Seguidor de línea con arduino

Andrés Felipe Veloz P, Diana Marcela Álzate, Juan Sebastián Sánchez Vega.

Universidad Del Quindío, Facultad de ingeniería, Armenia-Quindío.

***Resumen*—***en el presente artículo esta implementado en estado físico, simulado en Proteus y plataforma de programación en c de Arduino*

***Índice de términos***—Arduino, código, moto reductor,

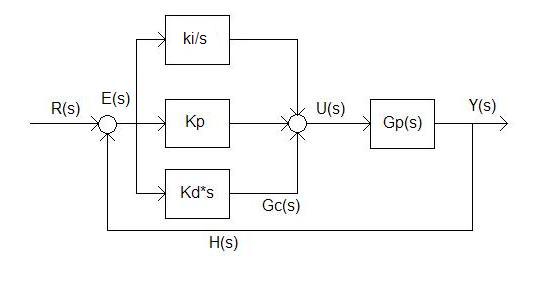
# Introducción

Se construyó desde la base un robot seguidor de línea que utiliza arduino como procesador de información, el cual se modela en la aplicación proteus y la parte lógica en la plataforma de arduino uno.

Este diseño se realizó utilizando las señales en el PWM para la conexión de motores y las señales PID como el procesamiento para el seguimiento en la línea.

# marco teórico.

Un controlador PID (Proporcional Integrativo Derivativo) es un mecanismo de control genérico sobre una realimentación de bucle cerrado, ampliamente usado en la industria para el control de sistemas. El PID es un sistema al que le entra un error calculado a partir de la salida deseada menos la salida obtenida y su salida es utilizada como entrada en el sistema que queremos controlar. El controlador intenta minimizar el error ajustando la entrada del sistema.



**Figura. Nº1. Control general del PID.**

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 sera 0, etc. Cada uno de estos parametros influye en mayor medida sobre alguna característica de la salida (tiempo de establecimiento, sobreoscilacion,...) pero también influye sobre las demas, por lo que por mucho que ajustemos no encontrariamos un PID que redujera el tiempo de establecimiento a 0, la sobreoscilacion a 0, el error a 0,... sino que se trata mas de ajustarlo a un término medio cumpliendo las especificaciones requeridas.

**Acción proporcional**

La respuesta proporcional es la base de los tres modos de control, si los otros dos, control integral y control derivativo están presentes, éstos son sumados a la respuesta proporcional. “Proporcional” significa que el cambio presente en la salida del controlador es algún múltiplo del porcentaje del cambio en la medición.

Este múltiplo es llamado “ganancia” del controlador. Para algunos controladores, la acción proporcional es ajustada por medio de tal ajuste de ganancia, mientras que para otros se usa una “banda proporcional”. Ambos tienen los mismos propósitos y efectos.

**Acción integral**

La acción integral da una respuesta proporcional a la integral del error. Esta acción elimina el error en régimen estacionario, provocado por el modo proporcional. Por contra, se obtiene un mayor tiempo de establecimiento, una respuesta más lenta y el periodo de oscilación es mayor que en el caso de la acción proporcional.

**Acción derivativa**

La acción derivativa da una respuesta proporcional a la derivada del error (velocidad de cambio del error). Añadiendo esta acción de control a las anteriores se disminuye el exceso de sobreoscilaciones.

Existen diversos métodos de ajuste para controladores PID pero ninguno de ellos nos garantiza que siempre encuentre un PID que haga estable el sistema. Por lo que el más usado sigue siendo el metodo de prueba y error, probando paramentros del PID y en función de la salida obtenida variando estos parámetros. [1]

El arduino uno tiene un micro controlador basado en el ATmega328P. Tiene 14 pines de entrada/salida (los cuales usan salidas PWM), 6entradas análogas, un botón de reseteo. Este cuenta con todo lo necesario para soportar el micro controlador; una simple conexión al computador por medio de un cable USB o energía la cual es de un adaptador AC hacia DC o una batería para iniciar. Puedes manipular el UNO sin importar mucho de las equivocaciones, en caso de ser necesario el chip puede ser cambiado por unos cuantos dólares y empezar otra vez.

