[[1]](#footnote-1)

Documentación Completa proyecto SEGUIDOR DE LINEA usando control PID:

*Laura Elena Aristizabal Millán,* Programa de Ingeniería Electrónica, Facultad de Ingeniería, Universidad del Quindío - Colombia.

1. ***INTRODUCCION:***

Los **robots seguidores de línea** son un tipo de robots relativamente sencillos que cumplen la misión de seguir una línea marcada en el suelo normalmente de color negro.

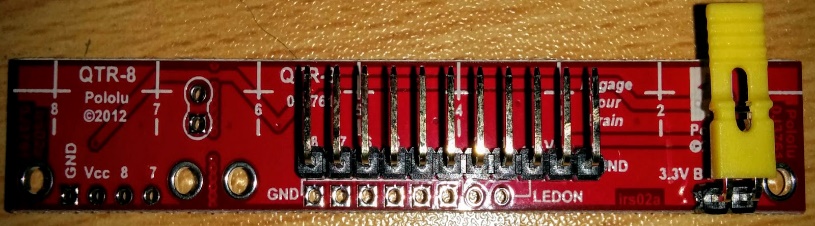
1. ***DESARROLLO:***

**Estructura Básica de un seguidor de línea (Fase de Diseño Electrónico y Mecánico)**

**Sensores:** Un rastreador detecta la línea a seguir por medio de sensores. Hay muchos tipos de sensores que se pueden usar para este fin, éste caso se usará un arreglo de sensores de referencia QTR-8: Este módulo sensor tiene 8 IR pares / fototransistor LED, por lo que es un buen detector para un robot SEGUIDOR DE LINEA.

Los pares de LEDS están dispuestos en serie para reducir a la mitad el consumo de corriente.

La línea a seguir, puede ser de color negro con fondo blanco o línea blanca con fondo negro y dependerá de la configuración electrónica con la cual se arme el circuito de dichos sensores.



***Fig.1****.* Regleta de sensores QTR-8.

**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, motores paso a paso o servomotores, para el caso de la implementación de este montaje se usaran motores de corriente continua con caja reductora tipo T con ruedas de 66 mm, voltaje de operación  desde 3V hasta 6 V., RPM entre 19 y 140 RPM.

