ROBOT INTELIGENTE QUE RESUELVE LABERINTOS BASADO EN UN SEGUIDOR DE LINEA

Román H. Jhoan Cód.1094949367, Velásquez G. Juan Cód. 1098310202, Morcillo O. Esneyder Cod. 109495146

***Resumen*—** **En el presente informe se dará a conocer los aspectos más relevantes del diseño de un robot seguidor de línea en búsqueda de una llegada sometido a un laberinto, haciendo uso de códigos programados en lenguaje c en las tarjetas Arduino, de igual forma hallar el objetivo primordial de funcionamiento por medio de los componentes electrónicos a implementar.**

*Índice de términos*— Tarjeta Arduino, Módulos, Ensamble.

# **Introducción**

Hoy en día en las empresas de mayor reconocimiento mundial, podemos observar la actividad y trabajo de robots; maquinas hechas por el hombre que se encargan de facilitar o incluso realizar completamente las tareas humanas de mayor riesgo, peligro y esfuerzo. Tareas que para el hombre resultan difíciles y tediosas de emprender, lo cual económicamente genera gastos, las empresas u organismos de cualquier índole lo que buscan es reducir gastos, pero no disminuir la calidad de sus trabajos. En contexto cada vez que surge un problema no tan grave, ya que cuentan con recursos modernos que nos permiten acceder a mecanismos fáciles de manejar construir y adquirir.

La robótica y la electrónica son áreas importantes en la historia del planeta ya que en todos los campos nos permite crear una cantidad ilimitada de aplicaciones prácticas, las podemos utilizar en nuestra vida cotidiana dándose por hecho. En el caso de los seguidores de línea nos

Permite aproximarnos a la programación y a la aplicación de la teoría estudiada el cual se presentan alternativas que puedan satisfacer las necesidades planteadas a nuestro proyecto.

**OBJETIVOS**

**Objetivo General**

Implementar un robot que busque el centro de un laberinto seguidor de línea, cuyas características cumplan requerimientos con respecto a modos de programación, iniciativa, ensamble y funcionamiento concerniente a esta tecnología muy común en la ingeniería.

**Objetivos Específicos**

Obtener un seguidor de línea que siga un camino marcado con líneas negras utilizando códigos de programación en tarjeta Arduino.

Adaptar los sensores necesarios para detectar las líneas negras con respecto al fondo blanco y activar el funcionamiento correspondiente.

**MARCO TEÓRICO**

Todos los rastreadores basan su funcionamiento en los sensores. Sin embargo, dependiendo de la complejidad del recorrido, el robot debe ser más o menos complejo (y, por ende, utilizar más o menos sensores).

Los rastreadores más simples utilizan 2 sensores, ubicados en la parte inferior de la estructura, uno junto al otro. Cuando uno de los dos sensores detecta el color blanco, significa que el robot está saliendo de la línea negra por ese lado. En ese momento, el robot gira hacia el lado contrario hasta que vuelve a estar sobre la línea. Esto en el caso de los seguidores de línea negra, ya que también hay seguidores de línea blanca.

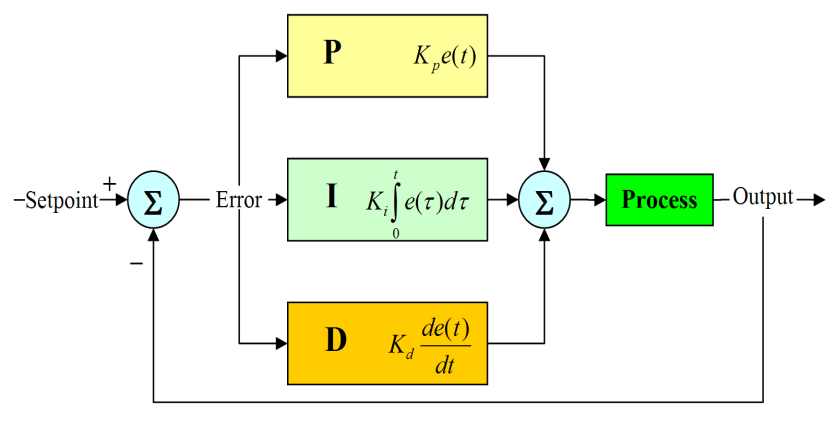
Estos robots tienen la capacidad de seguir una línea marcada en el suelo (normalmente una línea negra sobre un fondo blanco).

III.1 Estructura b**á**sica

Estos robots pueden variar desde los más básicos (van tras una línea única) hasta los robots que recorren laberintos. Todos ellos, sin embargo, poseen (por lo general) ciertas partes básicas comunes entre todos sensores bonitos y baratos, como debe ser. Un rastreador detecta la línea a seguir por medio de sensores. Hay muchos tipos de sensores que se pueden usar para este fin; sin embargo, por razones de costos y practicidad los más comunes son los sensores infrarrojos (IR), que normalmente constan de un LED infrarrojo y un fototransistor.

**PID**

El PID (control proporcional, integral y derivativo) es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener (set point, target position o punto de consigna), para aplicar una acción correctora que ajuste el proceso.



*Fig. 1 Diagrama que representa el PID*

Para nuestro seguidor de línea el PID (que es una rutina basada matemáticamente), procesa los datos de los sensores infrarrojos y los usa para controlar la velocidad y posición del robot.

