Avance proyecto 2 Seguidor de línea con arduino

Nathaly Sanchez Hincapie, Beymar Ruidiaz Martinez.
Universidad del Quindío, laboratorio general de electrónica, Microprocesadores

Armenia, Quindío, Colombia

ns7hincapie@gmail.com
beymar.ruidiaz.m@gmail.com

***Resumen*— Este es un avance del proyecto dos, el cual es la implementación un seguidor de línea con la tarjeta programable arduino. Este está constituido por una parte en el software Arduino, que es la parte programable o (nivel de control) y la parte de implementación y mecánica o (nivel físico), con sus respectivos componentes como : ruedas, motores, batería, puente H (L293D),etc.**

Palabras clave — . puente H, ruedas,batería.

I. INTRODUCCIÓN

Un seguidor de línea es un robot móvil que se clasifica en el grupo de la rama de la robótica móvil. Su tarea fundamental es el desplazamiento en un entorno ya sea conocido o desconocido, por ende es necesario que posea tres funciones fundamentales, la locomoción (nivel físico),la percepción (nivel sensorial) y la decisión (nivel de control).En este caso la pista va a presentar una trayectoria marcada la cual se le configura de manera que debe diferenciar el entorno en este caso, negra con un fondo blanco. Para poder lograr este objetivo, se implementó un sensor de infrarrojos de corto alcance (CNY70), por su propiedad de emisor de luz y un receptor siempre apuntando en la misma dirección.

II. OBJETIVOS

- Objetivo General:
  Diseñar un Robot seguidor de línea (en este caso negra) en el suelo, con código de C++ en el programa arduino.
- Objetivo especifico:
  1. Estudiar el funcionamiento de un Robot seguidor de línea y sus características.
  2. Despertar la creatividad de los alumnos para tener un buen desempeño en un desarrollo de análisis y tecnológico.
  3. Implementar un algoritmo, para realizar la toma de decisiones del Robot seguidor de línea.
  4. Construir la estructura adecuada para que se adapte y facilite el desplazamiento del robot en la pista.
  5. Aprender a interpretar lo que se tiene a nivel físico y transportarlo al software de simulación Proteus.

III. DESARROLLO

En base a lo anterior se implementa el esquemático en el software Proteus, para comprobar los integrados como el puente H (L293D) físicamente estuviera correcto. Para esto, se hace uso de la biblioteca simulino que permite tener acceso a un ARDUINO digital, la versión de Proteus que se implementa es la 8. Los Pasos que se siguieron fueron :

1. Abrir Isis Proteus.
2. Crear un nuevo esquemático.
3. Dar clic en el menú “Library” y espués en “Library Manager”.
4. Dar clic en el botón “Create library”.
5. En la pantalla que se abra, se arrastran los archivos descargados ahí.
6. Cerrar Proteus, para después volver a abrir y listo ya se podrá escoger las placas arduino que se deseen.

A partir de esto, se inicia un nuevo archivo para implementar el esquemático del seguidor de línea, se hace uso de ARDUINO Uno r3, el puente H (l293D) ayudará a controlar los motores de corriente continua, se tiene en cuenta el datasheet de este, y se realizaron las conexiones correctas.

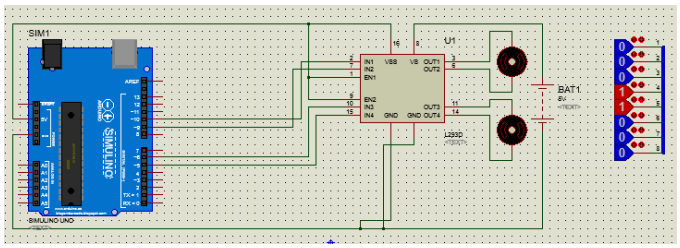

Figura 1. Esquemático del nivel físico del seguidor de línea.

A partir de esto se realiza el código para el buen funcionamiento del Puente H.

```
int izqA = 5;
int izqB = 6;
int derA = 9;
int derB = 10;
int vel = 255; // Velocidad de los motores (0-255)

void setup()  {
  pinMode(derA, OUTPUT);
  pinMode(derB, OUTPUT);
  pinMode(izqA, OUTPUT);
  pinMode(izqB, OUTPUT);
 }

void loop()  {
    analogWrite(derB, 0);  // Detiene los Motores
    analogWrite(izqB, 0);
    delay (500);
    analogWrite(derA, vel);  // Frente 2 segundos
    analogWrite(izqA, vel);
    delay (2000);

    analogWrite(derA, vel);  // Derecha 0,5 segundos
    analogWrite(izqA, 0);
    delay (500);

    analogWrite(derA, 0);    // Izquierda 0,5 segundos
```

