Robot autónomo para recorrer un laberinto

Fernando Sanchez[1] fernandosanchezutn@yahoo.com.ar, Omar E. Rodriguez[1] rodriguezomar@argentina.com
Héctor Hugo Mazzeo [2] hhmgvm@yahoo.com, José A. Rapallini [2] josrap@gmail.com.ar

[1] Alumnos del Trabajo Final de Aplicaciones en Tiempo Real, [2] Profesor -Director del Trabajo
[2] ***CODAPLI*** *(Proyecto Codiseño Hardware Software para Aplicaciones en Tiempo Real) – Departamento de Ingeniería en Sistemas de Información. Universidad Tecnológica Nacional. Facultad Regional La Plata, Calle 60 y 124, La Plata 1900, Argentina.*

Resumen **- El objetivo de este trabajo es presentar los detalles principales del diseño del hardware y software de un robot autónomo que es capaz de desplazarse a través de un laberinto prediseñado, siguiendo trayectorias rectilíneas con desviaciones a 90 o 180º ante la detección de un borde o pared del laberinto, cuando se interpone en su camino recto.**

El diseño y la construcción de un prototipo de este robot se realizaron como parte del proyecto de grado de la materia "Aplicaciones en Tiempo Real" correspondiente al quinto año de la carrera de Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional - Facultad Regional La Plata.

Palabras clave: Robot autónomo, laberinto, sensores ultrasónicos, servomotores, placa Arduino.

I. INTRODUCCIÓN

El proyecto para la creación de un robot autónomo capaz de recorrer los pasillos de un laberinto surgió como el desafío de presentar un trabajo de desarrollo novedoso y ambicioso para lo que es un proyecto de una materia de grado.

El objetivo fue desarrollar e implementar un robot autónomo capaz de recorrer un laberinto, con una distancia entre paredes de los pasillos de un mínimo de 20 centímetros y un máximo de 300 centímetros.

Se diseñó un prototipo basado en "Arduino", plataforma electrónica abierta (código y hardware abiertos) que permite la creación de prototipos basados en software y hardware flexibles [1].

En particular, para la construcción del sistema de control del robot se utilizó una placa Arduino Uno [2], sensores ultrasónicos para medir distancias y servos modificados para lograr el desplazamiento del robot dentro del laberinto (fig. 1).

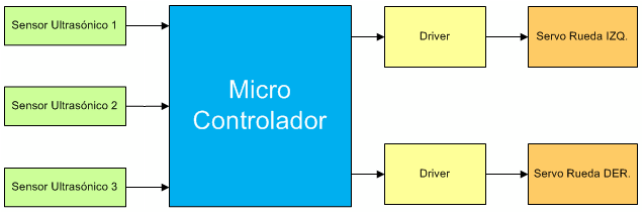

Fig. 1. Diagrama de bloques del robot

Se utilizaron tres sensores ultrasónicos de los cuales dos fueron dispuestos en los laterales del robot para medir las distancias a las paredes del pasillo del laberinto y un sensor frontal para detectar un obstáculo (por ejemplo,elfinal de un pasillo).

Respecto al código de programación -que comanda los distintos componentes electrónicos de entrada y salida a través de la placa- se utilizó la interfaz de desarrollo nativa de Arduino.

II. DESARROLLO Y DISEÑO

Al momento del diseño del prototipo se tuvieron en cuenta varias características que afectarían a la implementación exitosa del mismo.

Entre las cuestiones que se analizaron en la etapa de diseño se pueden mencionar: el chasis, los movimientos a realizar por el robot, los tipos de sensores, los actuadores y la alimentación necesaria para el funcionamiento de los componentes electrónicos.

A. Chasis

El diseño del chasis 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.

Para este caso se diseñó un chasis de forma circular que luego, acompañado con otras características de implementación, nos permitiría realizar movimientos más exactos (utilizamos otros aspectos en el diseño para lograr que el centro de giro de la circunferencia sea exactamente el centro del chasis). Los elementos utilizados para la construcción se basaron en CD's reciclados los cuales aportan una circunferencia exacta para tal fin.

B. Movimientos

La trayectoria que realiza el robot es en línea recta con giros de 90º o 180° cuando detecta un obstáculo.