“UNO” mientras es Italiano y es elejida la marca de lanzamiento es el software de Arduino (IDE) 1.0 el UNO es una placa y versión 1.0 software Arduino (IDE) teniendo de referencia como la versión del Arduino, ahora avanzando para nuevos parámetros. La tarjeta del UNO es la primera en las series de tarjetas de Arduino con USB, y el modelo de referencia para la plataforma del arduino; por la extensa lista de corriente, pasan por las salidas de datos de la tarjeta que se ven en la tarjeta [2]

El desvanecimiento de demostraciones de ejemplos del uso de salidas analógicas (PWM) de apagar un LED. Esto esta disponible en “File - > Ketchbook - > Examples - > Analog” en el menú del software de Arduino.

Modulación de ancho de pulso o PWM en sus siglas en inglés, es la técnica que consigue resultados por medios digitales. El control digital es usado para crear una onda cuadrada, una señal que cambia en te el encendido y el apagado. Este encendido-apagado es un patrón que simula voltajes completo en alto (5 voltios) y bajo (0 voltios) por el cambio de la parte de la señal del paso en el tiempo en el versus el paso de la señal en el tiempo en bajo. La duración de “un tiempo” es llamado ancho de pulso. Esto varía los valores análogos, usted cambia, o modula, el ancho de pulso. Esto se repite entre el bajo-alto lo suficientemente rápido el patrón con un LED por ejemplo, el resultado es de la señal de voltaje estable entre 0V y 5V controlando el brillo del LED.

En el gráfico siguiente, las líneas verdes representan un período de tiempo regular. Esta duración o período es la inversa de la frecuencia PWM. En otras palabras, con frecuencia PWM de Arduino a unos 500Hz, las líneas verdes mediría 2 milisegundos. Una llamada a analogWrite() es en una escala de 0 - 255, tal que analogWrite(255) pide un ciclo de deber del 100% (siempre encendido), y analogWrite(127) corresponde a un 50% del ciclo (en la mitad del tiempo) por ejemplo.

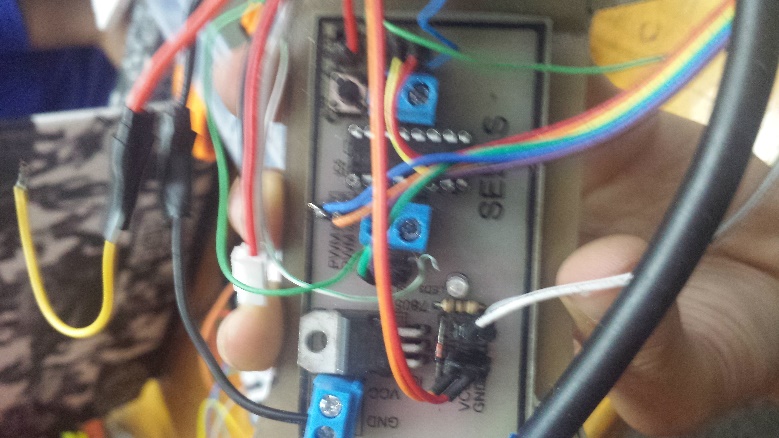


**Figura Nº2. duracion del pulso por PWM**

Una vez que este ejemplo funcionando, se toma el arduino y moverlo hacia delante y hacia atrás. Lo que está haciendo aquí es esencialmente tras tiempo a través del espacio. A nuestros ojos, el movimiento desdibuja cada parpadeo de LED en una línea. Como el LED se va adentro y hacia fuera, ésas líneas poco crecen y decrecen en longitud. Ahora están viendo el ancho de pulso. [3]

# Análisis y Procedimiento

Se inicializo el trabajo con la búsqueda de los elementos necesarios para el implante del seguidor de línea, en donde utilizamos 8 sensores CNY70 los cuales tienen una señal analógica, la cual se usa para leer la línea a seguir, dos (2) motores 12GA de 1000 RPM para la velocidad y la dirección del vehículo, unas celdas de batería para obtener un dispositivo inalámbrico.