Figura 2. Primera parte del código del seguidor de línea.

```
void loop()  {
    analogWrite(derB, 0);  // Detiene los Motores
    analogWrite(izqB, 0);
    delay (500);
    analogWrite(derA, vel);  // Frente 2 segundos
    analogWrite(izqA, vel);
    delay (2000);

    analogWrite(derA, vel);  // Derecha 0,5 segundos
    analogWrite(izqA, 0);
    delay (500);

    analogWrite(derA, 0);    // Izquierda 0,5 segundos
    analogWrite(izqA, vel);
    delay (500);

    analogWrite(derA, 0);    // Detiene los Motores
    analogWrite(izqA, 0);
    delay (500);
    analogWrite(derB, vel);  // Reversa 2 segundos
    analogWrite(izqB, vel);
    delay (2000);
}
```

Figura 3. Segunda parte del código del seguidor de línea.

Este código se realiza, para poder verificar que el integrado (L293D) realizara correctamente su funcionamiento, dándole un buen control manejo a los motores respectivos con sus llantas.

Una vez comprobado esto, se implementó el código para el control del robot seguidor de línea.

```
// Definimos los pines digitales que usaremos
int boton = 8;
int motor_derechaadelante = 11;   // Con este pin, dirigimos el motor derecho hacia adelante
int motor_izquierdaadelante = 12; // Con este pin, dirigimos el motor izquierdo hacia adelante
int motor_derechaatras = 4;    // Con este pin, dirigimos el motor derecho hacia atras
int motor_izquierdaatras = 2;  // Con este pin, dirigimos el motor izquierdo hacia atras
int pwm_izquierdo = 10;            // pin de manejo de velocidad del motor izquierdo
int pwm_derecho = 9;             // Pin de manejo de la velocidad del motor derecho
int ledOn = 6;
int ledPlay = 7;

// Pines analogos
int sensor_1 = 0;
int sensor_2 = 1;
int sensor_3 = 2;
int sensor_4 = 3;



// Variables iniciadas en cero
float proporcional = 0;
float proporcional_anterior= 0;
int posicion = 0;
int posicion_ultima = 0;
float integral = 0;
float derivativo = 0;
float pwm_valor = 0;
```

Figura 4. Primera parte del programa para el nivel de control del robot seguidor de línea.

En esta parte se inicia con la declaracion de las variables que se van a utilizar, definiendo el tipo de variable, es decir, en este caso int (entero), que quiere decir que todas las variables serán implementadas tipo entero, cada una con su respectivo pin, adicional a esto se inicializan algunas variables, ya que al hacer esto, facilita más la implementación del PID.

Se definen cuatro sensores, (sensor_1, sensor_2,sensor_3,sensor_4) cada uno de estos van a los pines del arduino analógico que se representan con A0, A1,A2,A3, respectivamente.

Adicionalmente se conectan unos pines (pwm) que representados en los puertos del arduino 9, 10, que van conectados a los pines del puente H, controlando características de los motores como la velocidad.

Se implementa un PID que está conformado por un mecanismo de control por realimentación ampliamente usado en sistemas de control industrial. Este calcula el error entre un valor medido y un valor deseado. Este control se basa en tres pautas, **proporcional** depende del error actual. El **Integral** depende de los errores pasados y el **Derivativo** es una predicción de los errores futuros, adicionalmente el derivativo corrige el sobreimpulso del sistema permitiendo de esta forma que el sistema llegue a su referencia. La suma de estas tres acciones es usada para ajustar al proceso por medio de un elemento de control como la posición de una válvula de control o la potencia suministrada a un calentador, o en este caso el robot seguidor de línea.

```
// Constantes del PID
float kp=0.20; //2,
float kd=12;
float ki=0.001;


void setup() {

  delay(2000);   // Hacemos una espera de 2 segundos

  Serial.begin(9600);  // iniciamos comunicacion serial con 9600 bouds


  // Configuramos los pines como entrada o salida
  pinMode(boton,INPUT_PULLUP);
  pinMode(motor_derechaadelante,OUTPUT);
  pinMode(motor_izquierdaadelante,OUTPUT);
  pinMode(motor_derechaatras,OUTPUT);
  pinMode(motor_izquierdaatras,OUTPUT);
  pinMode(ledOn,OUTPUT);
  pinMode(ledPlay,OUTPUT);
  pinMode(pwm_izquierdo,OUTPUT);
  pinMode(pwm_derecho,OUTPUT);

  int b = digitalRead(boton); // Leemos el valor en el pin boton (8)

// Ciclo de espera del pulso del boton de inicio
```

Figura 5. Segunda parte del código establecido para el robot seguidor de línea.