La tracción utilizada es diferencial (fig. 2), lo que hace que los giros sean más sencillos y además permite realizar los movimientos sin más espacio que el que está ocupando (gira sobre su propio eje).

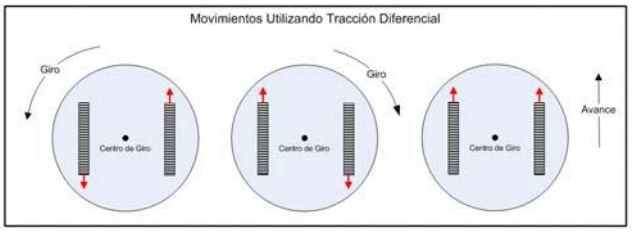

Fig. 2. Tracción diferencial

Otro aspecto del diseño que se definió a priori fue la ubicación de las ruedas, dado que esto también impacta en la ubicación del centro de giro. En el prototipo se ubicaron en forma equidistante al centro.

Para las ruedas se utilizaron mini CD´s (2 por rueda para aumentar la superficie de rozamiento contra el suelo) y como eje se le añadió uno de los brazos que viene con el servo.

C. Sensores

Los sensores de distancia utilizados son ultrasónicos HC-SR04 (fig. 3). Estos son capaces de detectar objetos próximos y calcular la distancia a la que se encuentra en un rango de 2 a 450 cm.

Funcionan enviando y recibiendo pulsos ultrasónicos y contienen toda la electrónica encargada de realizar la medición. Su utilización se basa en enviar un pulso de arranque y medir el tiempo del pulso de retorno luego de la reflexión en el objeto.

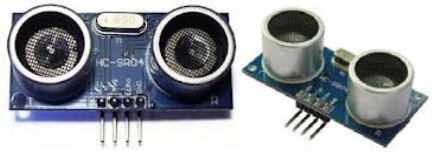

Fig. 3. Sensores ultrasónicos HC-SR04

Se utilizaron 3 sensores similares para detectar los obstáculos del robot, uno ubicado en el frente y dos en los laterales (fig. 4).

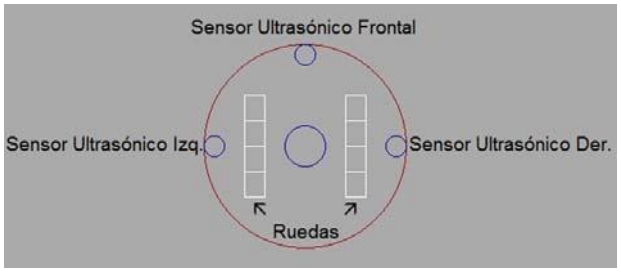

Fig. 4. Ubicación de los sensores vista desde arriba

D. Motores

Para el movimiento de las ruedas se usaron servomotores. Se modificaron los topes mecánicos y se reemplazó el potenciómetro por una resistencia fija, haciéndolos funcionar de forma similar a un motor de corriente contínua.

El tamaño reducido del servo MG90S [3] lo hace ideal para la construcción del prototipo (fig. 5).

Fig. 5. MicroServo MG90S

Los servomotores hacen uso de la modulación por ancho de pulsos (PWM) para controlar la dirección o posición de los motores de corriente continua. La mayoría trabaja en la frecuencia de los cincuenta hercios, así las señales PWM tendrán un periodo de veinte milisegundos.

La electrónica dentro del servomotor responderá al ancho de la señal modulada. Si los circuitos dentro del servomotor reciben una señal de entre 0,5 a 1,4 milisegundos, se moverá en sentido horario; entre 1,6 a 2 milisegundos se moverá en sentido antihorario y 1,5 milisegundos representa un estado neutro para los servomotores, es decir sin movimiento.

E. Alimentación

Para lograr un prototipo autónomo se utilizaron dos fuentes de alimentación independientes: un pack de pilas recargables AA para la placa Arduino y una batería de 9 V para la alimentación de los servos.

Las cuatro pilas recargables proporcionan un voltaje de 4 x 1,2 v = 4.8 volts, valor que es suficiente para alimentar la placa.

