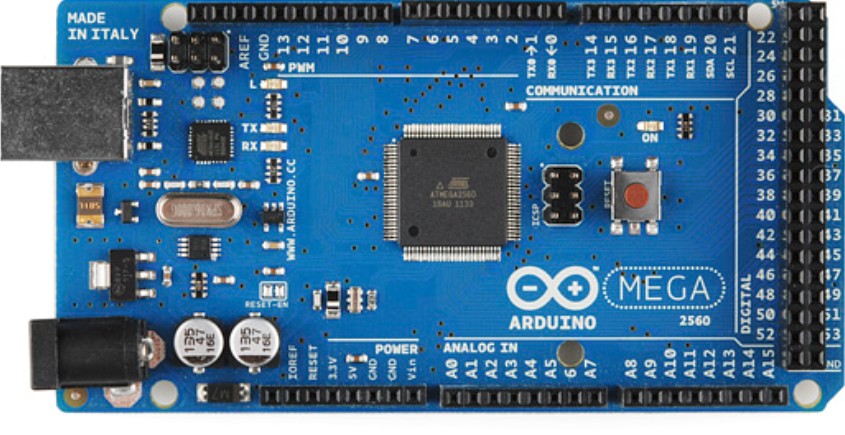
**IMPLEMENTACIÓN DE UN SEGUIDOR DE LINEA CON CONTROL PID EN ARDUINO**

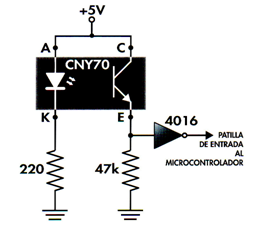
Materiales a usar:

- Arduino Mega (Atmega 2560)

  
*Fig. 1 Arduino MEGA*

- Optoacoplador reflexivo (referencia aún no definida)

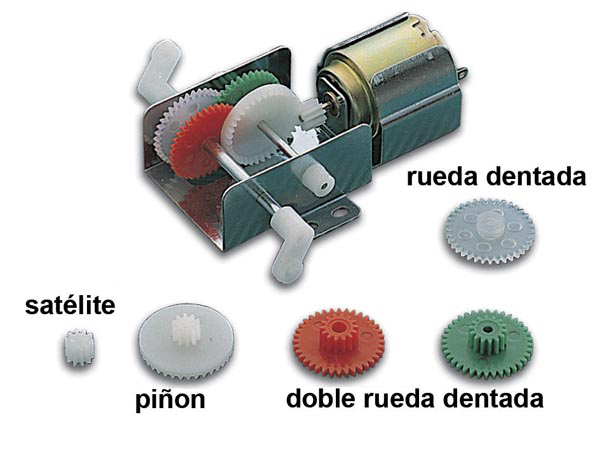
  
*Fig. 2 Optoacoplador reflexivo*

  
*Fig. 3 Diagrama del sensor en módulo*

- Carro (Se está decidiendo en usar 2 ruedas de un carro en una plataforma, o usar la carrocería de un pequeño carro a control remoto).

