de frente

   Serial.print("\nVa hacia la izquierda GR"); //se imprime la oración en el serial

  }

}


//función para mover el robot hacia adelante, estando detenido inicialmente

void moverAdelante () {

  motores1.run(FORWARD); //los motores se mueven hacia adelante

  motores2.run(FORWARD);

  for (velocidad=0; velocidad<MAXIMA_VELOCIDAD; velocidad+=2) {//comienzan a ganar velocidad

   motores1.setSpeed(velocidad); //los motores aceleran en cuestión al valor proporcionado

   motores2.setSpeed(velocidad);

   delay (5); //esperan 0.005 segundos entre cada aceleración

  }

  adelante=true; //adelante se vuelve true

}

//función para mover el robot hacia atrás

void moverAtras () {

  motores1.run(BACKWARD); //los motores se mueven hacia atrás

  motores2.run(BACKWARD);

  delay (700); //mantiene el movimiento durante 0.7 segundos

}


//función para girar a la derecha

void girarDerecha () {

  motores1.run(FORWARD); //los motores1 giran hacia adelante

  motores2.run(BACKWARD); //los motores2 giran hacia atrás

  delay (1185); //se mantienen esos movimientos durante 1.185 segundos

  motores1.run(FORWARD); //los 4 motores se mueven hacia adelante

  motores2.run(FORWARD);

  delay (1340); //continuan el movimiento durante 1.34 segundos

}


//función para girar a la izquierda

void girarIzquierda () {

  motores1.run(BACKWARD); //los motores1 giran hacia atrás

  motores2.run(FORWARD); //los motores2 giran hacia adelante

  delay (1185); //se mantienen esos movimientos durante 1.185 segundos

  motores1.run(FORWARD); //los 4 motores se mueven hacia adelante

  motores2.run(FORWARD);

  delay (1340); //continuan el movimiento durante 1.34 segundos
```

```
}


//función para detenerse
void detener () {
  motores1.run(RELEASE); //los motores se detienen
  motores2.run(RELEASE);
}


//función para alinear a la derecha; se aleja de la pared izquierda y luego se centra en el camino
void alinearDerecha () {
  motores1.run(FORWARD); //se gira hacia la derecha
  motores2.run(BACKWARD);
  delay (700); //continúa girando durante 0.7 segundos
  motores1.run(FORWARD); //se avanza hacia adelante
  motores2.run(FORWARD);
  delay (650); //sigue avanzando por 0.65 segundos
  motores1.run(BACKWARD); //se gira a la izquierda
  motores2.run(FORWARD);
  delay (340); //se mantiene girando durante 0.34 segundos
  motores1.run(FORWARD); //vuelve a avanzar hacia adelante
  motores2.run(FORWARD);
}


//función para alinear a la izquierda; se aleja de la pared derecha y luego se centra en el camino
void alinearIzquierda () {
  motores1.run(BACKWARD); //se gira a la izquierda
  motores2.run(FORWARD);
  delay (700); //sigue girando durante 0.7 segundos
  motores1.run(FORWARD); //se mueve hacia adelante
  motores2.run(FORWARD);
  delay (650); //continúa avanzando por 0.65 segundos
  motores1.run(FORWARD); //gira a la derecha
  motores2.run(BACKWARD);
  delay (340); //mantiene el giro por 0.34 segundos
  motores1.run(FORWARD); //vuelve a moverse hacia adelante
  motores2.run(FORWARD);
}


//función para obtener los valores de cada componente del color que se encuentre debajo del robot
void color () {
  rojo=leerColor(RED); //la variable rojo almacena el valor que le devuelve la función leerColor respecto a RED
  verde=leerColor(GREEN); //la variable verde almacena el valor que le devuelve la función leerColor respecto a GREEN
  azul=leerColor(BLUE); //la variable azul almacena el valor que le devuelve la función leerColor respecto a BLUE
  limpio=leerColor(CLEAR); //la variable limpio almacena el valor que le devuelve la función leerColor respecto a CLEAR
}
```

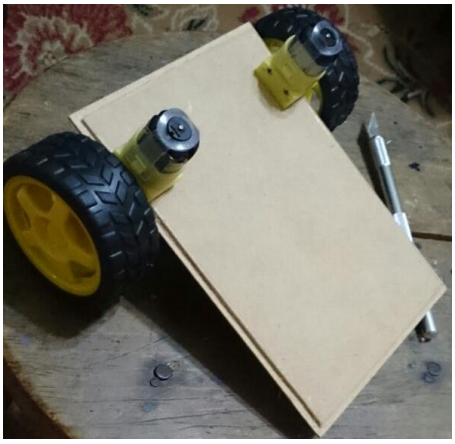
```
//función que lee los colores y devuelve un valor entre 0 y 255

byte leerColor (byte color)

