**INFORME PROYECTO 3**

**ROBOT AUTONOMO DE LABERINTOS**

**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 microcontroladores correspondiente al quinto semestre de la carrera de Ingeniería electrónica.

**Introducción**:

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 30 centímetros. Se diseñó un prototipo basado en “Arduino uno”, plataforma electrónica abierta (código y hardware abiertos) que permite la creación de prototipos basados en software y hardware flexibles.



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



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, el final 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 cuadrada 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 del cuadrado sea exactamente el centro del chasis). Los elementos utilizados para la construcción se basaron en váquela y chasis metálico los cuales aportan un cuadro 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, 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).

C. Ubicación de los sensores vista desde abajo, para el movimiento de los Motores 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 continua. La mayoría trabaja en la frecuencia de los cincuenta hercios, así las señales PWM tendrán un periodo de 8 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 lipo 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.

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

#include <NewPing.h>

#include <Servo.h>

#define TRIGGER\_PIN\_ADELANTE 2 // pin triggersensor adelante

#define ECHO\_PIN\_ADELANTE 3 // pin eco sensor adelante

#define TRIGGER\_PIN\_IZQUIERDA 4 // pin triggersensor adelante

#define ECHO\_PIN\_IZQUIERDA 5 // pin eco sensor adelante

#define TRIGGER\_PIN\_DERECHA 6 // pin triggersensor adelante

#define ECHO\_PIN\_DERECHA 7 // pin eco sensor adelante

#define TRIGGER\_PIN\_ATRAS 8 // pin triggersensor 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);//setup de los sensores

NewPing sonar\_derecha(TRIGGER\_PIN\_DERECHA,ECHO\_PIN\_DERECHA, MAX\_DISTANCE);//setup de los sensores

NewPing sonar\_atras(TRIGGER\_PIN\_ATRAS,ECHO\_PIN\_ATRAS, MAX\_DISTANCE);

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

int d\_pared\_atras = 0;

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 mover(int est);

void controlar();

void retroceder();

void avanzar();

void girar\_izquierda();

void girar\_derecha();

void media\_vuelta();

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);

}

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;

}

}

**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 económicos 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.