**Pasos para calibrar el PID**

Para poder calibrar el PID se siguen los siguientes pasos:

Paso 1: Acción Proporcional

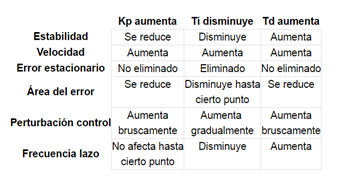
* Tiempo integral (TI), a su máximo valor.
* Tiempo derivativo (TD), a su mínimo valor.
* Empezando con ganancia baja se va aumentando hasta obtener las características de respuesta deseadas.

Paso 2: Acción integral

* Reducir el TI hasta anular el error en estado estacionario, aunque la oscilación sea excesiva.
* Disminuir ligeramente la ganancia.
* Repetir hasta obtener las características de respuesta deseadas.

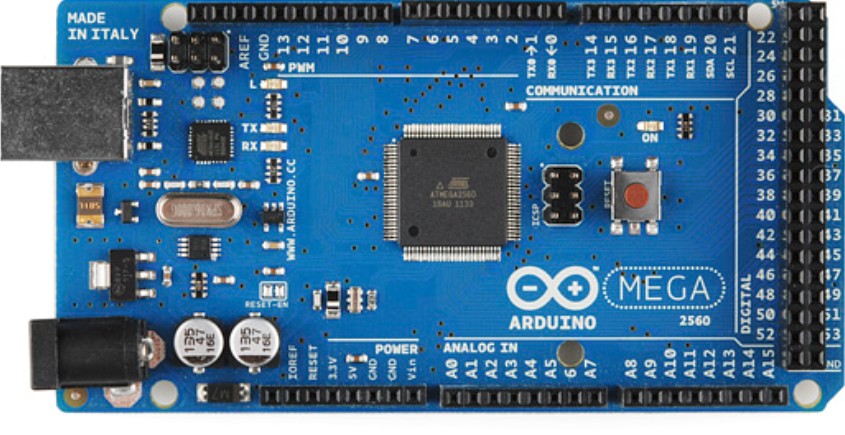
Paso 3: Acción Derivativa

* Mantener ganancia y tiempo integral obtenidos anteriormente.
* Aumentar el TD hasta obtener características similares, pero con la respuesta más rápida.
* Aumentar ligeramente la ganancia si fuera necesario.

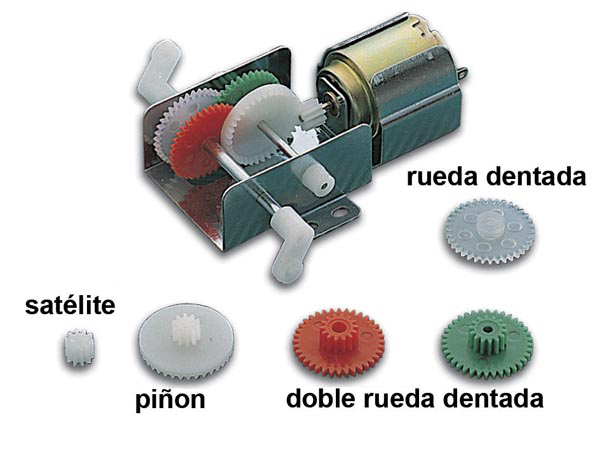
*Fig. 2 Esquema que representa el PID*

# **materiales**

- Arduino Mega (Atmega 2560)

*Fig. 3 Arduino MEGA*

**Tarjeta de control Arduino (MEGA):** La toma de decisiones y el control de los motores están generalmente a cargo de un micro controlador. La tarjeta de control contiene dicho elemento, junto a otros componentes electrónicos básicos que requiere el micro controlador para funcionar, para ello se hace una programación a un código en c para lograr que el micro funcione a lo requerido.



*Fig. 4 Motor acoplado con piñones*

**Motores:** El robot se mueve utilizando motores. Dependiendo del tamaño, el peso, la precisión del motor, entre otros factores, éstos pueden ser de varias clases: motores de corriente continua, motor paso a paso o servomotores.

  
*Fig. 5 Ruedas acopladas a los motores*

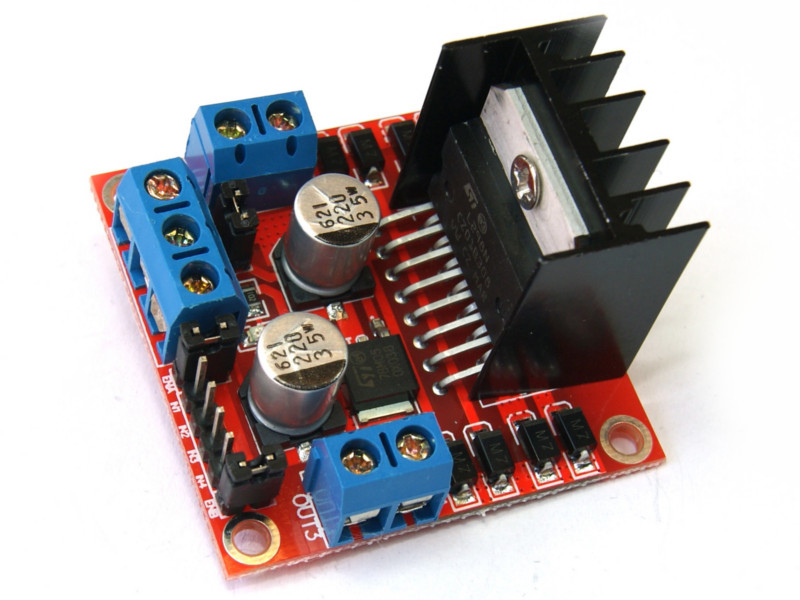
**Ruedas:** Las ruedas del robot son movidas por los motores. Normalmente se usan ruedas de materiales

anti-deslizantes para evitar fallas de tracción. Su tamaño es otro factor a tener en cuenta a la hora de armar el robot, sin embargo, las ruedas que fueron utilizadas tenían más fricción que como debía usarse.