  
*Fig. 4 llanta ejemplo*

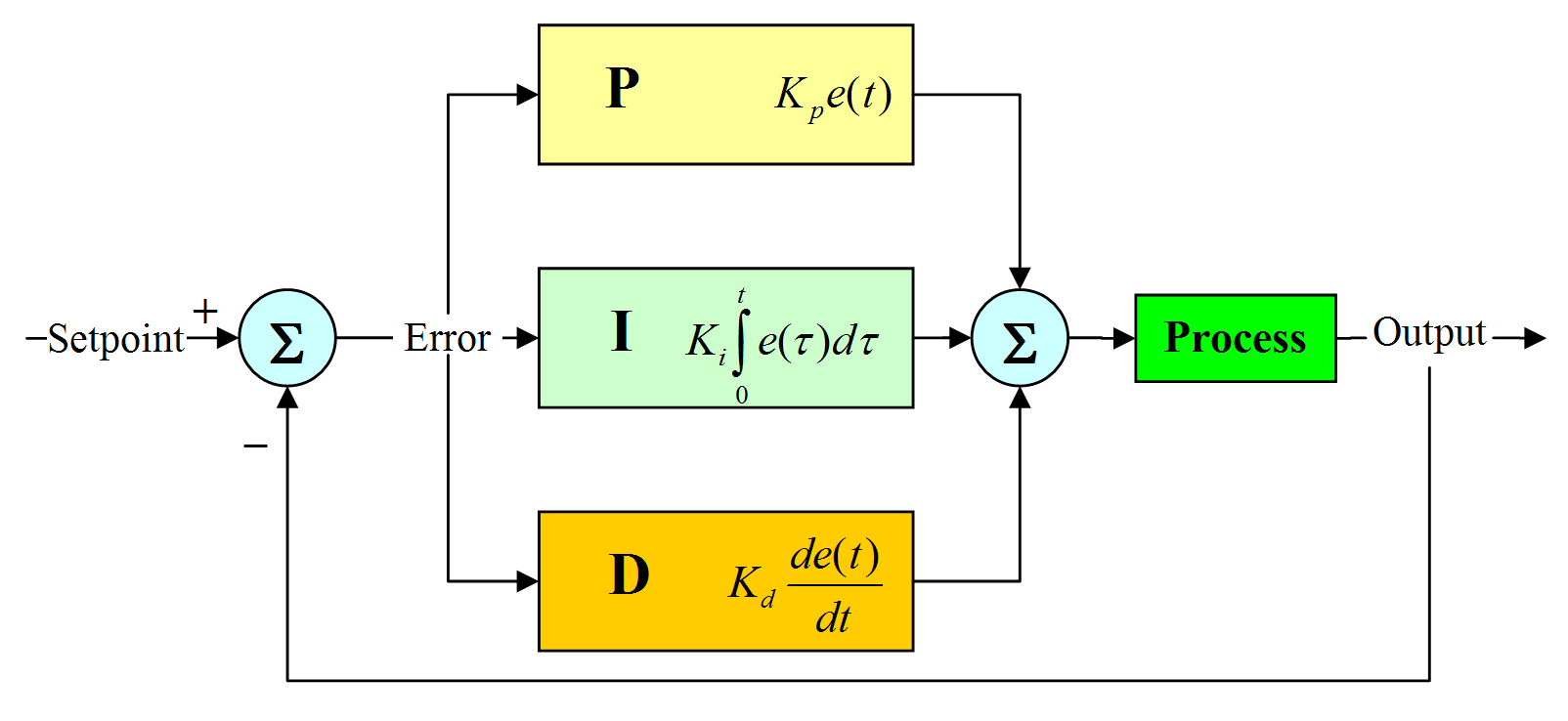
-Motores inducidos con piñoneria para una mayor fuerza



*Fig. 5 Motor acoplado con piñones*

**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. 6 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.

Para el diseño del PID se tiene en cuenta el esquema de la figura4, la cual se rige bajo la siguiente ecuación:

**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

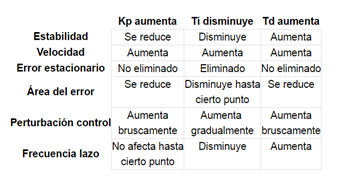
* Reducir el TI hasta anular el error en estado estacionario, aunque la oscilación sea excesiva.
* Disminuir ligeramente la ganancia.
* Repetir hasta obtener las características de respuesta deseadas.



