[[1]](#footnote-1)

Documentación acorde al segundo avance del 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.

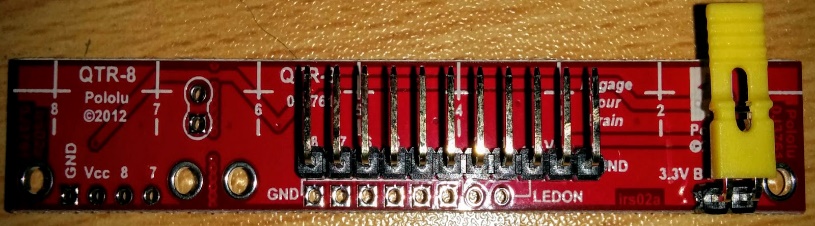
1. ***DESARROLLO:***

**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.

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.

**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.

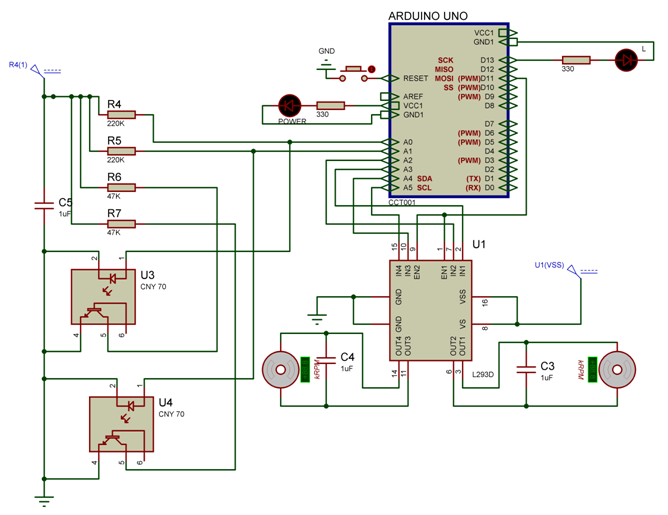
**Salida\_pwm =**

**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.

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.3.*** Simulación en Proteus

# **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.

#include <QTRSensors.h>

#define NUM\_SENSORS   5 //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 mot\_i         7

#define mot\_d         8

#define sensores      6

#define pin\_pwm\_i     9

#define pin\_pwm\_d     10

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

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

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

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

             delay(100);

             break; //saltamos hacia el bucle principal

          }

        }

}

void loop()

{  
  
                    //pid(0, 120, 0.18, 4, 0.001);

  pid(linea,velocidad,Kp,Ki,Kd); //funcion para algoritmo pid  
                        //(tipo,flanco de comparacion)  
  
                    //frenos\_contorno(0,700);

  frenos\_contorno(linea,700); //funcion para frenado en curvas tipo

                          //0 para lineas negras, tipo 1 para lineas blancas

                  //flanco de comparación va desde 0 hasta 1000 , esto para ver

                 //si esta en negro o blanco

}

////////funciones para el control del robot////

 void pid(int linea, int velocidad, float Kp, float Ki, float Kd)

{

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

}

}

**Referencias**

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

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

1. [↑](#footnote-ref-1)