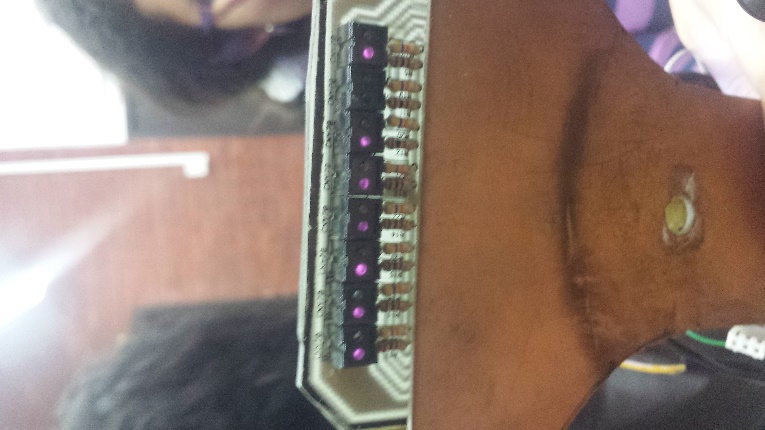
**Figura Nº3. Controlador de los motores.**

Controlador de los motores actúan como puente H el cual controla el flujo de los voltajes suministrados para regular la velocidad de giro y la capacidad de giro del carro, usando un amplificador lm293 como administrador de las ganancias deseadas de las señales enviadas del arduino.



**Figura Nº4. Motor utilizado.**

Tipo de motor utilizado 12GA de 1000RPM con llantas de goma adhesiva para mayor protección en las curvas, estos motores pueden ser conectados en polarización directa e inversa, teniendo capacidad de disminuir la velocidad y no la potencia del torque.



**Figura Nº5. Sensores probados.**

Sensores CNY70 teniendo como una distancia de alejamiento optimo contra el suelo de 3mm de separación, teniendo como función de control el PID, en la figura Nº5 se puede ver los ledes de los sensores funcionando con la iluminación ultravioleta, se usaron 8 sensores para usar la máxima capacidad de la placa de arduino.