*Fig. 6 Batería usada como fuente de energía*

**Fuente de energía:** El robot obtiene la energía que necesita para su funcionamiento de baterías o de una fuente de corriente alterna, siendo esta última menos utilizada debido a que le resta independencia al robot. En este caso se hizo uso de las baterías mostradas en la presente figura correspondiente a las baterías de un celular Nokia 1100, ya que generaban el voltaje correspondiente y necesitado para dar buen funcionamiento al módulo puente H previamente a los motores y a la tarjeta Arduino.



*Fig. 7 Modulo Puente H L298N*

**Modulo puente H** **L298N** es una tarjeta para el control de motores de corriente directa, motores a pasos, solenoides y en general cualquier otra carga inductiva. La tarjeta está construida en torno al circuito integrado **L298N**, el cual dispone en su interior de 2 puentes H independientes con capacidad de conducir 2 amperios constantes o 4 amperios en picos no repetitivos. La tarjeta expone las conexiones hacia el motor a través de bloques de terminales (clemas), mientras que las entradas de control y habilitación del puente H se exponen a través de headers macho estándar para facilitar todas las conexiones.

Esta tarjeta es ideal para controlar motores en pequeños robots como seguidores de líneas, zumos, robots de laberinto, etc. El L298N también es una excelente opción para manejar motores a pasos bipolares.



*Fig. 8 Modulo Qtr-8RC*

**Sensor Pololou QTR-8RC:** El sensor de reflectancia de QTR-8RC está pensado como un sensor de línea, pero puede ser utilizado como una proximidad de propósito general o un sensor de reflectancia. El módulo es un vehículo conveniente para ocho pares IR emisor y receptor (fototransistor) uniformemente espaciados a intervalos de 0,375 "(9,525 mm). Para utilizar un sensor, debe cargar primero el nodo de salida mediante la aplicación de un voltaje a su OUT pasador. Usted a continuación, puede leer la reflectancia mediante la retirada de la tensión suministrada externamente y de temporización el tiempo que tarda el voltaje de salida a la descomposición debido al fototransistor integrado. Shorter tiempo de decaimiento es una indicación de una mayor reflexión. Este método de medición tiene varias ventajas, especialmente cuando se combina con la capacidad del módulo de QTR-8RC para desactivar la alimentación del LED:

* No se requiere ninguna conversión analógica-digital (ADC)
* Mejora de la sensibilidad sobre la salida analógica del divisor de voltaje
* la lectura en paralelo de múltiples sensores es posible con la mayoría de los microcontroladores
* lectura paralela permite un uso optimizado de la energía LED opción de habilitar

Las salidas son todos independientes, pero los LEDs están dispuestos en pares reducir a la mitad el consumo de corriente. Los LEDs son controlados por un MOSFET con una puerta normalmente sacaron alta, permitiendo que los LEDs se apaguen mediante el establecimiento de la puerta MOSFET a una tensión baja. Volviendo los LEDs off podría ser ventajoso para limitar el consumo de energía cuando los sensores no están en uso o para variar el brillo efectivo de los LEDs a través del control PWM.

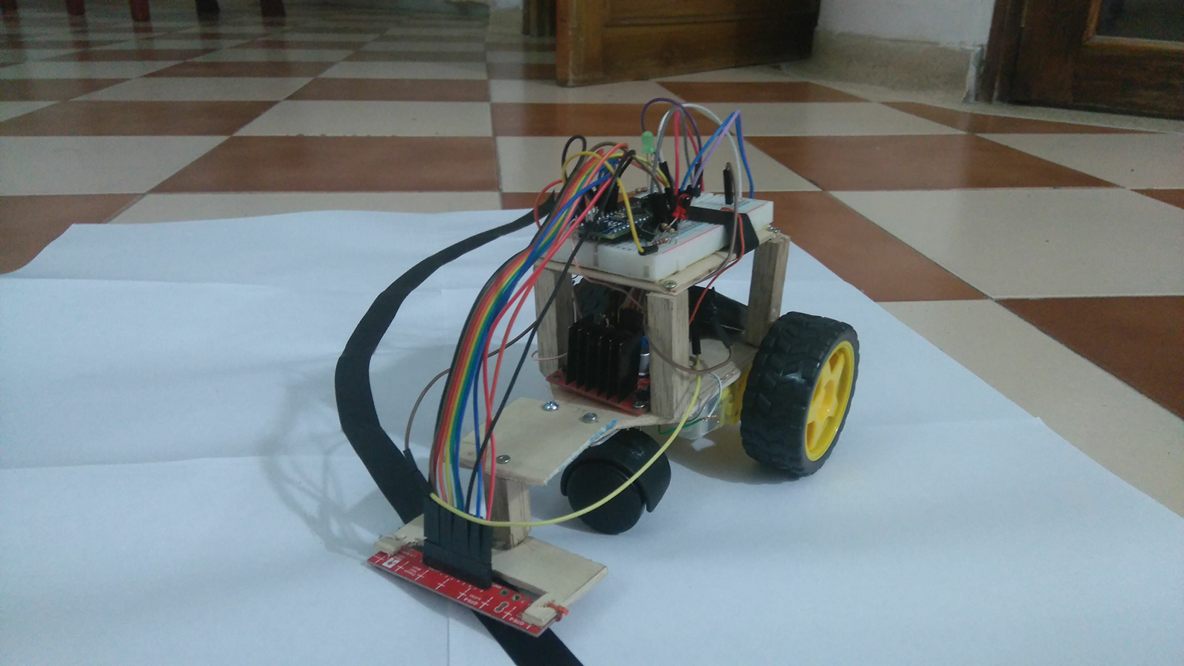
Este sensor fue diseñado para ser utilizado con la placa paralela a la superficie que está siendo detectada.

