SEGUIMIENTO DE TRAYECTORIAS EN UN ROBOT DIFERENCIAL POR MEDIO DE UN CONTROLADOR PID Y CODIFICADOR ÓPTICO

Reporte de práctica 2.2

Asignatura : Robótica

Profesor: Dr. José Hugo Barrón Zambrano

Alumno: Angel Arturo Ramírez Suárez

Maestría en Ingeniería

Fecha: Martes 19 de mayo de 2015

Cd. Victoria, Tam.

Índice

1. Introducción

La robótica representa una ciencia multidisciplinaria que comprende campos diversos de la ciencia y la ingeniería como son ingeniería mecánica y de materiales, electrónica, ciencia computacional y control, además de otras disciplinas como economía, inteligencia artificial o psicología que contribuyen a facilitar su objetivo, el estudio y creación de sistemas autómatas capaces de realizar diversas acciones de forma autónoma [1].

Los robots tienen una gran cantidad de aplicaciones diferentes entre las que se encuentran la manipulación de sustancias peligrosas para el ser humano o la realización de actividades en condiciones extremas (cambios de temperatura bruscos o alta radiación), además de la automatización de tareas repetititvas o de alta precisión [2].

Debido a la gran cantidad de elementos que componen a un sistema robótico, su elaboración resulta una tarea de alto costo. Esto se vuelve particularmente notable en los robots industriales o de propósitos muy específicos como es el caso de los robots médicos en los cuales se puede requerir una gran cantidad de recursos y tiempo para su elaboración y prototipado [2].

Entre las diversas aplicaciones que se da a la robótica se encuentra aplicaciones de rescate y exploración, en las cuales los robots requieren trasladarse a través de zonas peligrosas o no aptas para los seres humanos y realizar diversas tareas desde tomar muestras de elementos locales, transmitir información sobre el estado de las áreas, o en el caso de robots de rescate más complejos, transportar material delicado o seres humanos por medio de un conjunto de actuadores especializados.

Otra de las aplicaciones en las cuales la robótica y el control tienen gran uso es la industria para la manipulación de elementos peligrosos para el ser humano o la realización de tareas tediosas y repetitivas. En estas tareas los robots se destacan por su capacidad de mantenerse operando aún bajo condiciones extremas y sin sufrir los efectos de la fatiga que afectan a un ser humano.

Los robots sin embargo aún cuentan con ciertas limitaciones en su operación, particularmente respecto a condiciones altamente cambiantes o que requieren hacer uso de juicio y creatividad debido a las capacidades actuales de las aplicaciones que limitan el margen de operación de los dispositivos robóticos. En años recientes con el creciente interés en mejores y más féxibles algoritmos de inteligencia artificial, ha sido posible el generar robots capaces de responder a situaciones fuera de lo previsto por los programadores y aprender del contexto de las mismas para implementar soluciones a un mayor rango de problemas.

Para el correcto funcionamiento de un sistema robótico, es necesario el contar con retroalimentación que permita al sistema el saber cuando es necesario el corregir el movimiento y la velocidad y fuerza a aplicar en un movimiento determinado. Para ello existe una amplia gama de sensores que permiten a los sistemas robóticos el orientarse y establecer rutas en diferentes condiciones.

Entre los sensores más utilizados se encuentran los codificadores ópticos, los cuales son sensores que permiten transmitir información relativa al cambio de posición de las juntas del dispositivo con respecto al tiempo. Para la realización de esta práctica se hizo uso de codificadores ópticos acoplados a la flecha de los motores de corriente directa implementados en un robot diferencial, los cuales fueron utilizados para implementar un controlador PID para mantener la trayectoria del robot en una linea recta y medir la distancia y velocidad de desplazamiento del mismo utilizando el lenguaje de programación Arduino [3].

2. Desarrollo

2.1. Elaboración de robot diferencial

Se realizó la adquisición de una base de pruebas de un robot diferencial para la implementación de los algoritmos ilustrados. Esta base consta de dos ruedas diferenciales controladas por motores de corriente directa de 6 volts encargados de dar impulso a la plataforma, además de una rueda loca que sirve como soporte al resto de la estructura y que se ubica en la parte trasera del robot.