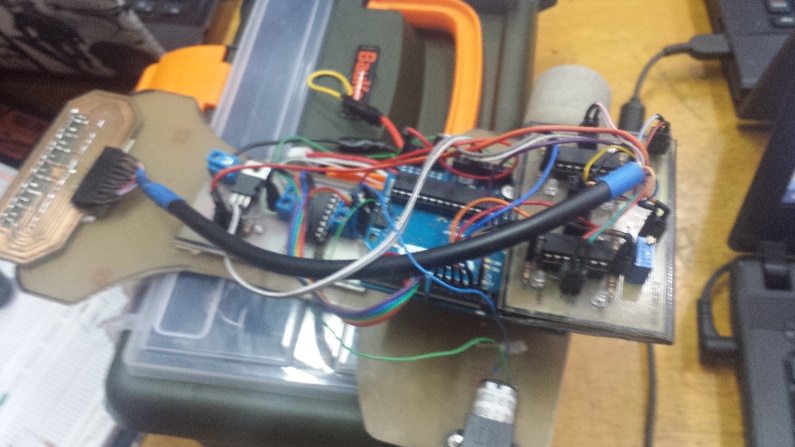
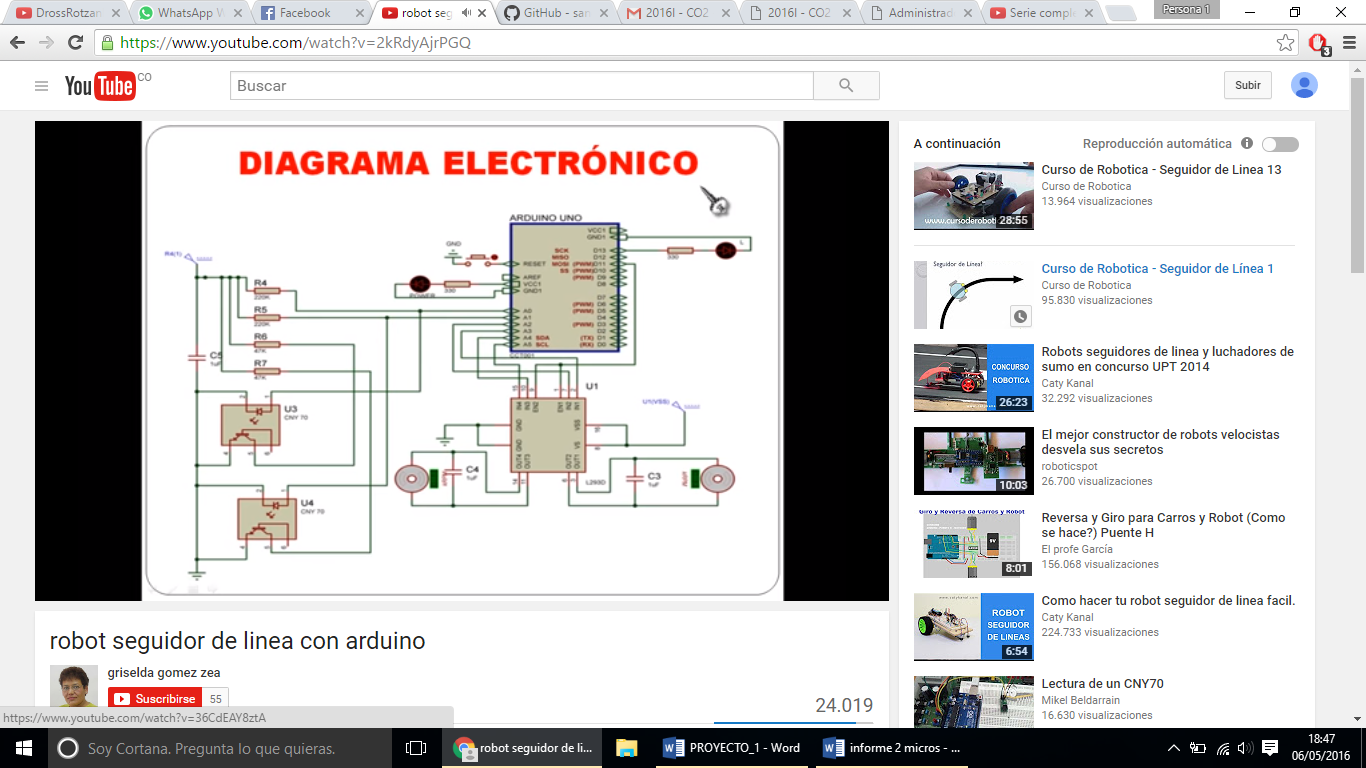


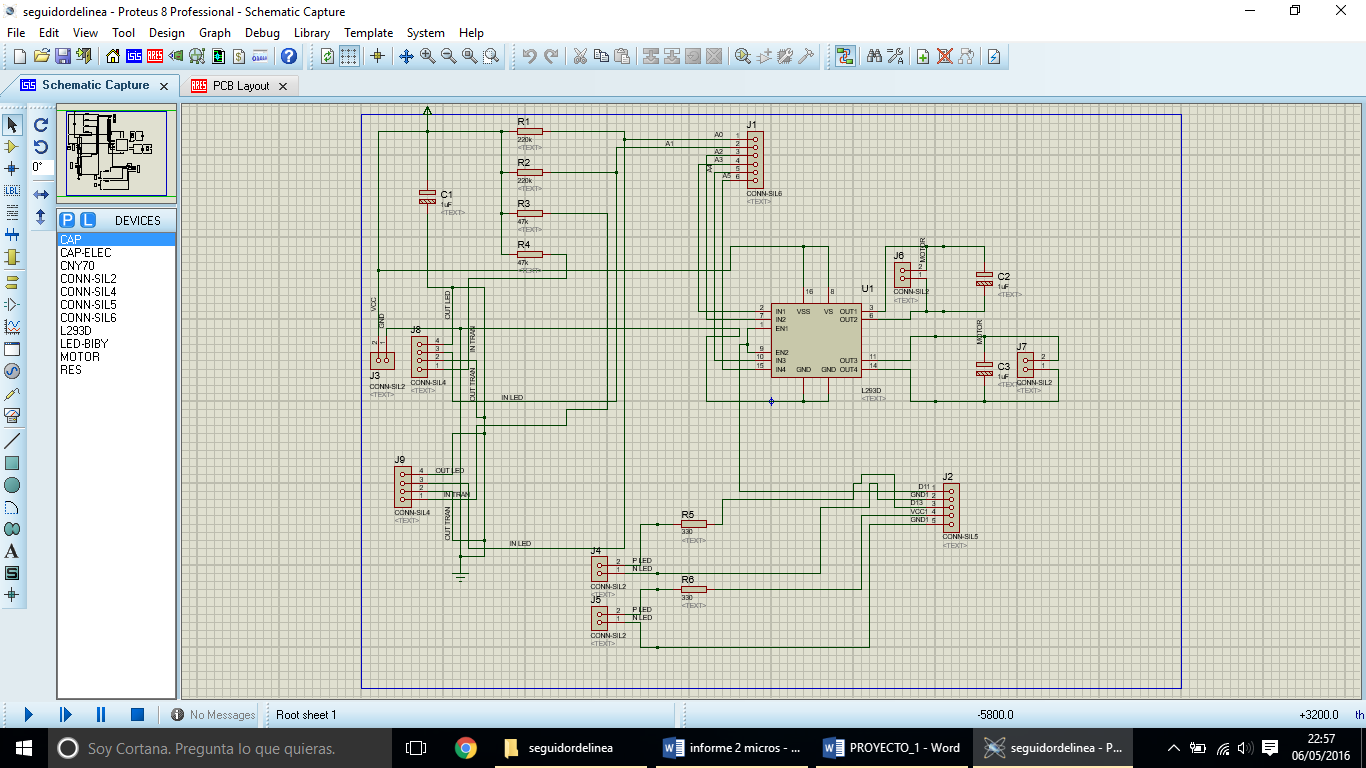
Figura Nº6. Robot completado.