Para la alimentación de los servos que proporcionaran el movimiento a las ruedas del robot, se utilizó una batería de 9 volts.

Tanto el pack de pilas como la batería se ubicaron en la parte más baja del robot para que el centro de gravedad se acerque lo más posible al suelo, logrando así gran estabilidad.

III. DETALLES DEL PROTOTIPO

Además de los elementos mencionados anteriormente para su construcción, se agregaron dos puntos de apoyo como ayuda al movimiento de desplazamiento que realiza el robot, con una pequeña rueda con giro libre de 360º y un punto de apoyo frontal, para evitar la pérdida de estabilidad (figs. 6 y 7).

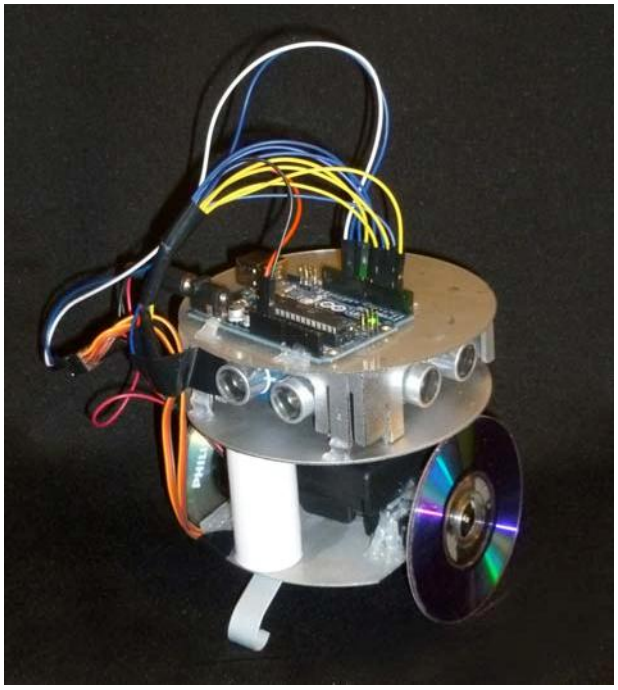

Fig. 6. Vista frontal del prototipo

Fig. 7. Vista lateral (izq.) y vista posterior (der.) del prototipo

IV. APLICACIÓN DE CONTROL DEL ROBOT

Se comenzó realizando un diagrama funcional de todo el sistema (fig. 8). Puede observarse allí el algoritmo utilizado para controlar los servomotores a partir de la información obtenida de los sensores S_1 (sensor laterial izquierdo), S_2 (sensor frontal) y S_3 (sensor lateral derecho). Se utiliza la siguiente convención: si el sensor $S_i = 1$ significa que se está sensando una pared próxima, por lo tanto no se habilita el giro hacia ese lado y si $S_i = 0$, se habilita el giro correspondiente.