{

  switch(color) //dependiendo de lo que reciba en la variable color, se dará un caso específico

  {

    case RED: //en caso de RED, se configuran los leds a manera de leer el valor del componente rojo

      digitalWrite (s2Pin, LOW); //se apagan ambos leds que corresponden a s2Pin y s3Pin

      digitalWrite (s3Pin, LOW);

      break;


    case GREEN: //en caso de GREEN, se configuran los leds a manera de leer el valor del componente verde

      digitalWrite (s2Pin, HIGH); //se encienden ambos leds que corresponden a s2Pin y s3Pin

      digitalWrite (s3Pin, HIGH);

      break;


    case BLUE: //en caso de BLUE, se configuran los leds a manera de leer el valor del componente azul

      digitalWrite (s2Pin, LOW); //se apaga el led que corresponde a s2Pin

      digitalWrite (s3Pin, HIGH); //se enciende el led que corresponde a s3Pin

      break;


    case CLEAR: //en caso de CLEAR, se configuran los leds a manera de leer el color sin filtro alguno

      digitalWrite (s2Pin, HIGH); //se enciende el led que corresponde a s2Pin

      digitalWrite (s3Pin, LOW); //se apaga el led que corresponde a s3Pin

      break;

  }


  return map (pulseIn (outPin, HIGH), 30, 2500, 255, 0); //devuelve un valor entre el rango de 255 a 0, en base al valor de pulseIn (outPin, HIGH) que va de 30 hasta 2500

}


//función para comparar los valores de rojo, verde, azul y limpio

void compararColor () {

  //si rojo es igual a 2, y verde es igual a 2, y azul es igual a 2 y limpio es mayor a 100, se entra a esta condición

  if (rojo==2 && verde==2 && azul==2 && limpio>100) {

    sigue_trabajando=false; //sigue_trabajando se vuelve false, lo que significa que el robot encontró el color negro debajo de él, indicando que ha llegado a la meta