El robot seguidor de linea completado con las conexiones con elarduino, el puente H los sensores y un arreglo para demostrar cuales sensores estan leyendo las señales con los ledes.



**Figura Nº7. Esquemático**

Esquemático de la conexión del puente H con los motores y el arduino, este a su vez es una guía de instalación de los componentes del robot seguidor de línea.



**Figura Nº8. Esquemático en proteus.**

El esquemático de la figura 5 esta diseñado para la implementación en una váquela.

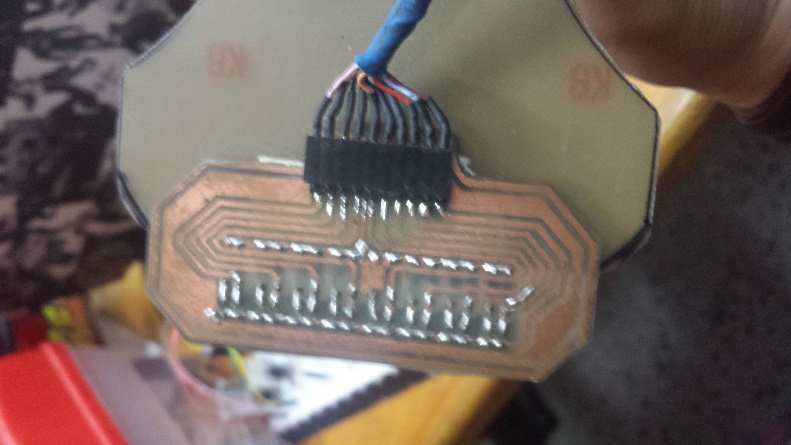


Figura Nº9. Circuito de conexión de los sensores quemados en la vaquera producida en ares de proteus.

A continuación se demuestra el código de programación del arduino con la lectura de los sensores y el controlador del puente H para la dirección del robot.

#define Left\_Dir1 5

#define Left\_Dir2 7

#define Left\_Speed 6

#define Right\_Dir1 2

#define Right\_Dir2 4

#define Right\_Speed 3

#define buzzer A4

#define button A5

int sensor[8] = { 8,9,10,11,12,13,14,15 }; // Pin assignment

int sensorReading[8] = { 0 }; // Sensor reading array

int memoria=0;

int s1=0;

int s2=0;

int s3=0;

int s4=0;

int s5=0;

int s6=0;

int s7=0;

int s8=0;

int actual=0;

int setpoint=0;

float activeSensor = 0; // Count active sensors

float totalSensor = 0; // Total sensor reading

float Kp = 700; // Max deviation = 8-4.5 = 3.5 || 255/3.5 = 72

float Ki = 1;

float Kd = 1.2;

float error = 0;

float previousError = 0;

float totalError = 0;

float power = 0;

int PWM\_Right, PWM\_Left;

void setup()