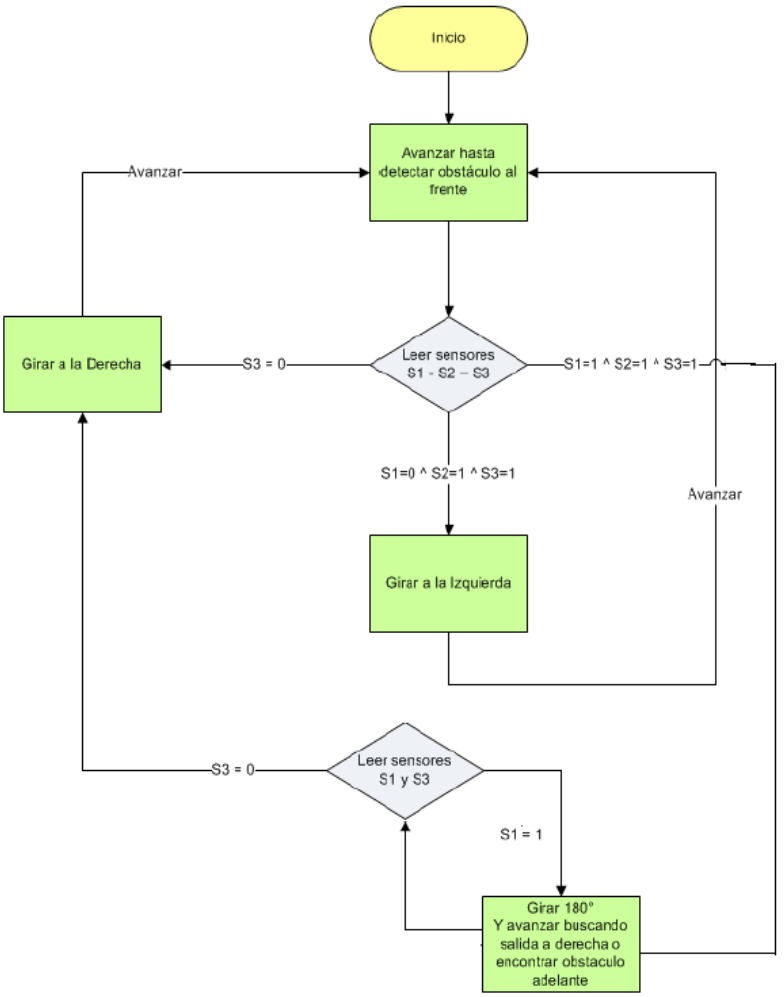

Fig. 8. Diagrama funcional del robot

Luego, aplicando la metodología de diseño de sistemas de tiempo real estudiados en la materia, se realizó la simulación funcional del sistema completo mediante un grafo de Petri [4] (Fig. 9).

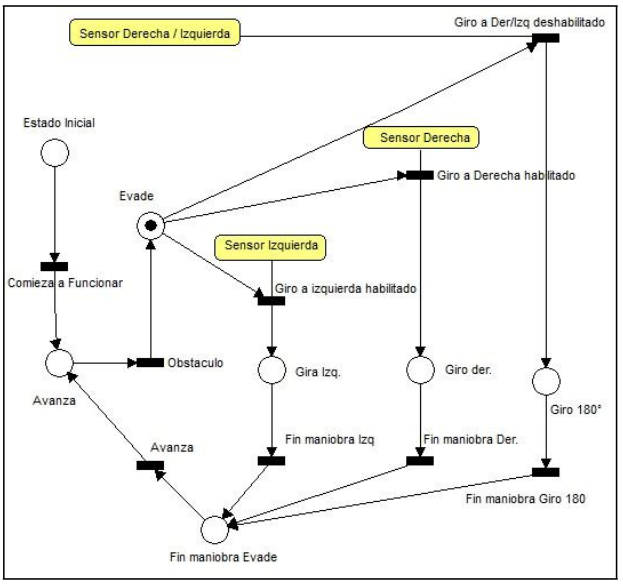

Fig. 9. Diagrama de Petri de control del robot

Los resultados del análisis estructural y de la ejecución de la red fueron utilizados para realizar la programación de la aplicación en el entorno de código abierto provisto por Arduino (fig. 10).

Una vez compilado y probado desde el entorno de programación, el código resultante se transfiere a la placa y se ejecuta en ella. Un bootloader inicia y ejecuta el código cargado en la memoria, logrando el funcionamiento autónomo del robot.