****

***Fig.2.*** Motorreductor tipo T.

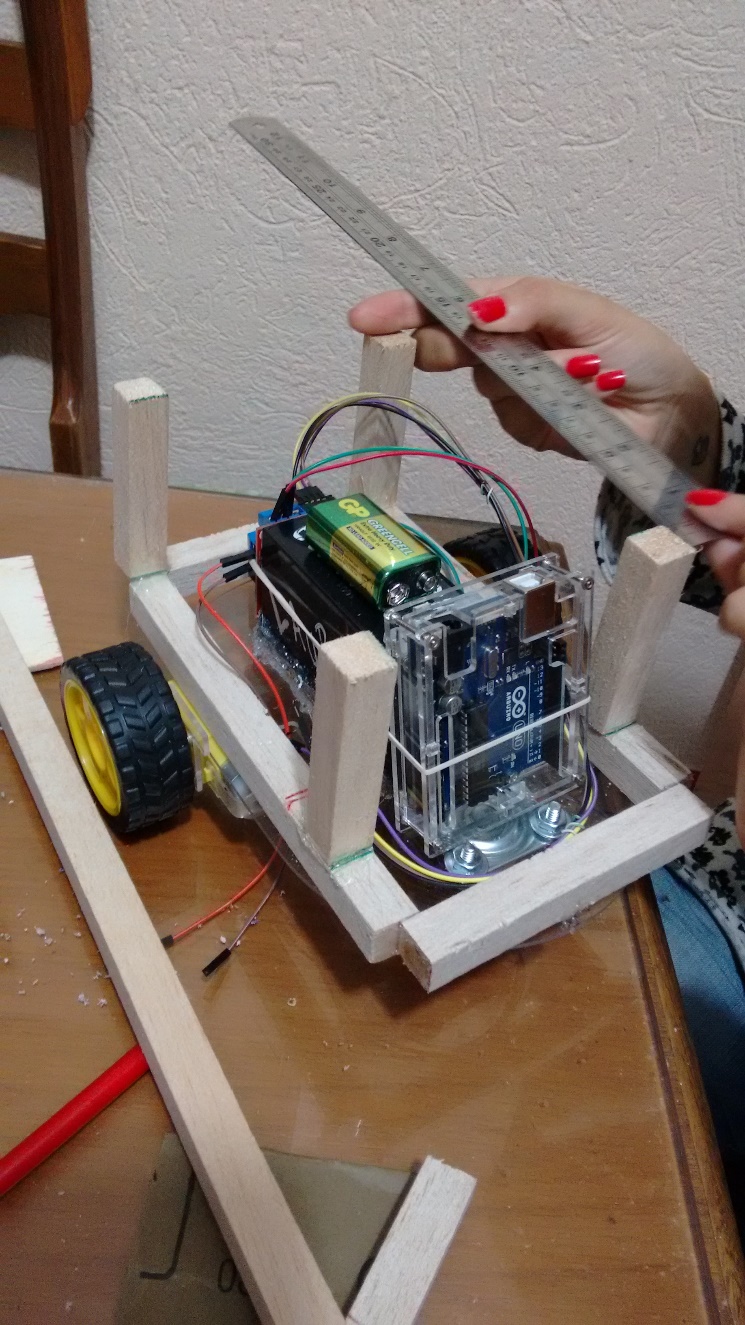
**Ruedas:** Las ruedas del robot son movidas por los motores. Normalmente se usan ruedas de materiales anti-deslizantes para evitar fallas de tracción. Su tamaño es otro factor a tener en cuenta a la hora de armar el robot.

**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.

**Tarjeta de control:** La toma de decisiones y el control de los motores están generalmente a cargo de un microcontrolador. La tarjeta de control contiene dicho elemento, junto a otros componentes electrónicos básicos que requiere el microcontrolador para funcionar, En este caso se usara el ARDUINO UNO como tarjeta de control, con un microcontrolador ATMEGA 328P, y a través de éste se implementara un control tipo PID. El **control PID** es un mecanismo de **control** que a través de un lazo de retroalimentación permite regular la velocidad, temperatura, presión y flujo entre otras variables de un proceso en general. El controlador **PID** calcula la diferencia entre nuestra variable real contra la variable deseada.

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.

En el caso de este robot seguidor de línea, el controlador PID, (que es una rutina basada matemáticamente), procesara los datos del sensor, y lo utiliza para controlar la dirección (velocidad de cada motor), para de esta forma  mantenerlo  en curso.



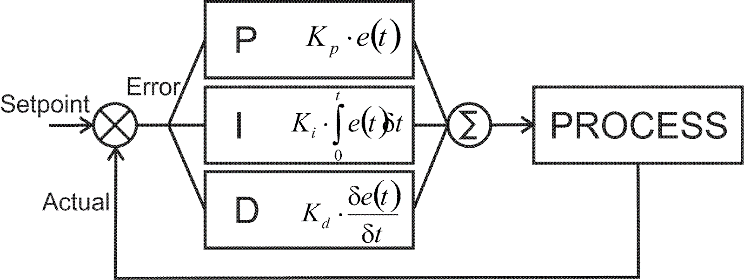
El PID (control proporcional, integral y derivativo) es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener (set point, target position  o punto de consigna), para aplicar una acción correctora que ajuste el proceso.

El controlador PID viene determinado por tres parámetros: el proporcional, el integral y el derivativo. Dependiendo de la modalidad del controlador alguno de estos valores puede ser 0, por ejemplo un controlador Proporcional tendrá el integral y el derivativo a 0 y un controlador PI solo el derivativo será 0, etc. Cada uno de estos parámetros influye en mayor medida sobre alguna característica de la salida (tiempo de establecimiento, sobre-oscilación,...) pero también influye sobre las demás, por lo que por mucho que ajustemos no encontraríamos un PID que redujera el tiempo de establecimiento a 0, la sobre-oscilación a 0, el error a 0,... sino que se trata más de ajustarlo a un término medio cumpliendo las especificaciones requeridas.