  }

}
```

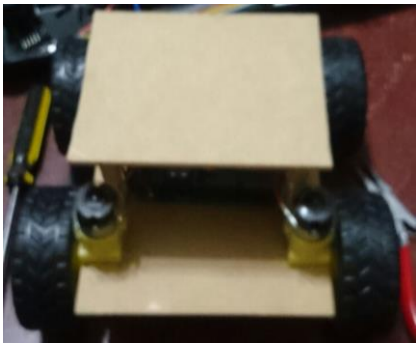
Resultados:

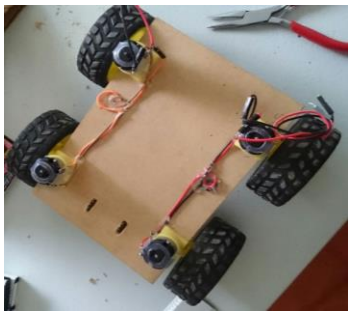
Como producto final se obtuvo con éxito la realización del robot “Shocobot” ya que se hicieron las pruebas necesarias para su correcto funcionamiento, siendo que con anterioridad la formulación que se hizo del algoritmo no era lo suficientemente óptimo para que “Shocobot” realizara sus movimientos de la manera más efectiva, es así como se llegó con el algoritmo final ya dicho anteriormente.

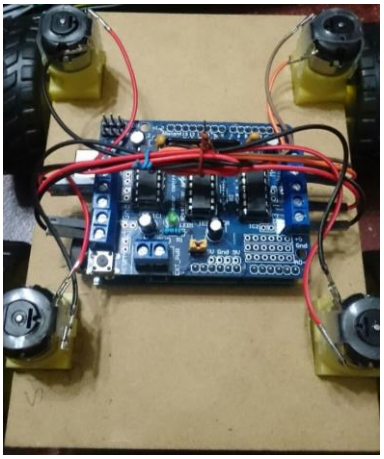
Es así como “Shocobot” resulta hacer sus movimientos de manera muy efectiva hasta llegar a solucionar un laberinto.

En las siguientes imágenes se puede observar el proceso que se efectuó desde el armado de los materiales hasta quedar totalmente terminado.





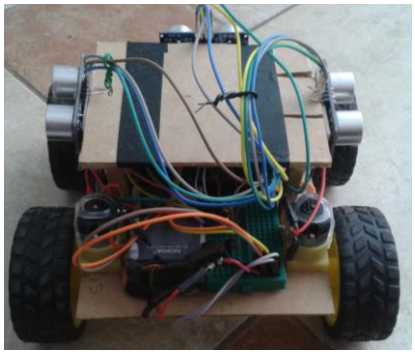


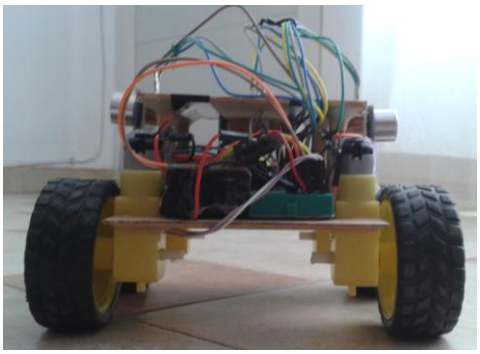


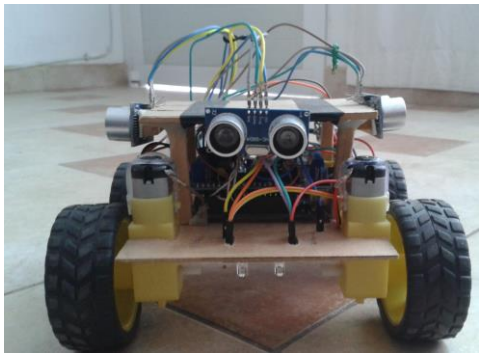
Instituto Tecnológico de Tuxtla Gutiérrez

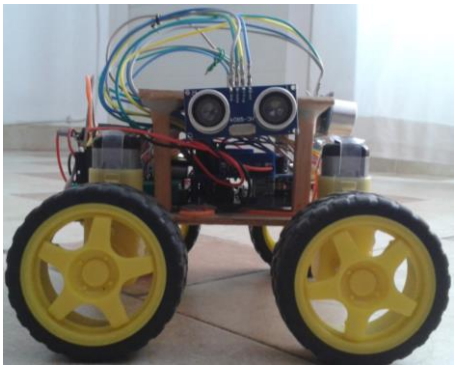
Ciencia y Tecnología con sentido Humano

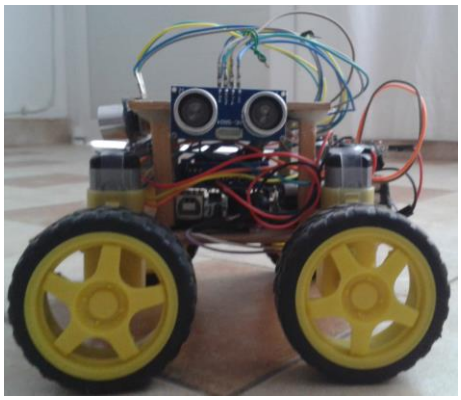
Ingeniería en Sistemas Computacionales

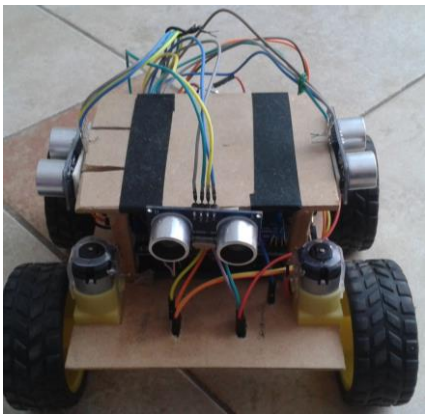


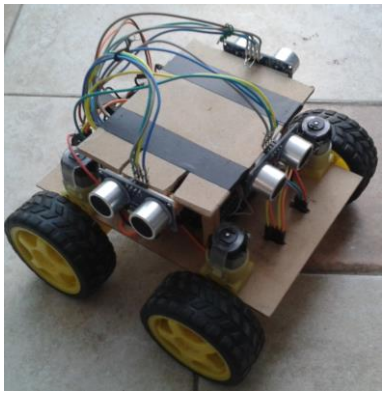












Aplicación del robot:

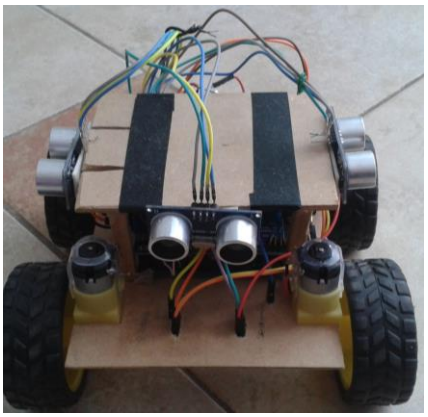
Los robots móviles poseen diversas aplicaciones en la industria, tales como la inspección, la exploración, el bodegaje y control de producción a distancia. Poseen diversas aplicaciones en el transporte en general. También, en el contexto de robots de servicio, los robots móviles se han venido introduciendo en forma de aspiradoras para el hogar (Trilobites Electro lux), mascotas (Atibo) e incluso humanoides (Asomo).

La principal aplicación de los robots tiene lugar en la industria, donde se necesitan repetir tareas, seguir cierto recorrido donde las personas no pueden llegar y estos robots llegarían a realizar ese trabajo.

Un robot está programado para realizar los mismos movimientos con una buena precisión, por lo que es perfecto para aplicaciones industriales; aunque todos los campos de la industria son susceptibles de emplear diferentes tipos de robots para determinadas operaciones.