En la figura 1 se puede observar la plataforma adquirida en su estructura básica ensamblada y cada una de las partes que la componen.

Figura 1: Plataforma robótica adquirida.

Esta base fue modificada añadiendo una estructura superior en la cual se plantea colocar el resto de sensores y microcontrolador a utilizar en proyectos futuros. En la figura 2 se puede observar el robot con cada uno de los componentes y la segunda plataforma elaborada en cartón.

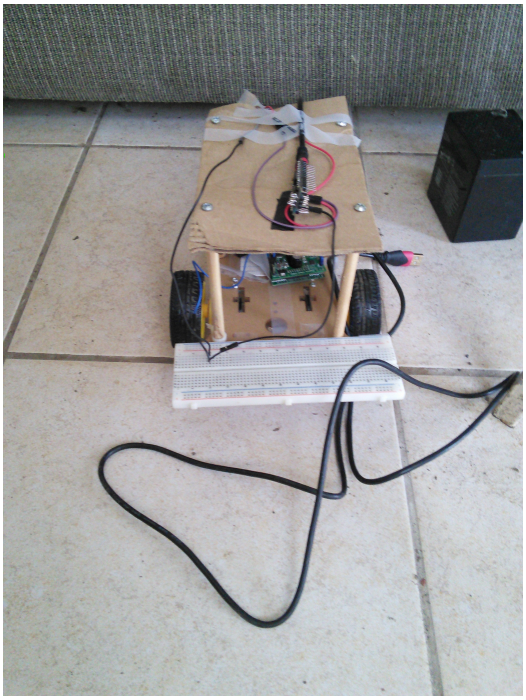


Figura 2: Robot elaborado para la implementación del controlador PID y de lazo abierto.

2.2. Control

Para el control del dispositivo se optó por implementar un microcontrolador, siendo en este caso seleccionado el dispositivo Arduino nano debido a las ventajas que ofrece respecto a dimensiones, costo y prestaciones ya que permite la implementación de señales PWM para el control de velocidad de los motores, puertos digitales para entrada y salida de señales digitales además de puertos analógicos para la captura de información por parte de los sensores instalados en la plataforma.

En la figura 3 se observa el arduino nano utilizado.

Cuadro 1: Principales características del Arduino nano

Característica	Valor
Microcontrolador	ATmega328
Voltaje de operación	5 volts
Entradas/Salidas digitales	14 con 6 de tipo PWM
Entradas analógicas	8 entradas
Corriente máxima por pin	40 mA
Memoria flash	32kB
EEPROM	1kB
Frecuencia de reloj	16 MHz

2.3. Etapa de potencia

Para la conexión entre la etapa de control compuesta por el Arduino uno en el cual se cargaron las aplicaciones, es necesario establecer una etapa de potencia encargada de suministrar la corriente requerida por los motores para operar. Para ello se utilizó una tarjeta controladora de motores de corriente directa Polulu T-REX cuyas características de operación se describen a continuación:

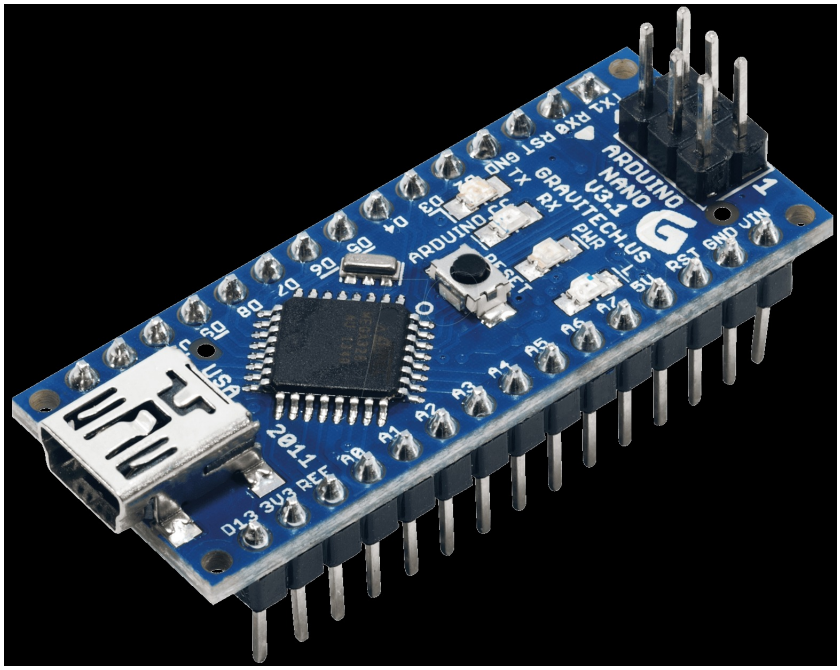


Figura 3: Microcontrolador para usos diversos Arduino nano.

Cuadro 2: Principales características del Arduino nano

Característica	Valor
Voltaje de operación	6 a 18 volts dc
Corriente máxima de operación	13 A por motor o 30 Amperes total
Puerto serial	RS-232 y TTL
Entradas analógicas	5 entradas
Entradas RC	5 entradas

Esta tarjeta que puede ser observada en la figura 4 fue elegida debido a la cantidad de potencia que es capaz de soportar y las diversas capacidades de control que permite. En este caso fue empleado el control por medio del envio de comandos por el puerto serial de la misma.

2.4. Alimentación

Para suministrar la corriente requerida para los motores se utilizó una batería recargable sellada de ácido-plomo con una capacidad de voltaje de 12 volts y 1.2 Amperes que puede observarse en la figura 5.

2.5. Sensado

Con el objetivo de mejorar la interacción del robot con el ambiente y reducir la incertidumbre en el control del mismo, se vuelve necesario el proveer al robot de la capacidad de percibir su ambiente. Para ello se hace uso de sensores que capturen información del ambiente y permitan al robot actuar en base a la información entregada por éstos.

Para la retroalimentación de la posición se hizo uso de un sensor ultrasónico modelo HC-SR04 que permite el cálculo de la distancia del robot por medio de la medición del tiempo de retorno de una señal sonora. El sensor opera al emitir un pulso a través del sensor y esperar la señal de retorno. Una vez que ha regresado,

Figura 4: Tarjeta de potencia para el control de los dos motores de corriente directa.

Figura 5: Bateria de ácido-plomo utilizada para la alimentación del robot.

calcula el tiempo de retorno de la señal, a través del cual obtiene la distancia considerando la velocidad de retorno del sonido.

En la figura 6 se puede observar el sensor utilizado para esta aplicación.

2.6. GPS

El sistema de posicionamiento global o GPS (Global Positioning System) es un sistema creado para permitir a los usuarios el contar con información respecto a la posición absoluta de un objeto relativo al eje terrestre, velocidad y tiempo de lectura por medio de la captura de información transmitida por un conjunto de satélites orbitando en la atmósfera terrestre.

Entre éstos, los más famosos son el sistema GALILEO y BAIDU que se encuentran en desarrollo por parte de la unión europea y China, además del sistema más utilizado conocido como GPSIII que continúa en desarrollo con cerca de 32 satélites [4].

Figura 6: Sensor ultrasónico HC-SR04 utilizado para la implementación del controlador PID.

Figura 7: Módulo GPS GY-GPS6MV1.

El sistema GPS opera a través de un conjunto de satélites que viajan a través de la atmósfera terrestre cuya posición es conocida dentro de un margen de tolerancia aceptado. Estos dispositivos transmiten sus coordenadas hacia la tierra en un intervalo conocido, señal que es capturada por los dispositivos pasivos en la tierra.

Los dispositivos pasivos se encargan de recibir la señal y calcular con respecto a la posición del GPS en el espacio, las coordenadas globales en x y y de los dispositivos con respecto al eje global terrestre. Esto tiene la ventaja de permitir, a diferencia del caso de acelerómetros y codificadores ópticos, el obtener una referencia absoluta que puede ser usada para mantener la posición con respecto al planeta y sin error acumulado.

El dispositivo utilizado para esta práctica es el sensor GY-GPS6MV1, el cual consiste en un módulo GPS de alta resolución que permite la captura de coordenadas absolutas con respecto al eje terrestre haciendo uso del servicio provisto por el sistema GPSIII.