{

Serial.begin(9600);

pinMode(Left\_Dir1, OUTPUT);

pinMode(Left\_Dir2, OUTPUT);

pinMode(Left\_Speed, OUTPUT);

pinMode(Right\_Dir1, OUTPUT);

pinMode(Right\_Dir2, OUTPUT);

pinMode(Right\_Speed, OUTPUT);

pinMode(button, INPUT);

for(int i=0; i<8; i++) {

pinMode(sensor[i], INPUT);

}

void Stop();

void Forward();

while(1) {

if(boolean checkButton()==1)

{

break; }

}

}

void loop()

{

void lineFollow();

}

void Left\_Forward()

{

digitalWrite(Left\_Dir1, LOW);

digitalWrite(Left\_Dir2, HIGH);

}

void Left\_Reverse()

{

digitalWrite(Left\_Dir1, HIGH);

digitalWrite(Left\_Dir2, LOW);

}

void Right\_Forward()

{

digitalWrite(Right\_Dir1, LOW);

digitalWrite(Right\_Dir2, HIGH);

}

void Right\_Reverse()

{

digitalWrite(Right\_Dir1, HIGH);

digitalWrite(Right\_Dir2, LOW);

}

void Go(int s)

{

analogWrite(Right\_Speed, s);

analogWrite(Left\_Speed, s);

}

void Forward()

{

Left\_Forward();

Right\_Forward();

}

void Reverse()

{

Left\_Reverse();

Right\_Reverse();

}

void Turn\_Left()

{

Right\_Forward();

Left\_Reverse();

}

void Turn\_Right()

{

Right\_Reverse();

Left\_Forward();

}

void Stop()

{

analogWrite(Left\_Speed,0);

analogWrite(Right\_Speed, 0);

}

boolean checkButton()

{

if(digitalRead(button)==1) {

delay(5);

if(digitalRead(button)==1) {

return 1;

}

}

return 0;

}

void PID\_program()

{

// readSensor();

s1=digitalRead(8);

s8=digitalRead(15);

s2=digitalRead(9);

s3=digitalRead(10);

s4=digitalRead(11);

s5=digitalRead(12);

s6=digitalRead(13);

s7=digitalRead(14);

if((s1==1)&&(s2==0)&&(s3==0)&&(s4==0)&&(s5==0)&&(s6==0)&&(s7==0)&&(s8==0))

{

actual=-6;

}

if((s1==1)&&(s2==1)&&(s3==0)&&(s4==0)&&(s5==0)&&(s6==0)&&(s7==0)&&(s8==0))

{

actual=-5;

}

if((s1==1)&&(s2==1)&&(s3==1)&&(s4==0)&&(s5==0)&&(s6==0)&&(s7==0)&&(s8==0))

{

actual=-4;

} if((s1==0)&&(s2==1)&&(s3==1)&&(s4==0)&&(s5==0)&&(s6==0)&&(s7==0)&&(s8==0))

{

actual=-3;

}

if((s1==0)&&(s2==1)&&(s3==1)&&(s4==1)&&(s5==0)&&(s6==0)&&(s7==0)&&(s8==0))

{

actual=-2;

}

if((s1==0)&&(s2==0)&&(s3==1)&&(s4==1)&&(s5==0)&&(s6==0)&&(s7==0)&&(s8==0))

{

actual=-1;

}

if((s1==0)&&(s2==0)&&(s3==1)&&(s4==1)&&(s5==1)&&(s6==0)&&(s7==0)&&(s8==0))

{

actual=0;

}

if((s1==0)&&(s2==0)&&(s3==0)&&(s4==1)&&(s5==1)&&(s6==1)&&(s7==0)&&(s8==0))

{

actual=1;

}

if((s1==0)&&(s2==0)&&(s3==1)&&(s4==1)&&(s5==1)&&(s6==1)&&(s7==0)&&(s8==0))

{

actual=2;