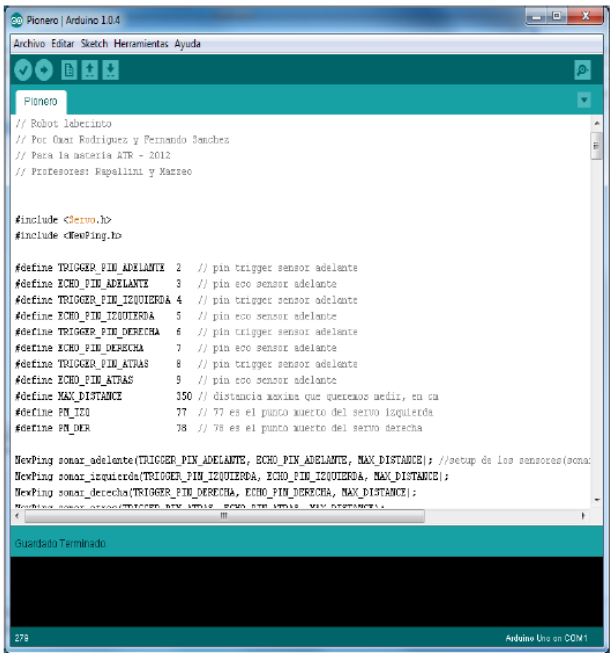

Fig. 10. Entorno de desarrollo Arduino

Se muestra a continuación el código fuente implementado, la definición de constantes, variables e inicialización del programa y rutina principal:

```
#include <Servo.h>
#include <NewPing.h>

#define TRIGGER_PIN_ADELANTE  2  // pin trigger sensor adelante
#define ECHO_PIN_ADELANTE     3   // pin eco sensor adelante
#define TRIGGER_PIN_IZQUIERDA 4 // pin trigger sensor adelante
#define ECHO_PIN_IZQUIERDA    5   // pin eco sensor adelante
#define TRIGGER_PIN_DERECHA   6 // pin trigger sensor adelante
#define ECHO_PIN_DERECHA      7   // pin eco sensor adelante
#define TRIGGER_PIN_ATRAS     8   // pin trigger sensor adelante
#define ECHO_PIN_ATRAS        9   // pin eco sensor adelante
#define MAX_DISTANCE          350  // distancia maxima que queremos
medir, en cm
#define PM_IZQ              77          // 77 punto muerto del servo izq
#define PM_DER              78          // 78 punto muerto del servo der

NewPing sonar_adelante(TRIGGER_PIN_ADELANTE,
ECHO_PIN_ADELANTE, MAX_DISTANCE); //setup de los sensores
NewPing sonar_izquierda(TRIGGER_PIN_IZQUIERDA,
ECHO_PIN_IZQUIERDA, MAX_DISTANCE);
NewPing sonar_derecha(TRIGGER_PIN_DERECHA,
ECHO_PIN_DERECHA, MAX_DISTANCE);
```

```
//Variables
unsigned int pingSpeed = 50;   // intervalo de medición, en ms.
unsigned long pingTimer = 100; // var. mantiene en que tiempo(en
milisegundos), desde que arranco el programa, hacemos los próxima
medicion.
int d_pared_izq = 0;          // Distancia a pared izquierda
int d_pared_der = 0;          // Distancia a pared derecha
int d_pared_adelante = 0;  // Distancia a pared adelante
boolean obstaculo = false; // Obstáculo delante?
boolean chocado = false;     // Si me choque algo
boolean fin_giro_derecha = true;   // fin girar 90 grados a la derecha
boolean fin_giro_izquierda = true;  // fin girar 90 grados a la izquierda
int estado = 0;               // Estados del robot
                              // 0 = parado
                              // 1 = avanzando
                              // 2 = girando izquierda
                              // 3 = girando derecha


Servo servo_der,servo_izq;     // Los servos que usamos de motor
int pos_servo_der = PM_DER;    // guardan la posición de cada servo
int pos_servo_izq = PM_IZQ;
int inc_izq = 10; // Incrementos para mover las ruedas
int inc_der = 9;int d_atras_giro = 0;              // Auxiliar para controlar giro de 90
grados. int d_lateral_izquierdo = 0;   // Auxiliar que mantiene la distancia.
void setup()
{
servo_der.attach(10);    //relaciono cada servo con un pin del arduino
servo_izq.attach(11);
servo_der.write(PM_DER);     //lo dejo quieto durante 8 segundos antes de arrancar
servo_izq.write(PM_IZQ);
delay(4000);    // Retraso para ubicar robot en laberinto.
}
void loop()
{
medir();
controlar();
mover(estado);
}
```