En la figura 7 se puede observar el dispositivo utilizado.

Las especificaciones de operación del dispositivo se muestran en la tabla 3.

El módulo fue conectado al Arduino haciendo uso de la librería TinyGPS, la cual provee una serie de funciones abstractas que permiten simplificar el proceso de captura y cálculo de las coordenadas del dispositivo. El código implementado se puede observar en A.

Para la conexión del dispositivo se hizo uso de los pines 4 y 5 que corresponden a RX y TX respectivamente en el Arduino.

Cuadro 3: Especificaciones de operación del sensor GPS GY-GPS6MV1.

Tipo	Módulo
Uso	Universal
Dimensiones	10.2x2.5x0.8 cm
Peso neto	0.017g
Voltaje de operación	3.3 a 5 volts

2.7. Codificador óptico

Un codificador óptico es un dispositivo de sensado que permite la captura de la posición y orientación de un elemento rotatorio por medio de la transmisión de pulsos eléctricos codificados que representan el desplazamiento del elemento. Consisten en un fotodiodo que transmite un haz de fotones a través de un disco ranurado y que es recibido por un dispositivo fotoreceptor.

El desplazamiento del haz a través de las ranuras del disco permite al dispositivo el conocer por medio de la cantidad de pulsaciones obtenidas, la presencia o no de desplazamiento en el actuador. Estos dispositivos suelen ser acoplados a dispositivos actuadores angulares como es el caso de motores.

El codificador óptico y su modo de conexión se describen en la figura **??** en la cual se muestra los pines y características principales del motor de corriente directa utilizado.

Para la interacción entre la aplicación y el codificador óptico se hizo uso de la librería **Encoder.h**, la cual permite la captura y procesamiento de información por parte de los codificadores ópticos sin requerir de la interacción directa con los elementos del microcontrolador, facilitando así la tarea de su implementación.

2.7.1. Cálculo de desplazamiento, velocidad y aceleración

El codificador óptico otorga al usuario pulsos a través de los cuales es posible inferir variables como desplazamiento, velocidad y aceleración. Para ello se requiere implementar una serie de fórmulas matemáticas considerando las condiciones de operación del dispositivo.

El cálculo del desplazamiento es logrado por medio de la fórmula:

$$Distancia = dimetroderueda * (\frac{numerodepulsos}{pulsosporrevolucin}) \tag{1}$$

A partir de esto y conociendo el tiempo de desplazamiento es posible calcular la velocidad del robot por medio de la fórmula:

$$velocidad = \frac{distancia}{tiempo} \tag{2}$$

Motorreductor con Encoder

Motorreductor con encoder

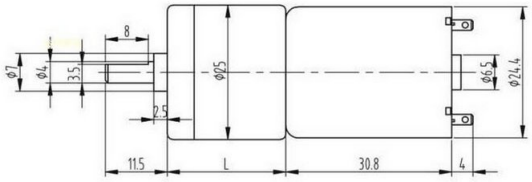
Este motorreductor cuenta con un motor de DC capaz de operar en un rango de 3V a 12V y una reducción de 75/2662 que nos multiplica el par del motor de DC en casi 36 veces y capaz de alcanzar un par de hasta 12 Kg*cm.
Su encoder incremental proporciona 334 pulsos por revolución, una cifra útil en aplicaciones donde se requiere conocer con presición la posición angular de su eje.
Su forma compacta y liviana le permite ser utilizado en robotica movil y prototipos accionados con ruedas.

Especificaciones

- Voltaje de operación de motor DC	3-12V
- Voltaje de operación de encoder	3-5V
- Pulsos por revolución	334
- Reducción	75/2662
- Par de torsión máximo	12 kg*cm
- Velocidad angular máxima	126 RPM
- Consumo @ 12V	1A

Posee 6 conexiones:

Negro:	GND Encoder
Blanco:	Salida A de encoder
Rojo	VCC Encoder
Verde:	Salida B de encoder
Naranja:	Alimentación del motor
Amrillo:	Alimentación del motor



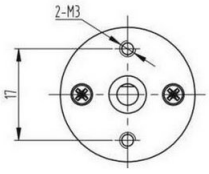


Figura 8: Principales características y modo de conexión del motor de corriente directa con codificador óptico acoplado.