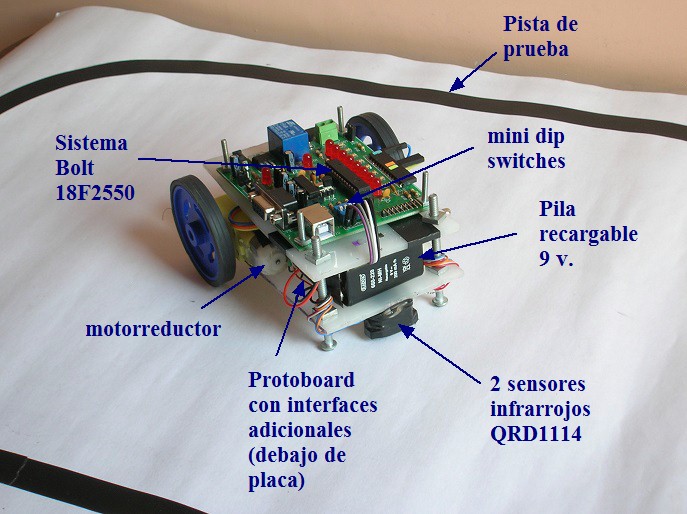
Paso 3: Acción Derivativa

* 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.

Los anteriores pasos se pueden evidenciar mejor en la siguiente tabla, en la que se puede observar los cambios que tiene el sistema variando los anteriores parámetros.



*Fig. 7 Módulo Arduino optoacopladores reflexivos*



*Fig. 8 Seguidor de linea Ideal*

Para la implementación del código se va a usar el siguiente algoritmo de arduino:

#define NUM\_SENSORS 8 //número 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

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

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

//variables para almacenar valores de sensores y posición

unsigned int sensorValues[NUM\_SENSORS];

unsigned int position=0;

/// variables para el pid

int derivativo=0, proporcional=0, integral=0; //parámetros

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

// AQUÍ CAMBIAREMOS LOS PARÁMETROS DE NUESTRO ROBOT

int velocidad=70; //variable para la velocidad, el máximo es 255

float Kp=0.18;

float Kd=2;

float Ki=0.01; //constantes

//variables para el control del sensado

int linea=0; // 0 para líneas negra, 1 para líneas 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

void setup()

{

delay(800);

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

pinMode(mot\_d, OUTPUT);//pin de dirección 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()

{

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////

//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)

{

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) //para linea blanca con 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;

}

}

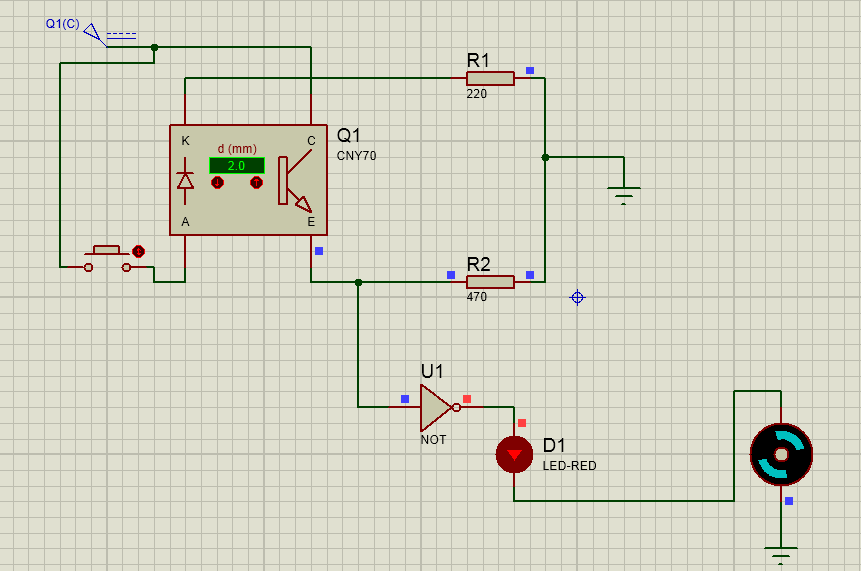
}

}

}

El anterior código está sujeto a cambios de acuerdo a los avances que se tenga en el proyecto.

**SIMULACIÓN DEL MÓDULO DEL OPTOACOPLADOR**



El sensor detecta cuando no hay luz incidiendo sobre él y esto es posible gracias al color oscuro que proporciona la cinta negra, si la luz es detectada el sensor envía una señal que es traducida como una desviación del vehículo y aunque acá es directa la forma en que se ve es mediante algoritmos y un controlador de motor lo que permitirá manipular la señal para un eficiente sistema.