**-Proporcional:** Es la respuesta  al error que se tiene que entregar de manera inmediata, es decir, si nos encontramos en el centro de la línea, los motores, tendrán en respuesta una velocidad de igual  valor, si nos alejamos del centro, uno de los motores reducirá su velocidad y el otro aumentara.

**-Integral:** La integral es la sumatoria de los errores acumulados, tiene como propósito el disminuir y eliminar el error en estado estacionario provocado por el modo proporcional, en otras palabras, si el robot velocista se encuentra mucho tiempo

**Derivativo:** Es la derivada del error, su función es mantener el error al mínimo, corrigiéndolo proporcionalmente con la mismo velocidad que se produce, de esta manera evita que el error se incremente, en otra palabra, anticipara la acción evitando así las oscilaciones excesivas.

[](http://3.bp.blogspot.com/-6w08_oEbwVE/VIFmmdGrzaI/AAAAAAAAATs/Qt8ig581Fag/s1600/PIDController+(1).png)

***Fig.3.*** Esquema Control PID (Constantes y Parámetros)

**CONSTANTES:**  
 **Factor (Kp)** - Es un valor constante utilizado para aumentar o reducir el impacto de Proporcional. Si el valor es excesivo, el robot tendera responder inestablemente, oscilando  excesivamente. Si el valor es muy pequeño, el robot responderá muy lentamente,  tendiendo a salirse de las curvas.  
  
**Factor (Ki)** - Es un valor constante utilizado para aumentar o reducir el impacto de la Integral, El valor excesivo de este provocara oscilaciones excesivas, Un valor demasiado bajo no causara impacto alguno.  
  
**Factor (Kd)** - Es un valor constante utilizado para aumentar o reducir el impacto de la Derivada. Un valor excesivo provocara  una sobre amortiguación.  Provocando inestabilidad.

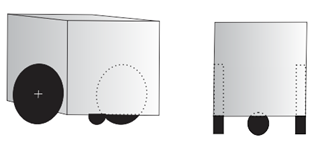
**Salida\_pwm =**

**Proporcional \* Kp + derivativo \* Kd + integral\*Ki**



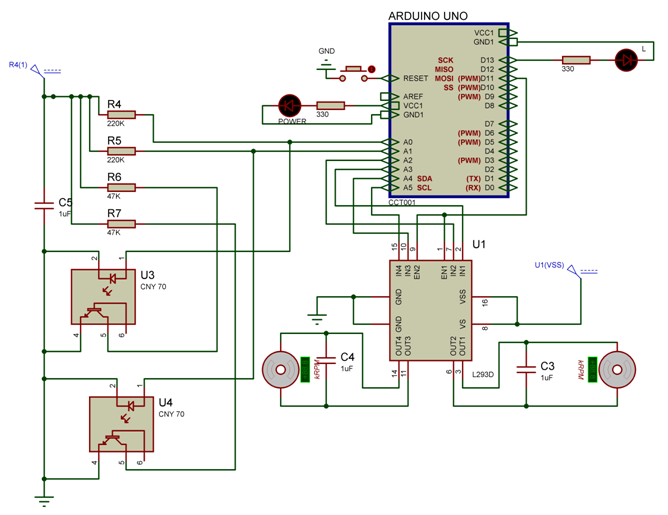
El robot actuará como un móvil seguidor de línea negra haciendo uso de sensores infrarrojos, los cuales detectan la línea y envían una señal al microcontrolador (Arduino) del robot. Allí, de acuerdo a la programación realizada, en éste caso respecto a la se tomará una decisión respecto a las acciones que el robot debe realizar para mantenerse en el camino y llegar a la meta. Este diseño electrónico cuenta con dos sistemas: percepción y potencia.