3. Resultados

Se implementó un controlador PID para el seguimiento de una trayectoria lineal por parte del sistema robótico con una distancia de 1.2 metros en línea recta, buscando lograr que el sistema navegara a través de la trayectoria especificada con la menor cantidad de error posible utilizando únicamente el codificador óptico como retroalimentación.

Se logró implementar de forma exitosa el controlador con una velocidad base de 3 centímetros por segundo manteniendo un error de 15 centímetros a los 1,200 centímetros de avance como se muestra en la figura 9.

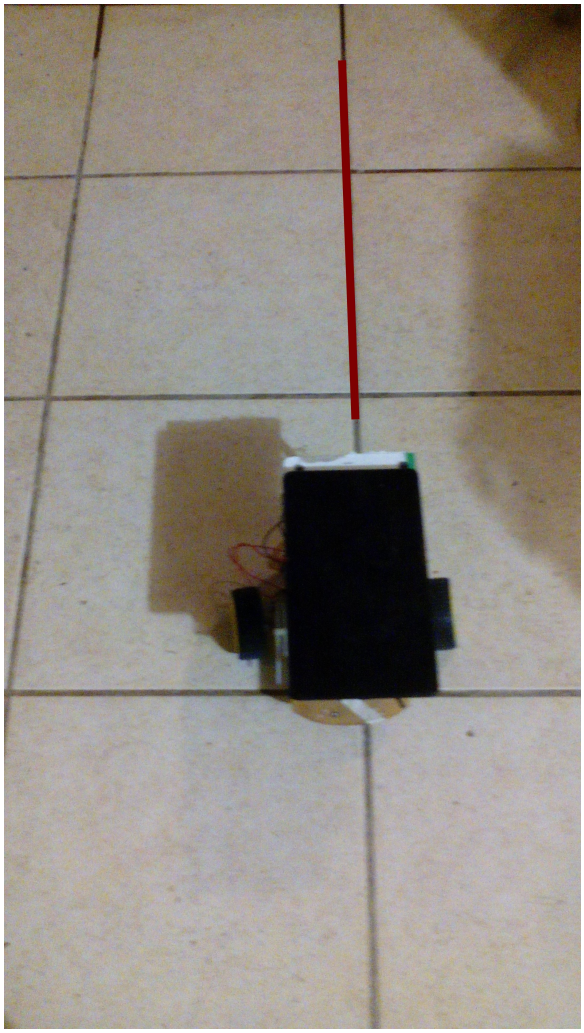


Figura 9: Trayectoria inicial a seguir por el dispositivo robótico.