Las resistencias limitadoras de corriente de LED para 5 V operación están dispuestos en dos etapas; esto permite un simple derivación de una etapa a habilitar la operación actual en el 3,3 V. El LED es de aproximadamente 20-25 mA, por lo que el consumo total de tablero de poco menos de 100 mA. El diagrama esquemático del módulo se muestra a continuación:

**PROCEDIMIENTO**

Para la implementación del proyecto se tomó parte del anterior trabajo como lo es la parte física del montaje y la utilización de los módulos reconocedores de la línea negra, se incluirá una parte que detalla la solución que el carro le proporciona al laberinto en donde se visualizara el mapeo en un programa externo al de arduino, este tiene como objetivo interpretar lo que llega del puerto serial de arduino y mostrarlo dinámicamente en una interfaz personalizada, el programa que muestra esta ilustración se denomina Processing y basa su funcionamiento en Java un lenguaje de programación al igual que C. El proceso que se llevó a cabo, en primera medida es el rediseño total y parcial del mini-auto que es el encargado a través de su chasis de llevar los componentes de funcionamiento y la alimentación, el vehículo como tal se hizo más compacto y pequeño tomando medidas muy precisas, esto se realizó con la finalidad de que no interfiriera con el camino del laberinto denotando su correcto giro sin chocar con las paredes y pudiendo así lograr una reducción de posibles errores a la hora de realizar el trayecto.

Debido a las fallas con los motores del proyecto anterior se optó por conseguir motores de mejor calidad que albergan más movimiento igualitario y proporcional, también se reemplazó los sensores hechos a mano desde cero por un módulo completo que incluía 8 sensores y una librería que facilitaba su uso (Pololu Qtr), el carro como tal quedo de nuevo hecho y se evidencia en la fig. 3 su montaje.



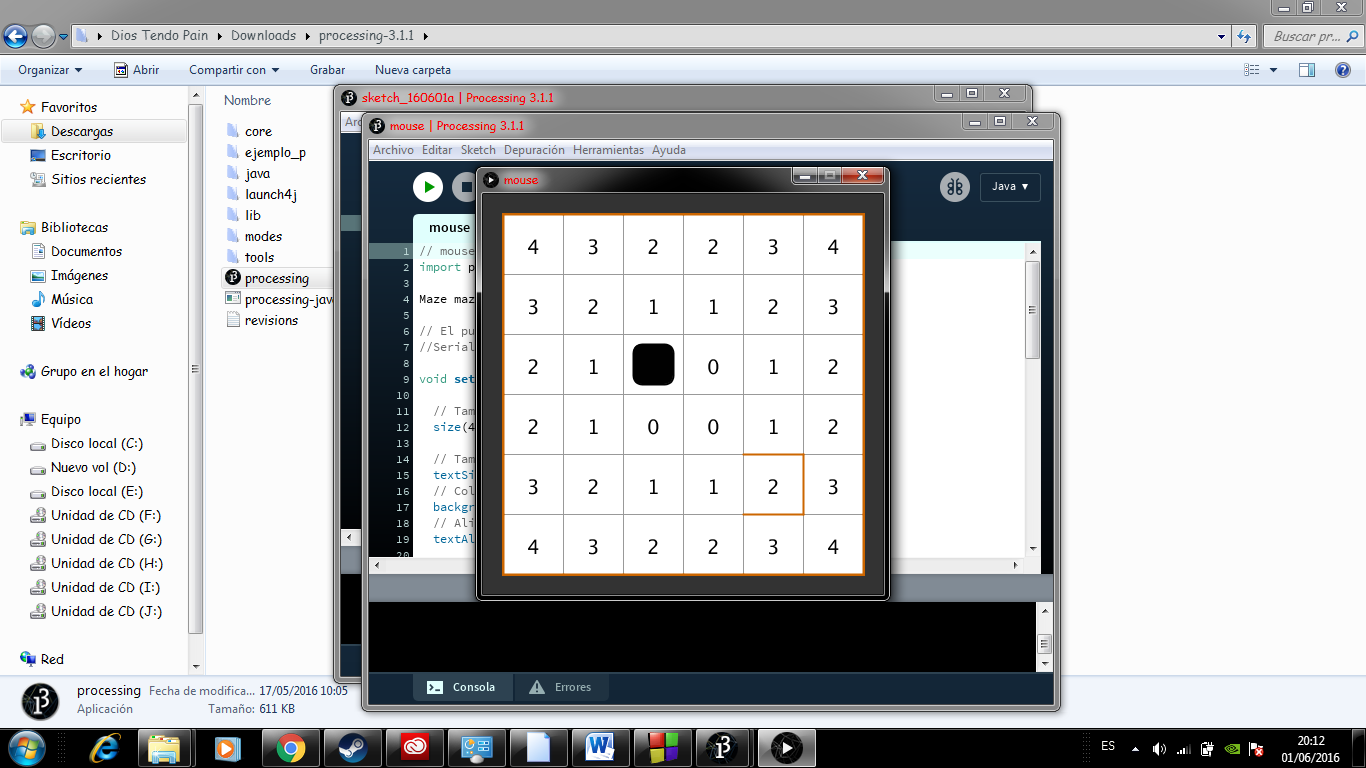
*Fig. 9 Carro finalizado, listo para pruebas*