 La señal de salida obtenida de los sensores infrarrojos se convierte en la señal de entrada a la fase de programación, en la que interviene un módulo electrónico (Arduino) para el control del robot. Para ello se utilizará un puente H quien realizará la función de indicar a los motores que actividad realizar dependiendo del momento.



***Fig.3.*** Diseño del Robot

Para la simulación en Proteus se tiene en cuenta la señal de salida de los sensores infrarrojos, es de tipo análoga porque su variación depende de la distancia entre los componentes, la cantidad de luz percibida y la línea negra, de la incidencia de la luz ambiente y de las pérdidas en el flujo de corriente en todo el circuito. Esta señal entra a la fase de programación y allí es transformada en una señal digital, la cual a su vez es la entrada al driver de los motores (Puente H).



***Fig.4.*** Simulación en Proteus

**SINTONIZACION PID**  
  
La sintonización PID, es aquí donde se tendrá que buscar las constantes que correspondan a las características físicas del robot, la forma más fácil de hacerlo es por ensayo y error, hasta obtener el valor deseado, siguiendo estos pasos:

1. Comience con Kp, Ki y Kd igualando 0 y trabajar con Kp primero. Pruebe  establecer Kp a un valor de 1 y observar el robot. El objetivo es conseguir que el robot siga la línea, incluso si es muy inestable. Si el robot llega más allá y pierde la línea, reducir el valor de Kp. Si el robot no puede navegar por una vez, o parece lenta, aumente el valor Kp.
2. Una vez que el robot es capaz de seguir un poco la línea, asignar un valor de 1 a Kd .Intente aumentar este valor hasta que vea menos oscilaciones.
3. Una vez que el robot es bastante estable en la línea siguiente, asigne un valor de 0,5 a 1,0 a Ki. Si el valor de Ki es demasiado alta, el robot se sacudirá izquierda y derecha rápidamente. Si es demasiado baja, no se verá ninguna diferencia perceptible. El Integral es acumulativo por lo tanto  el valor Ki tiene un impacto significativo.  puede terminar  ajustando por 0,01 incrementos.
4. Una vez que el robot está siguiendo la línea con una buena precisión, se puede aumentar la velocidad y ver si todavía es capaz de seguir la línea. La velocidad afecta el controlador PID y requerirá re-sintonizar como los cambios de velocidad.

**INCLUYENDO LA LIBRERÍA QTRSensors:**

La instalación es como cualquier otra librería convencional de Arduino, previamente descargada se procede a añadir ésta en la carpeta donde están instaladas las librerías de Arduino.

En la librería  QTRSensors se tiene la opción de cambiar el tipo de línea que queramos sensar, cambiando desde la función de lectura de línea:

**qtrrc.readLine (sensorValues, QTR\_EMITTERS\_ON, 0);**

En las indicaciones del Datasheet y las funciones .h incluidas en esta librería nos dice que si queremos sensar líneas blancas, entonces debemos colocar en el último parámetro de la función un "1" en lugar del 0 que esta por defecto para las  líneas negras. 

Se agregan 3 nuevos parámetros a la librería: flanco color, en línea, ruido.  
  
**qtrrc.readLine (sensorValues, QTR\_EMITTERS\_ON, 1, flanco color, en\_linea, ruido);**  
  
**flanco color:** Este parámetro sirve para aumentar  o disminuir valores de lectura. Como sabemos cada  sensor lee valores desde 0 (mínima reflactancia) hasta 1000 (máxima reflactancia), pero debido a que los sensores tienen un margen de error, no

Se llega a obtener un valor adecuado, por eso se creó una variable que aumentara o disminuyera esos valores forzosamente para así obtener valores ideales que sean medibles. El valor por defecto es 0.  
  
**en\_linea:** Con este parámetro podemos variar la forma en que la librería  tomara los valores de los sensores y los considerara "si está o no en una línea", luego la librería tomara los valores de cada sensor y los  promediara para así obtener un valor de posición con respecto a esa línea. Su valor por defecto es de 200.  
  