El objetivo fue alcanzado exitosamente como se muestra en la figura 10

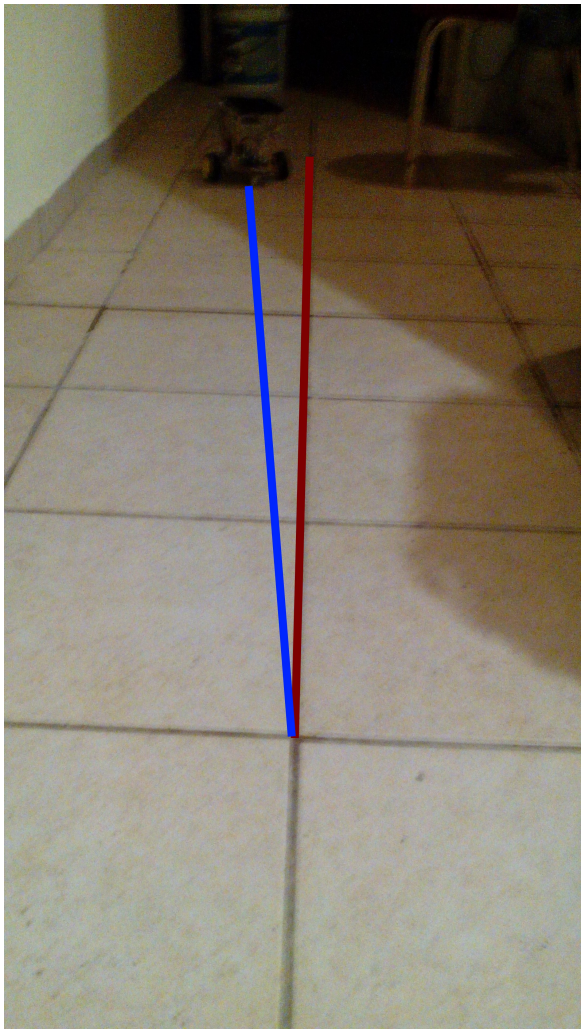


Figura 10: Objetivo alcanzado por el dispositivo robótico (azul) junto al objetivo inicial (rojo).

4. Conclusiones

Por medio de la realización de la práctica de implementación de controlador PID utilizando codificadores ópticos fue posible realizar el seguimiento de una trayectoria en línea recta utilizando el robot móvil elaborado y el cálculo de la velocidad y distancia de desplazamiento del dispositivo.

Fue posible conocer el método para conexión de los codificadores ópticos, los cuales pueden hacer uso de una referencia a tierra en algunos casos. En este caso al contar con el acoplamiento necesario de forma interna, fue posible conectar el dispositivo de manera directa a la tarjeta controladora Arduino Nano.

También se comprendió la importancia de una adecuada librería para la operación de los sensores ya que facilitan enormemente el proceso de instalación e implementación de los elementos sin requerir el interactuar de manera directa con cada uno de los elementos bit internos o realizar cálculos complejos para poder obtener la información requerida. Esto permite el crear una capa de abstracción y facilitar el proceso de lectura de la información de los sensores de forma que el robot pueda ser controlado de forma más sencilla.

Como trabajo a futuro se plantea el mejorar el controlador y realizar la misma práctica utilizando un controlador difuso, con lo cual se espera obtener resultados superiores y capacidad adaptativa mejorada.