Como tal el código fue reemplazado en su totalidad debido a que la lógica nuestra difiere en gran medida en la que proporciona el sensor, ya que este basa su funcionamiento en la librería propia del módulo (la librería se anexara en el repositorio), las pruebas de giro aún están siendo realizadas y cuando se tenga algo contundente será anexado.

La lógica que seguirá el carro es simple, primero se sabrá que la matriz será de 6x6, el vehículo iniciara en la posición (0,0) siendo esta la primer parte coordenada de la matriz, el carro deberá buscar el camino descendiente más cercano para esto se le asigna valores que descienden de 1 en 1, cuando el valor es ascendente el carro no debería seguir ese camino (esto ocurre porque hay un muro que le impide seguir), una vez haya dado con el centro del laberinto tendrá que recorrer el laberinto sin choques y sin meterse en caminos inciertos. Al final y durante el proceso debe haber comunicación serial entre el computador y el arduino para constatar el correcto mapeo en el Processing.

El vehículo se supone que no debe saberse el laberinto y por lo tanto a medida que lo soluciona debe graficar en la interfaz de la figura 10

De la figura 10 también se visualiza una pared que fue puesta a modo de prueba para verificar la gráfica resultante de las paredes



*Fig. 10**cuadricula a llenar*

Ahora, se analizaron las funciones establecidas en el código proporcionado por el profesor Luis Miguel Capacho

**Setwalls:** Esta función correlaciona las diferentes ubicaciones entre celdas y utilizando la predefinición NORT, EAST, SOUTH, WEST se direcciona el lado que se quiere dibujar.

**Setvalue:** esta función relaciona el valor donde está ubicado el seguidor de línea, modificándolo.

**Getvalue:** getvalue es una función que extrae el valor donde se encuentra el seguidor de línea en el laberinto

**Getwalls:** esta función se caracteriza por obtener los valores donde se encuentran las paredes y así gracias a ello poder dar solucionar al laberinto. Esta función retorna un valor numérico potencia de 2 (8 si es WEST, 1 NORTH, 4 SOUTH, 2 EAST)

**Display:** es la función encargada de dibujar la cuadricula a utilizar dándole parámetros como el tamaño y el color, además es la encargada de dibujar el mouse que simula la posición del carro en la matriz.

# **código, ensamble y funcionamiento**

Para el desarrollo de este proyecto se analizaron muchos factores como los motores a usar, como se iba a quemar la baquela que contenía los sensores seguidores de línea, los trimer para dar las constantes del PID. Teniendo en cuenta los parámetros anteriores se diseñó un código, tal que este pueda realizar los procedimientos necesarios para el correcto funcionamiento del proyecto.

A continuación, se presenta el código usado y unas breves especificaciones de lo que realizan algunas líneas específicas:

A continuación, se presenta el código con el que se le hace el control al robot, en este se muestra un programa extenso en el que el vehículo es manejado con condiciones if con las cuales se analiza la posición del carro con respecto a la línea negra que entrega el sensor QTR. El carro al censar con un QTR de 8 pines, está diseñado de tal forma que los cuatro sensores del medio, únicamente se centren el seguir la línea y los sensores restantes están distribuidos (dos en la derecha y dos en la izquierda) de tal forma que su objetivo sea censar los puntos que representan el cambio de cuadro en el laberinto. Ya teniendo idea sobre cómo va a estar distribuido el vehículo se crean los if con los cuales lo que se hace es que, para cada posición, el vehículo tiene diferente comportamientos, por ejemplo el valor 0b11100111 representa que el carro va en todo el centro de la línea con lo cual la orden que se le da es que los 2 motores estén sincronizados girando hacia adelante para que el vehículo siga la línea sin salirse de esta.