**Ruido:** Con este parámetro se podrá variar  desde que valores  de reflactancia es  considerado como ruido, el valor por defecto es de 50.

# **CONCLUSIONES:**

Los controladores PID permiten mejorar la respuesta de un sistema, aunque esta respuesta no siempre sea óptima. Las reglas de ajuste propuestas presentan una forma de obtener los parámetros del controlador PID, siempre y cuando se tenga un modelo matemático del sistema. Un controlador PID permite que la respuesta de un sistema pueda llegar a tener un error nulo.

Obteniendo los parámetros de un controlador PID y observando la respuesta del controlador y el sistema, se puede trabajar en un sistema que permita obtener esos parámetros de manera autónoma y así permitir que el controlador PID pueda ser auto-ajustado.

Los microcontroladores son capaces de recibir información del ambiente o entorno (conexiones y módulos) en el que se encuentren, procesarla, manipularla y controlar posteriormente una salida en la que pueden encontrarse motores, luces, sonido, entre otras

#include <QTRSensors.h>

//#include <LiquidCrystal.h>

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

#define TIMEOUT 2500 // 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 mot\_i 7

#define mot\_d 8

#define sensores 6

#define boton\_1 2 //pin para boton

#define pin\_pwm\_i 9

#define pin\_pwm\_d 10

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variables Globales \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

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

//LiquidCrystal lcd(5, 4, 3, 2, 1, 0); // Inicializa la libreria con los números de los pines de interfaz

/\*

int Trimer0 = A0; // Flotante de 32 bit: Pin analógico utilizado para conectar el potenciómetro

int Trimer1 = A1; // Flotante de 32 bit: Pin analógico utilizado para conectar el potenciómetro

int Trimer2 = A2; // Flotante de 32 bit: Pin analógico utilizado para conectar el potenciómetro

\*/

//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=160; // Variable para la velocidad, el maximo es 255

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Constantes del PID \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

float Kp=0.18; // Constante Proporsional

float Ki=0.01; // Constante Integral

float Kd=2; // Constante Derivativa

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variables para el control del sensado \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

int flanco\_color =0; // Aumenta o disminuye el valor del sensado

int en\_linea=500; // Valor al que considerara si el sensor esta en linea o no

int ruido= 50; // Valor al cual el valor del sensor es considerado como ruido

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Inicio Programa Principal \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void setup()

{

//lcd.begin(16, 2); // set up the LCD's number of columns and rows:

delay(800);

pinMode(mot\_i, OUTPUT);//pin de direccion motor izquierdo

pinMode(mot\_d, OUTPUT);//pin de direccion motor derecho

pinMode(led1, OUTPUT); //led1

pinMode(led2, OUTPUT); //led2

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

for (int i = 0; i < 50; 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 x=digitalRead(boton\_1); //leemos y guardamos el valor del boton en variable x

delay(100);

if(x==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

}

}

}

void loop()

{

/\*

Kp = analogRead(Trimer0); // Lee el valor del potenciómetro (valor entre 0 y 1023)

Kp = Kp/1000;

Ki = analogRead(Trimer1); // Lee el valor del potenciómetro (valor entre 0 y 1023)

Ki = Ki/1000;

Kd = analogRead(Trimer2); // Lee el valor del potenciómetro (valor entre 0 y 1023)

\*/

// //ADC\_Potenciometro(); // Invoca la función donde lee y escribe el valor del ADC

// //Mensajes\_LCD(Kp,Ki,Kd);

PID(linea, velocidad, Kp, Ki, Kd, flanco\_color, en\_linea, ruido); //funcion para algoritmo pid(modificado )

//(tipo\_linea,velocidad,kp,ki,kd,flanco\_color,en\_linea,ruido)

Frenos\_Contorno(linea,700); // Funcion para frenado en curvas tipo flanco de comparación va desde 0 hasta 1000,

// esto para ver si esta en negro o blanco

}

////\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Funciones para el control del robot \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

////\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Inicio Funciones \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