La competencia de robot laberinto tiene como fin la construcción de un robot móvil que debe ser completamente autónomo y auto contenido, no pudiendo estar conectado a

ningún aparato externo como ordenadores o fuentes de alimentación, ni ningún otro dispositivo.



Conclusión:

Finalmente, como conclusión se tiene que el robot que resuelve laberintos es un proyecto realmente competitivo y ambicioso, ya que estos dos aspectos llegan a aumentar dependiendo en qué ámbito se le quiere dar aplicación y también las modificaciones o mejoras que se desee realizar para lograr el objetivo principal de dicho aparato.

Como todo proyecto se tienen fallas, correcciones, pruebas y mejoras, ya que tanto la formulación de los algoritmos como la lógica que debe seguir dicho robot, así como también las conexiones electrónicas fueron los puntos principales que se fueron mejorando cada vez que se continuaba con la realización y ahora exitosa culminación de "Shocobot".

Este proyecto fue hecho de investigaciones y análisis sumamente exhaustivo, ya que al ser la primera experiencia de realizar un robot que soluciona laberintos estuvo lleno de satisfacciones y claramente obtención de nuevos conocimientos adquiridos que forman parte de nuestra formación profesional.

Bibliografía:

https://www.researchgate.net/publication/27557693_Diseno_y_construccion_de_un_robot_movil_que_pueda_desplazarse_dentro_de_un_laberinto

http://www.unipamplona.edu.co/unipamplona/portalIG/home_88/recursos/01general/11112013/documento_laberinto.pdf

http://bkargado.blogspot.mx/2013/09/todosobrehc-sr04.html

http://electronilab.co/tienda/sensor-de-distancia-de-ultrasonido-hc-sr04/

https://www.openhacks.com/page/productos/id/95/title/TCS3200-Sensor-de-Color#.V0Nmn-RniHM

http://www.prometec.net/motorshieldv1/

http://www.electronicoscaldas.com/shields-escudos-arduino/511-shield-de-motores-para-arduino-shd-mstepper.html

http://www3.gobiernodecanarias.org/medusa/ecoblog/ralvgon/files/2013/05/Caracter%C3%ADsticas-Arduino.pdf

http://www.ehowenespanol.com/definicion-motor-corriente-directa-sobre_55810/