*#include <QTRSensors.h>*

*#define NUM\_SENSORS 8 //numero de sensores usados*

*#define TIMEOUT 2000 // tiempo de espera para dar resultado en uS*

*#define EMITTER\_PIN 6 //pin led on*

*//pines arduino a utilizar*

*#define led1 13*

*#define led2 4*

*#define sensores 6*

*#define boton 2 //pin para boton*

*#define enable1 9*

*#define motor1\_at*

*#define motor1\_ad*

*#define enable2*

*#define motor2\_at*

*#define motor2\_ad*

*int valorCel = 6; //valor inicial para la celda*

*int celda[6][6]; // matriz de posiciones de las celdas*

*// variables que definen la posicion de una celda*

*int i=0, i1=0, p=0, p1=0;*

*// variables que definen la orientacion del robot*

*boolean arriba = false;*

*boolean abajo = false;*

*boolean derecha = true;*

*boolean izquierda = false;*

*QTRSensorsRC qtrrc((unsigned char[]) {19, 18, 17, 16,15,14,11,12}*

*,NUM\_SENSORS, TIMEOUT, EMITTER\_PIN);*

*//variables para almacenar valores de sensores y posicion*

*unsigned int sensorValues[NUM\_SENSORS];*

*unsigned int position=0;*

*/// variables para el pid*

*//int derivativo=0, proporcional=0, integral=0; //parametros*

*//int salida\_pwm=0, proporcional\_pasado=0;*

*//AQUI CAMBIEREMOS LOS PARAMETROS DE NUESTRO ROBOT*

*int velocidad=120; //variable para la velocidad, el maximo es 255*

*//float Kp=0.18, Kd=4, Ki=0.001; //constantes*

*//variable para escoger el tipo de linea*

*int linea=0; // 0 para lineas negra, 1 para lineas blancas*

*void setup()*

*{*

*delay(800);*

*pinMode(motor1\_ad, OUTPUT);//pin de direccion hacia adelante motor izquierdo*

*pinMode(motor1\_at, OUTPUT);//pin de direccion hacia atras motor izquierdo*

*pinMode(motor2\_ad, OUTPUT);//pin de direccion hacia adelante motor derecho*

*pinMode(motor2\_at, OUTPUT);//pin de direccion hacia atras motor derecho*

*pinMode(led1, OUTPUT); //led1*

*pinMode(led2, OUTPUT); //led2*

*// pinMode(boton\_1, INPUT); //boton 1 como pull up*

*for (int i = 0; i < 40; i++) //calibracion durante 2.5 segundos,*

*{ //para calibrar es necesario colocar los sensores sobre la superficie negra y luego*

*digitalWrite(led1, HIGH); //la blanca*

*delay(20);*

*qtrrc.calibrate(); //funcion para calibrar sensores*

*digitalWrite(led1, LOW);*

*delay(20);*

*}*

*digitalWrite(led1, LOW); //apagar sensores para indicar fin*

*//de calibracion*

*delay(400);*

*digitalWrite(led2,HIGH); //encender led 2 para indicar la*

*// espera de pulsacion de boton*

*while(true)*

*{*

*int b=digitalRead(boton); //leemos y guardamos el valor*

*// del boton en variable x*

*delay(100);*

*if(b==0) //si se presiona boton*

*{*

*digitalWrite(led2,LOW); //indicamos que se presiono boton*

*digitalWrite(led1,HIGH); //encendiendo led 1*

*delay(100);*

*break; //saltamos hacia el bucle principal*

*}*

*}*

*position = qtrrc.readLine(sensorValues, QTR\_EMITTERS\_ON, linea); //0 para linea*