////\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Declaración Funciones \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//aqui esta modificado la funcion del pid para que reciba los nuevos parametros para la libreria modificada

void PID(int linea, int velocidad, float Kp, float Ki, float Kd,int flanco\_color, int en\_linea,int ruido)

{

position = qtrrc.readLine(sensorValues, QTR\_EMITTERS\_ON, linea,flanco\_color, en\_linea, ruido ); //0 para linea negra, 1 para linea blanca

proporcional = (position) - 3500; // set point es 3500, asi obtenemos el error

integral=integral + proporcional\_pasado; //obteniendo integral

derivativo = (proporcional - proporcional\_pasado); //obteniedo el derivativo

if (integral>1000) integral=1000; //limitamos la integral para no causar problemas

if (integral<-1000) integral=-1000;

salida\_pwm =( proporcional \* Kp ) + ( derivativo \* Kd )+(integral\*Ki);

if ( salida\_pwm > velocidad ) salida\_pwm = velocidad; //limitamos la salida de pwm

if ( salida\_pwm < -velocidad ) salida\_pwm = -velocidad;

if (salida\_pwm < 0)

{

motores(velocidad+salida\_pwm, velocidad);

}

if (salida\_pwm >0)

{

motores(velocidad, velocidad-salida\_pwm);

}

proporcional\_pasado = proporcional;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

void motores(int motor\_izq, int motor\_der)

{

if ( motor\_izq >= 0 ) // Motor izquierdo

{

digitalWrite(mot\_i,HIGH); // con high avanza

analogWrite(pin\_pwm\_i,255-motor\_izq); // se controla de manera inversa para mayor control

}

else

{

digitalWrite(mot\_i,LOW); // con low retrocede

motor\_izq = motor\_izq\*(-1); // cambio de signo

analogWrite(pin\_pwm\_i,motor\_izq);

}

if ( motor\_der >= 0 ) //motor derecho

{

digitalWrite(mot\_d,HIGH); //

analogWrite(pin\_pwm\_d,255-motor\_der); //

}

else

{

digitalWrite(mot\_d,LOW);

motor\_der= motor\_der\*(-1);

analogWrite(pin\_pwm\_d,motor\_der);

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

void Frenos\_Contorno(int tipo,int flanco\_comparacion)

{

if(tipo==0) // Si tipo = 0 la linea es NEGRA y el fondo BLANCO

{

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

{

motores(-80,90); // debido a la inercia, el motor tendera a seguri girando por eso

// le damos para atras , para que frene lo mas rapido posible

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;

}

}

}

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

{

motores(90,-80);

while(true)

{

qtrrc.read(sensorValues);

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

{

break;

}

}

}

}

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

if(tipo==1) // Si tipo = 1 la linea es BLANCA y el fondo NEGRO

{

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

{

motores(-80,90); // debido a la inercia, el motor tendera a seguri girando por eso le damos para atras,

// para que frene lo mas rapido posible

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;

}

}

}

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

{

motores(90,-80);

while(true)

{

qtrrc.read(sensorValues);

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

{

break;

}

}

}

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

/\*void Mensajes\_LCD(float Kp, float Ki, float Kd)

{

lcd.setCursor(0, 0); // Ponte en la line 1, posicion 6

lcd.print("Kp=");// Print a message to the LCD.

lcd.print(Kp);// Print a message to the LCD.

lcd.setCursor(8, 0); // Ponte en la line 1, posicion 6

lcd.print("Ki=");// Print a message to the LCD.

lcd.print(Ki);// Print a message to the LCD.

lcd.setCursor(5, 1); // Ponte en la line 1, posicion 6

lcd.print("Kd=");// Print a message to the LCD.

lcd.print(Kd);// Print a message to the LCD.

delay(500);

lcd.clear();

}\*/

**Referencias**

[1] Ing. Capacho, Luis Miguel, Asesorías y *apuntes en clase*

[2] Ing. Lopez, Gerardo, Asesorías y *apuntes de laboratorio*.

1. [↑](#footnote-ref-1)