A. Código fuente de captura de información de codificadores ópticos y seguimiento de trayectoria por control PID

```
/* CONTROLADOR DE ROBOT DIFERENCIAL.
Autor: Angel Arturo Ramirez Suarez
Asignatura: Robotica
Carrera: Maestria en Ingenieria
Universidad Politecnica de Victoria
*/
//Libreria para comunicacion serial. Solo permite una senal a la vez.
#include <SoftwareSerial.h>
//Libreria para el codificador optico.
#include <Encoder.h>
Encoder knobLeft(3, 10);
Encoder knobRight(2, 11);
long positionLeft  = 0;
long positionRight = 0;

float distanciaRecorrida = 0;
float wheelDiam = 6.5 / 100.0;
float countsPRev = 145.0;

int leftSpeed = 20;
int rightSpeed = 20;

unsigned long initTime;
float tiempo;

#define TXPIN 6
#define RXPIN 7
#define maxRange 200

float prev_error = 0, int_error = 0;
float proportional_gain = 0.5;
float integral_gain = 0.1;
float derivative_gain = 0.2;
float error = 0;
float dt = 1, diff;
int ddeseada = 20;
int P, I, D, PID;
int counter = 0;

SoftwareSerial pololu(RXPIN, TXPIN);

void setup() {
  Serial.begin (115200);
  pololu.begin(19200);
  delay(10);
  pinMode(RXPIN, INPUT);
  pinMode(TXPIN, OUTPUT);
  delay(3000);
  Serial.println("Starting");
}

void loop() {
```

```
  if(counter < 0){
    pololu.end();
  }
  else if(counter == 0){
    SetSpeed(0, true, leftSpeed);
    SetSpeed(1, true, rightSpeed);
    counter = counter + 1;
    initTime = millis();
  }
  else if(counter < 3){
    long newLeft, newRight;
    newLeft = knobLeft.read();
    newRight = knobRight.read();
    if (newLeft != positionLeft || newRight != positionRight) {
      Serial.print("Left = ");
      Serial.print(newLeft);
      Serial.print(", Right = ");
      Serial.print(newRight);
      Serial.println();
      positionLeft = newLeft;
      positionRight = newRight;

      //Correct error in left and right motors.
      error = newLeft - newRight;
      if(error < 0){
        //Right motor is moving faster.
        calcularPID(newLeft, newRight);
        leftSpeed = abs(leftSpeed+PID);
        if(leftSpeed > 50){
          leftSpeed = 30;
        }
        rightSpeed = abs(rightSpeed+PID);
        if(rightSpeed > 50){
          rightSpeed = 30;
        }
        SetSpeed(0, true, leftSpeed);
        SetSpeed(1, true, rightSpeed);
        Serial.print("PID: "); Serial.println(PID, 2);
        Serial.print("Nueva velocidad: "); Serial.println(leftSpeed);
      }
      else if(error > 0){
        //Left motor is moving faster.
        calcularPID(newLeft, newRight);
        rightSpeed = abs(rightSpeed+PID);
        if(rightSpeed > 50){
          rightSpeed = 30;
        }
        leftSpeed = abs(leftSpeed+PID);
        if(leftSpeed > 50){
          leftSpeed = 30;
        }
        SetSpeed(0, true, leftSpeed);
        SetSpeed(1, true, rightSpeed);
        Serial.print("PID: "); Serial.println(PID, 2);
        Serial.print("Nueva velocidad: "); Serial.println(rightSpeed);
      }

      //Average of both encoders to get middle counts.
      float averageCounts = (newLeft + newRight) / 2.0;
```

```
      distanciaRecorrida = (wheelDiam) * (averageCounts / countsPRev);
      Serial.print("Distancia recorrida: "); Serial.println(distanciaRecorrida, 5);

      //Calculate speeed.
      tiempo = (float)(millis() - initTime) / 1000.0;
      Serial.print("Tiempo: "); Serial.println(tiempo);
      float rspeed = distanciaRecorrida / tiempo;
      Serial.print("Velocidad: "); Serial.println(rspeed, 5);

      if(distanciaRecorrida >= 250){
        SetSpeed(0, true, 0);
        SetSpeed(1, true, 0);
        counter = -1;
      }
    }
  }
}

float calcularPID(int val1, int val2){
  error = (val1 - val2) / 300;
  if(error < 0){
        diff = (prev_error - error) / dt;
        P = proportional_gain * error;
        I = 0;
        D = derivative_gain * diff;
        prev_error = error;
        PID = P + I + D;
        return abs(PID);
    }
    else if(error > 0){
        diff = (prev_error - error) / dt;
        P = proportional_gain * error;
        I = 0;
        D = derivative_gain * diff;
        prev_error = error;
        PID = P + I + D;
        return abs(PID);
    }
}

void SetSpeed(int MotorIndex, boolean Forward, int Speed)
{
  // Validate motor index
  if(MotorIndex < 0 || MotorIndex > 2)
    return;

  // Validate speed
  if(Speed < 0)
    Speed = 0;
  else if(Speed > 127)
    Speed = 127;

  // Send the "set" command based on the motor
  // Note that we do not accelerate to the
  // speed, we just instantly set it
  unsigned char SendByte = 0;
  if(MotorIndex == 0)
    SendByte = 0xC2;
  else if(MotorIndex == 1)
```

```
      SendByte = 0xCA;
    else if(MotorIndex == 2)
      SendByte = 0xF0;

    // If we go backwards, the commands are the same
    // but minus one
    if(!Forward)
      SendByte--;

    // Send the set speed command byte
    //pololu.write(SendByte);
    pololu.write(SendByte);


    // Send the speed data byte
    pololu.write(Speed);
  }
```

Referencias

[1] J. Angelo, *Robotics: A Reference Guide to the New Technology.* Greenwood Press, 2007.

[2] K. Boyer, L. Stark, and H. Bunke, *Applications of AI, Machine Vision and Robotics.* Series in machine perception and artificial intelligence, World Scientific, 1995.

[3] B. Drury, C. T. (Firm), and I. of Electrical Engineers, *Control Techniques Drives and Controls Handbook.* IEE Power Series, Institution of Engineering and Technology, 2001.

[4] E. Kaplan and C. Hegarty, *Understanding GPS: Principles and Applications, Second Edition.* Artech House mobile communications series, Artech House, 2005.