*//negra, 1 para linea blanca*

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

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

*while(b==1){*

*b = digitalRead(boton);*

*digitalWrite(ledOn,255);*

*delay(300);*

*}*

*digitalWrite(ledOn,0);*

*}*

*void loop()*

*{*

*delay(1000);*

*int b = digitalRead(boton);*

*for(p=0;p<6;p++){ // asignamos el valor de 10 a cada celda de la matriz*

*for(p1=0;p1<6;p1++){*

*celda[p][p1]=10;*

*}*

*}*

*celda[i][i1] = valorCel; // asignamos a la posicion inicial [0,0] el valor 6*

*while(b==1){ // bucle de inicio de recorrido del laberinto*

*b = digitalRead(boton);*

*if((valorCel == 0){*

*digitalWrite(motor2\_ad,LOW);*

*digitalWrite(motor2\_at,LOW);*

*analogWrite(enable2,0);*

*digitalWrite(motor1\_at,LOW);*

*digitalWrite(motor1\_ad,LOW);*

*analogWrite(enable,0);*

*}*

*else*

*{*

*for(p=0;p<6;p++){ // asignamos el valor de 10 a cada celda de la matriz*

*for(p1=0;p1<6;p1++){*

*celda[p][p1]=celda[p][p1]+1;*

*}*

*}*

*valorCel=celda[i][i1];*

*// datos que son enviados a processing*

*Serial.print(i,',',i1,',');*

*Serial.println(celda[i][i1]);*

*if(posicion == ob11100111){*

*if((valorCel>0) && (valorCel<7)){*

*if(cuadricula == 0){*

*if(arriba){*

*i1-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(abajo){*

*i1+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(derecha){*

*i+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(izquierda){*

*i-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*}*

*digitalWrite(motor2\_ad,HIGH);*

*digitalWrite(motor2\_at,LOW);*

*analogWrite(enable2,100);*

*digitalWrite(motor1\_ad,HIGH);*

*digitalWrite(motor1\_at,LOW);*

*analogWrite(enable1,100);*

*}*

*}*

*if(posicion == ob11100011){*

*if((valorCel>0) && (valorCel<7)){*

*if(cuadricula == 0){*

*if(derecha){*

*arriba=true;*

*derecha=!derecha;*

*i+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(izquierda){*

*abajo=true;*

*izquierda=!izquierda;*

*i-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(arriba){*

*izquierda=true;*

*arriba=!arriba;*

*i1-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(sur){*

*derecha=true;*

*abajo=!abajo;*

*i1+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*}*

*digitalWrite(motor1\_ad,LOW);*

*digitalWrite(motor1\_at,LOW);*

*analogWrite(enable1,0);*

*digitalWrite(motor2\_ad,HIGH);*

*digitalWrite(motor2\_at,LOW);*

*analogWrite(enable2,150);*

*delay(1000);*

*digitalWrite(motor\_izquierda\_ad,LOW);*

*digitalWrite(motor\_izquierda\_at,HIGH);*

*analogWrite(pwm\_iz,150);*

*while(posicion!=9){ //el robot gira a la izquierda hasta encontrar la linea*

*posicion = posi();*

*b = digitalRead(boton);*

*}*

*}*

*}*

*if(posicion == ob11000111){*

*if(valorCel <= 7 & valorCel > 0){*

*if(cuadricula == 0){*

*if(derecha){*

*abajo=true;*

*derecha=!derecha;*

*i+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(izquierda){*

*arriba=true;*

*izquierda=!izquierda;*

*i-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(arriba){*

*derecha=true;*

*arriba=!arriba;*

*i1-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(abajo){*

*izquierda=true;*

*abajo=!abajo;*

*i1+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*}*

*digitalWrite(motor2\_ad,LOW);*

*digitalWrite(motor\_at,LOW);*

*analogWrite(enable2,0);*

*digitalWrite(motor1\_at,LOW);*

*digitalWrite(motor1ad,HIGH);*

*analogWrite(enable,150);*

*delay(1000);*

*digitalWrite(motor2\_at,HIGH);*

*digitalWrite(motor2\_ad,LOW);*

*analogWrite(enble2,150);*

*while(posicion!=ob11100111){//gira a la derecha hasta encontrar la linea*

*posicion = posi();*

*}*

*}*

*}*

*if((posicion<0b11111111)&(posicion!=0b11100111)&(posicion!=0b11100011)&(posicion!=0b11000111)){ // Si los sensores estan en otra posicion*

*if((valorCel>0) && (valorCel<7)){*

*if(valor\_pwm < 0){ // Gira para la derecha*

*digitalWrite(motor2\_at,HIGH);*

*digitalWrite(motor2\_ad,LOW);*

*analogWrite(enable2,50+(-1)\*valor\_pwm);*

*digitalWrite(motor1\_ad,HIGH);*

*digitalWrite(motor1\_at,LOW);*

*analogWrite(enable1,150);*

*}*

*if(valor\_pwm > 0){ // Gira para la izquierda*

*digitalWrite(motor2\_ad,HIGH);*

*digitalWrite(motor2\_at,LOW);*

*analogWrite(enable2,150);*

*digitalWrite(motor\_izquierda\_at,HIGH);*

*digitalWrite(motor\_izquierda\_ad,LOW);*

*analogWrite(pwm\_iz,50+valor\_pwm);*

*}*

*}*

*}*

*if(posicion == 0b11000011){ // si todos los sensores estan sobre la linea negra, el carro gira a la derecha. & cuadricula == 2*

*if((valorCel>0) && (valorCel<7)){*

*if(cuadricula == 0){*

*if(derecha){*

*abajo=true;*

*derecha=!derecha;*

*i+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(izquierda){*

*arriba=true;*

*izquierda=!izquierda;*

*i-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(arriba){*

*derecha=true;*

*arriba=!arriba;*

*i1-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(abajo){*

*izquierda=true;*

*abajo=!abajo;*

*i1+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*}*

*digitalWrite(motor2\_ad,LOW);*

*digitalWrite(motor2\_at,LOW);*

*analogWrite(enable2,0);*

*digitalWrite(motor1\_at,LOW);*

*digitalWrite(motor1\_ad,HIGH);*

*analogWrite(enable1,150);*

*delay(1000);*

*digitalWrite(motor2\_at,HIGH);*

*digitalWrite(motor2\_ad,LOW);*

*analogWrite(enable2,150);*

*while(posicion!=0b11100111){*

*posicion=posi();*

*}*

*}*

*}*

*if(posicion == 0b11111111){ // si el camino termina hace un giro de 180 grados y actualiza el valor de la celda*

*if((valorCel>0) && (valorCel<7)){*

*if(cuadricula == 0) {*

*if(derecha){*

*izquierda=true;*

*derecha=!derecha;*

*}*

*if(izquierda){*

*este=true;*

*izquierda=!izquierda;*

*}*

*if(arriba){*

*abajo=true;*

*}*

*if(abajo){*

*arriba=true;*

*abajo=!abajo;*

*}*

*valorCel+=2;*

*celda[i][i1] = valorCel;*

*}*

*digitalWrite(motor2\_at,LOW);*

*digitalWrite(motor2\_ad,HIGH);*

*analogWrite(enable2,150);*

*digitalWrite(motor1\_ad,LOW);*

*digitalWrite(motor1\_at,HIGH);*

*analogWrite(enable1,150);*

*while(posicion!=0b11100111){*

*posicion = posi();*

*}*

*}*

*}*

*}*

*// secuencia de espera*

*delay(300);*

*digitalWrite(ledPlay,LOW);*

*delay(3000);*

*digitalWrite(motor2\_at,LOW);*

*digitalWrite(motor2\_ad,LOW);*

*analogWrite(enable2,0);*

*digitalWrite(motor1\_ad,LOW);*

*digitalWrite(motor1\_at,LOW);*

*analogWrite(enable1,0);*

*// reiniciamos las condiciones iniciales*

*i=0;*

*i1=0;*

*arriba = false;*

*abajo = false;*

*derecha = true;*

*izquierda = false;*

*valorCel=celda[i][i1];*

*while(b==1){*

*digitalWrite(ledPlay,HIGH);*

*delay(300);*

*digitalWrite(ledPlay,LOW);*

*delay(300);*

*}*

*delay(2000);*

*digitalWrite(ledPlay,HIGH);*

*while(true){ // en este bucle infinito, el robot tiene memorizado la trayectoria anterior y buscara el camino mas corto para llegar a la meta*