Se definen algunos pines con configuración de entrada(INPUT) o de salida(OUTPUT), dependiendo de la variable que se vaya a medir para hacer uso de este valor y de esta forma implementarlo en el resto de líneas de código. Como se observa en la Figura 5.

.

```
digitalWrite(ledPlay,HIGH); // Led que indica inicio de lectura de sensores

 // Lectura de los sensores en las entradas analogas
int s_1 = analogRead(sensor_1);
int s_2 = analogRead(sensor_2);
int s_3 = analogRead(sensor_3);
int s_4 = analogRead(sensor_4);



int s_11 = 0; // Variables usadas para la conversion
int s_12 = 0;
int s_13 = 0;
int s_14 = 0;
//Serial.println(s_2);

// Calibracion de sensores para convertirlos a binario
if(s_1>150){
  s_11=1;
}
if(s_2>250){
  s_12=1;
}
if(s_3>150){
  s_13=1;
}
if(s_4>300){
  s_14=1;
```

Figura 6. Cuarta parte del código seguidor de línea.

Una vez estén definidas las entradas análogicas, se hace uso de unas variables que permiten la conversión, realizando una calibración para convertirlos en binario, haciendo uso de la función de decisión IF, las variables relacionadas con los números de calibración de los sensores. Ver Figura6.

```
    posicion = (1*s_11)+(2*s_12)+(4*s_13)+(8*s_14);  // Convertimos los numeros binarios obtenidos de los sensores en decimales

if(posicion <= 14){
  posicion_ultima = posicion;   // Almacenamos las posiciones leidas
}

Serial.println(posicion_ultima);  // Mostramos en pantalla el valor de la posicion leida por los sensores

   // Calculamos los valores proporcional, integral y derivativo
   proporcional = (posicion - 9);
   integral = (integral + proporcional_anterior);
   derivativo = (proporcional - proporcional_anterior);

   // Valor obtenidos del PID
   pwm_valor = (kp*proporcional)+(ki*integral)+(kd*derivativo);

   //Serial.println(valor_pwm);  // Mostramos el valor del PID obtenido

   // Limitamos el valor de la integral
   if(integral>200){
    integral=200;
   }
   if(integral<-200){
```

Figura 7. Quinta parte del código seguidor de línea.

Gracias a los números convertidos a binario, se puede saber la posición en la que se encuentra el seguidor, una vez se tiene esta, se va almacenando sucesivamente las posiciones que se van leyendo,
Serial.println(posicion_ultima) esta línea muestra el valor en pantalla de la posición leída por los sensores.
Una vez calculados los valores proporcional, integral y derivativo, por medio de fórmulas establecidas. Los valores obtenidos del PID harán parte de la modulación por ancho de pulso.

```
   // Almacenamos el ultimo valor de proporcional
   proporcional_anterior = proporcional;


// Condiciones de la lectura de los sensores
if(posicion == 9){ // Si esta en linea, avanza
    digitalWrite(motor_derechaadelante,HIGH);
    digitalWrite(motor_derechaatras,LOW);
    analogWrite(pwm_derecho,200);
    digitalWrite(motor_izquierdaadelante,HIGH);
    digitalWrite(motor_izquierdaatras,LOW);
    analogWrite(pwm_izquierdo,200);
   }
if(posicion == 8){
    digitalWrite(motor_derechaadelante,HIGH);
    digitalWrite(motor_derechaatras,LOW);
    analogWrite(pwm_derecho,250);
    digitalWrite(motor_izquierdaadelante,LOW);
    digitalWrite(motor_izquierdaatras,HIGH);
    analogWrite(pwm_izquierdo,120+pwm_valor);
   }


if((posicion > 11 && posicion < 13)||(posicion > 13 && posicion < 15) || posicion < 8){  // Si los sensores estan en otra posicion

   if(pwm_valor < 0){  // Gira para la derecha
```

Figura 8. Sexta parte del código del seguidor de línea.
Se establecen condiciones para la lectura de los sensores, ya que lo que se lee de estos les dará la dirección a motores. Las condiciones que se usaron fueron IF, y cada una de las salidas fue puesto en pines ya sea analogWrite, todo esto basándose en si los sensores se encuentran en las posiciones 9 y 8.

```
if(pwm_valor < 0){  // Gira para la derecha
 digitalWrite(motor_derechaadelante,LOW);
 digitalWrite(motor_derechaatras,HIGH);
 analogWrite(pwm_derecho,120+pwm_valor);
 digitalWrite(motor_izquierdaadelante,HIGH);
 digitalWrite(motor_izquierdaatras,LOW);
 analogWrite(pwm_izquierdo,250);
}
if(pwm_valor > 0){ // Gira para la izquierda
 digitalWrite(motor_derechaadelante,HIGH);
 digitalWrite(motor_derechaatras,LOW);
 analogWrite(pwm_derecho,250);
 digitalWrite(motor_izquierdaadelante,LOW);
 digitalWrite(motor_izquierdaatras,HIGH);
 analogWrite(pwm_izquierdo,120-pwm_valor);
}

 if(posicion == 0){  // si todos los sensores estan sobre la linea negra, el carro frena.
 digitalWrite(motor_derechaadelante,LOW);
 digitalWrite(motor_derechaatras,LOW);
 analogWrite(pwm_derecho,0);
 digitalWrite(motor_izquierdaadelante,LOW);
 digitalWrite(motor_izquierdaatras,LOW);
 analogWrite(pwm_izquierdo,0);
 }

if(posicion == 15){  // Si los sensores se salieron de la linea negra, se lee el ultimo valor de la posicion leida por estos
```