}

if((s1==0)&&(s2==0)&&(s3==0)&&(s4==0)&&(s5==1)&&(s6==1)&&(s7==0)&&(s8==0))

{

actual=3;

}

if((s1==0)&&(s2==0)&&(s3==0)&&(s4==0)&&(s5==1)&&(s6==1)&&(s7==1)&&(s8==0))

{

actual=4;

}

if((s1==0)&&(s2==0)&&(s3==0)&&(s4==0)&&(s5==0)&&(s6==1)&&(s7==1)&&(s8==1))

{

actual=5;

} if((s1==0)&&(s2==0)&&(s3==0)&&(s4==0)&&(s5==0)&&(s6==1)&&(s7==1)&&(s8==0))

{

actual=6;

}

if((s1==0)&&(s2==0)&&(s3==0)&&(s4==0)&&(s5==0)&&(s6==0)&&(s7==1)&&(s8==1))

{

actual=7;

}

if((s1==0)&&(s2==0)&&(s3==0)&&(s4==0)&&(s5==0)&&(s6==0)&&(s7==0)&&(s8==1))

{

actual=8;

}

if((s1==0)&&(s2==0)&&(s3==0)&&(s4==0)&&(s5==0)&&(s6==0)&&(s7==0)&&(s8==0))

{

if(actual>8)

{

actual=9;

}

}

if((s1==0)&&(s2==0)&&(s3==0)&&(s4==0)&&(s5==0)&&(s6==0)&&(s7==0)&&(s8==0))

{

if(actual<0)

{

actual=-7;

}

}

error= actual - setpoint;

if(power<150 && power>-150);

totalError += error;

}

float previousError = error; // save previous error for differential

float power = (Kp\*error) + (Kd\*(error-previousError)) + (Ki\*totalError);

if( power>150.0 ) { power = 150.0; }

if( power<-150.0 ) { power = -150.0; }

// Serial.print("out : ");

// Serial.print(power);

Serial.print(" ki x error : ");

Serial.print(kp\*totalError);

Serial.print(" kd x error : ");

Serial.print(kd\*(error-previousError));

Serial.print(" actual : ");

Serial.print(actual);

if(power<0) // Turn left

{

PWM\_Right = 50;

PWM\_Left = 50 - abs(int(power));

}

else // Turn right

{

PWM\_Right = 50 - int(power);

PWM\_Left = 50;

}

}

void lineFollow(void) {

PID\_program();

analogWrite(Left\_Speed, PWM\_Left);

analogWrite(Right\_Speed, PWM\_Right);

}

void readSensor(void) {

for(int i=0; i<8; i++)

{

sensorReading[i] = !digitalRead(sensor[i]);

if(sensorReading[i]==1) { activeSensor+=1; }

totalSensor += sensorReading[i] \* (i+1);

}

avgSensor = totalSensor/activeSensor;

activeSensor = 0; totalSensor = 0;

}

void testSensor(void) {

for(int i=0; i<8; i++) {

Serial.print(!digitalRead(sensor[i]));

}

Serial.println("");

delay(20);

}

boolean checkSensor() {

boolean state = 0;

for(int i=0; i<8; i++) {

if(digitalRead(sensor[1])==0) {state=1;}

}

return state;

}

# Conclusiones

* La herramienta de la plataforma de programación para el arduino está basado en el lenguaje c y c++.
* Arduino es una plataforma multipropósitos que se puede utilizar para diversos casos.
* Las señales de los sensores PID son ecuaciones utilizadas para la estabilización y una mejor calidad de muestreo en la señal
* Diversos componentes existen para la manipulación entre los elementos, dando a una gran gama de combinaciones para formar un mismo robot, dando a diferentes características también.

# Bibliografía

|  |  |
| --- | --- |
| [1] | desconocido, «CONTROL-PID,» [En línea]. Available: http://control-pid.wikispaces.com/. [Último acceso: 03 05 2016]. |
| [2] | desconosido, «arduino,» [En línea]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Último acceso: 04 05 2016]. |
| [3] | T. Hirzel, «arduino,» [En línea]. Available: https://www.arduino.cc/en/Tutorial/PWM. [Último acceso: 04 05 2016]. |