*posicion = posi(); // pedimos el valor de la posicion*

*cuadricula = cuad();*

*if(valorCel == 0b11000011){*

*digitalWrite(motor2\_ad,LOW);*

*digitalWrite(motor2\_at,LOW);*

*analogWrite(enable2,0);*

*digitalWrite(motor1\_at,LOW);*

*digitalWrite(motor1\_ad,LOW);*

*analogWrite(enable1,0);*

*}*

*// datos que son enviados a processing*

*Serial.print(i,',',i1,',');*

*Serial.println(celda[i][i1]);*

*if(posicion == 0b11100111){ // Si esta en linea, avanza*

*if(valorCel > 0){*

*if(cuadricula == 0){*

*if(arriba){*

*i1-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(abajo){*

*i1+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(derecha){*

*i+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(izquierda){*

*i-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*}*

*digitalWrite(motor2\_ad,HIGH);*

*digitalWrite(motor2\_at,LOW);*

*analogWrite(enble2,150);*

*digitalWrite(motor1\_ad,HIGH);*

*digitalWrite(motor1\_at,LOW);*

*analogWrite(enable1,150);*

*}*

*}*

*if(posicion == 0b11100011){ // Gira para la izquierda*

*if(valorCel > 0){*

*if(cuadricula == 0){*

*if(derecha){*

*arriba=true;*

*derecha=!derecha;*

*i+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(izquierda){*

*abajo=true;*

*izquierda=!izquierda;*

*i-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(arriba){*

*izquierda=true;*

*arriba=!arriba;*

*i1-=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*if(abajo){*

*derecha=true;*

*abajo=!abajo;*

*i1+=1;*

*valorCel-=1;*

*celda[i][i1]=valorCel;*

*}*

*}*

*digitalWrite(motor1\_ad,LOW);*

*digitalWrite(motor1\_at,LOW);*

*analogWrite(enable1,0);*

*digitalWrite(motor2\_ad,HIGH);*

*digitalWrite(motor2\_at,LOW);*

*analogWrite(enable2,150);*

*digitalWrite(motor1\_ad,LOW);*

*digitalWrite(motor1\_at,HIGH);*

*analogWrite(enable1,150);*

*}*

*}*

*}*

*void frenos(int tipo,int flanco\_comparacion)*

*{*

*if(tipo==0)*

*{*

*if (position<=500) //si se salio por la parte derecha de la linea*

*{*

*while(true)*

*{*

*qtrrc.read(sensorValues); //lectura en bruto de sensor*

*if ( sensorValues[0]>flanco\_comparacion || sensorValues[1]>flanco\_comparacion )*

*{*

*break;*

*}*

*}*

*}*

*if (position>=6500) //si se salio por la parte izquierda de la linea*

*{*

*while(true)*

*{*

*qtrrc.read(sensorValues);*

*if (sensorValues[7]>flanco\_comparacion || sensorValues[6]>flanco\_comparacion )*

*{*

*break;*

*}*

*}*

*}*

*}*

*//\*\*\*\*\*\*\**

*if(tipo==1) //para linea blanca con fondo negro*

*{*

*if (position<=500) //si se salio por la parte derecha de la linea*

*{*

*while(true)*

*{*

*qtrrc.read(sensorValues); //lectura en bruto de sensor*

*if ( sensorValues[0]<flanco\_comparacion || sensorValues[1]<flanco\_comparacion )*

*//asegurar que esta en linea*

*{*

*break;*

*}*

*}*

# **CONCLUSIONES**

* La comunicación de forma inalámbrica entre el Arduino y el computador genera beneficios, siendo estos una utilidad al momento de censar o utilizar cualquier tipo de aplicación que requiera información en tiempo real
* La resolución de un laberinto por parte de un carro robot no es compleja, pues este utiliza de forma constantes direcciones sabiendo de forma exacta qué camino seguir y optar por rechazar de forma permanente los caminos erróneos
* La lógica secuencial de un seguidor de línea sirvió para introducir el proyecto, donde los muros se simulaban de forma horizontal
* El espacio temporal manejado por el vehículo que recorre el laberinto estaba dado por matrices permitiendo así la obtención de datos a través de una sola variable
* Se pudo observar claramente que diversos tipos de censado son aptos para ser utilizados en la resolución de un laberinto

# **BIBLIOGRAFÍA**

[1]https://es.wikipedia.org/wiki/Robot\_seguidor\_de\_l%C3%ADnea

[2]http://www.buenastareas.com/materias/informe-seguidor-de-linea/0

[3]https://es.scribd.com/doc/78191609/seguidor

[4]http://seguidordelineaups.blogspot.com.co/