Figura 8. Séptima parte del código del seguidor de línea.
Basándose en la variable pwm_valor, se usa la estructura de decisión IF, para establecer el direccionamiento del motor si este gira hacia la derecha o hacia la izquierda.
Debido a que los sensores que se utilizaron no tienen el voltaje convencional es decir (5 en negro, 0 en blanco), si todos los sensores se encuentran sobre la línea negra el carro frena, es decir, para.

```
if(posicion == 15){  // Si los sensores se salieron de la linea negra, se lee el ultimo valor de la posicion leida por estos

 if(posicion_ultima < 9){  // Gira para la derecha
   digitalWrite(motor_derechaadelante,LOW);
   digitalWrite(motor_derechaatras,HIGH);
   analogWrite(pwm_derecho,120-pwm_valor);
   digitalWrite(motor_izquierdaadelante,HIGH);
   digitalWrite(motor_izquierdaatras,LOW);
   analogWrite(pwm_izquierdo,250);
 }
 if(posicion_ultima > 9){ // Gira para le izquierda
   digitalWrite(motor_derechaadelante,HIGH);
   digitalWrite(motor_derechaatras,LOW);
   analogWrite(pwm_derecho,250);
   digitalWrite(motor_izquierdaadelante,LOW);
   digitalWrite(motor_izquierdaatras,HIGH);
   analogWrite(pwm_izquierdo,120-pwm_valor);
 }
}
```

Figura 9. Octava parte del código del seguidor de línea.
En esta parte se configura la opción en la que dado el caso los sensores se salieran de la línea negra, este leerá el ultimo valor de la posición leida por estos y por último se configuro cuando este girara hacia la izquierda

IV. CONCLUSIONES

- La herramienta de programación Arduino permite simplificar la funcionalidad del robot seguidor de línea.
- Estimula la creatividad y desarrollo de los conocimientos adquiridos hasta el momento, tanto en la parte programable como en la parte mecánica y técnica de este.
- Los sensores CNY70 depende de los valores correctos de las resistencias y de una buena ubicación de los sensores es fundamental ya que estos son los encargados de regular que

el carro siga la trayectoria de la línea negra.

- El diseño implementado para el montaje del seguidor de línea fue el correcto, porque permitió la ubicación de los sensores, las llantas fueron aptas y tuvieron buen desempeño, manteniendo en equilibrio su eje y los componentes que estas sostenían.
- Los emisores infrarrojos se deben calibrar a una distancia prudente del suelo ya que deben tener el espacio necesario para que cuando el emita un rayo de luz tenga la oportunidad de reflejarse y regresar para ser captada por el sensor.

V. REFERENCIAS BBLIOGRAFICAS

[1] http://es.slideshare.net/Michael_CrL/seguidor-de-linea-13770660

[2] https://prezi.com/zs1t6ijtmr3g/robot-seguidor-de-linea/

[3] http://aprender.tdrobotica.co/seguidor-de-lineaprofesional/

[4] http://www.jmnlab.com/cny70/cny_70.html.

[5] http://www.x-robotics.com/robots_simples.html.

[6] http://expoelectronica.upbbga.edu.co/pdf_2010_XVI/jhair_rodriguez_c.pdf.

Nathaly Sanchez Hincapie. nació en Toro, Valle Colombia, en enero 6 de 1998.Se graduó de bachiller en el Colegio Corazonista. En la actualidad se encuentra estudiando ingeniería electrónica en la Universidad Del Quindío. Ponente en la VIII Jornada Técnico-científica de la Facultad de Ingeniería.

Beymar Ruidiaz Martinez. Nació en Puente nacional Santander, Colombia, el 22 de diciembre de 1986. Actualmente es estudiante de decimo semestre de Ingeniería Electrónica de la Universidad del Quindío, área de interés automatización y control, perteneció al semillero de investigación de automatización y control.