Las funciones medir(), controlar() y mover() realizan las tareas de leer la información provista por los sensores de proximidad y de ejecutar las acciones de desplazamiento a través de los servomotores.

```
void medir(){
    int adelante,atras,izquierda,derecha;
    if(millis() >= pingTimer){    //  mido si han pasado pingTimer ms.
      pingTimer += pingSpeed;     // Incremento pingTimer
      adelante = sonar_adelante.ping_cm();
      if(adelante != 0)
        d_pared_adelante = adelante;    //uso estas auxiliares para
filtrar medidas de 0 de los sensores.(errores)
      izquierda = sonar_izquierda.ping_cm();
      if(izquierda != 0)
        d_pared_izq = izquierda;
      derecha = sonar_derecha.ping_cm();
      if(derecha != 0)
        d_pared_der = derecha;
      atras = sonar_atras.ping_cm();
      if(atras != 0)
        d_pared_atras = atras;
      if(d_pared_adelante <= 14 && d_pared_adelante >0) //no
considero el 0 por si hay un error de lectura(dan 0).
        obstaculo = true;
      if(obstaculo == true && d_pared_adelante >= 45)
obstaculo = false;
      if(d_pared_adelante > 0 && d_pared_adelante < 4)
        chocado = true;
}
```

```
void controlar(){

   if(estado == 0 && obstaculo == false)   //lo pongo a andar, solo se
usa en el inicio
      estado = 1;

   if(estado == 1 && obstaculo == true){   //Voy avanzando y
encuentro un obstaculo -> giro para el lado que tenga mas espacio.
      if(d_pared_izq > 22)
        estado = 2;
      if(d_pared_der > 22)
        estado = 3;
   }

   if(d_pared_izq > 30 && d_pared_der > 30)
     estado = 0; //Encontre la salida

   if(chocado == true){
     retroceder();
     chocado = false;
     estado = 1;
   }
```

```
void mover(int est){
  switch (est){
    case 0:
      servo_der.write(PM_DER);    //lo dejo quieto
      servo_izq.write(PM_IZQ);
      break;
    case 1:
      avanzar();
      break;
    case 2:
      girar_izquierda();
      break;
    case 3:
      girar_derecha();
      break;
    case 4:
      retroceder();
    case 5:
      media_vuelta();
      break;
    case 6:
      retroceder();
      break;
  }
}
```

Se realizaron pruebas de funcionamiento dentro de un laberinto construido con paredes de 250 mm de ancho y 150 mm de alto, suficientes para que los sensores ultrasónicos puedan detectarlas. Luego de algunos intentos fallidos se logró un funcionamiento aceptable, dentro de los límites de precisión previstos. En la fig. 11 se muestra una fotografía de la etapa de prueba del robot dentro de un pasillo del laberinto.

Fig. 11. Prueba del prototipo en un laberinto prediseñado

V. CONCLUSIONES

En este trabajo se realizó una investigación y análisis exhaustivo del funcionamiento de un robot autónomo que fuera capaz de recorrer un laberinto con cualquier tipo de formato, a condición de que los trayectos fueran rectos y con desviaciones a 90º en ambos sentidos.

Es de destacar el armado y el logro exitoso del funcionamiento de un prototipo de muy bajo costo, ya que se construyó utilizando componentes reciclados y de descarte para el chasis, lo que favoreció el abaratamiento en el desarrollo del proyecto.

Como contraparte, este hecho genera dificultades para lograr un movimiento perfectamente rectilíneo, lo cual debe ser corregido constantemente mediante el software de control. Una construcción más refinada del sistema de tracción, mejoraría este aspecto.

V. TRABAJOS FUTUROS

Este robot sienta un precedente en los proyectos de este tipo, diseñado y construido como parte de los proyectos desarrollados en la materia de grado y quedará totalmente disponible para continuar adaptando y mejorando su diseño en el transcurso del tiempo.

Uno de los detalles a mejorar tiene que ver con el sistema de tracción, haciéndolo más estable. Otro aspecto a continuar evaluando, es el relacionado con los sensores ultrasónicos, probando distintas alternativas de ubicación o inclusive utilizando otros de mayor sensibilidad.

Se prevée que también sea utilizado por los alumnos de cursos posteriores para realizar prácticas de programación, mejorando los algoritmos de control (por ejemplo utilizando técnicas de control mediante inteligencia artificial).

REFERENCIAS

[1] Arduino: www.arduino.cc

[2] Arduino Uno: www.arduino.cc/en/Main/arduinoBoardUno

[3] Servos: www.servodatabase.com/servo/towerpro/mg90

[4] ¨Simuladores de Redes de Petri¨:
www.frlp.utn.edu.ar/materias/atr/