

# Fase Final: SISTEMA ANTIDERRAPE PARA SILLA DE RUEDAS\*

José Alejandro, Rodas Alvarez, 201801526,<sup>1, \*\*</sup> Josue Eduardo, Nimatuj Piedrasanta, 201632173,<sup>1, \*\*\*</sup> and Jerry Javier, Pernilla Pérez, 201807342<sup>1, \*\*\*\*</sup>

<sup>1</sup>Facultad de Ingeniería, Laboratorios de Electrónica, Universidad de San Carlos, Edificio T1, Ciudad Universitaria, Zona 12, Guatemala.

En la Fase final se implementaron las programaciones de las practicas 1, 2 y 3 del laboratorio para el control de las revoluciones de las ruedas mediante el control de la velocidad angular por medio de los encoder de cuadratura, así como la implementación de la detección de la distancia de un objeto al prototipo de silla, todo esto se montó en una maqueta impresa en 3D.

## I. OBJETIVOS

### A. Generales

- Elaborar un modelo de una silla de ruedas que sea totalmente capaz de ser inmune a perturbaciones que la hagan derrapar.

### B. Específicos

- \* Desarrollar un código en el IDE de Arduino que sea capaz de controlar la velocidad de las ruedas acorde a un sensor ultrasónico.
- \* Implementar un control PID (Control Proporcional, Integral y Derivativo) que permita corregir las variaciones que se presenten en una rueda o en el circuito entero de la silla de ruedas.
- \* Elaborar un código en el IDE de Arduino que permita la comunicación inalámbrica entre el microcontrolador y el ordenador que permitirá la visualización de las variables medibles.

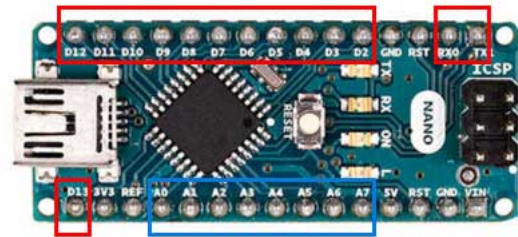


Figura 1: Arduino nano

## II. MARCO TEÓRICO

### A. Microcontrolador

Arduino Nano es una placa de desarrollo basada en el microcontrolador ATmega328P (el mismo del Arduino UNO). Provee la ventaja de su tamaño, Arduino Nano cuenta con un total de 14 pines digitales de entrada/salida, de los cuales 6 pueden ser utilizados como salidas analógicas (utilizando señales PWM). Además de contener diferentes pines para distintos protocolos de comunicación.

### B. Actuadores

Los actuadores eléctricos usan una fuente de energía externa, como una batería, para producir movimiento.

#### ■ Driver Controlador de Motores L298N

El puente H doble L298N permite controlar la dirección y velocidad de dos motores.

Este módulo trabaja desde 3V hasta 35V, y con una intensidad de hasta 2A, consume 3v así que los motores reciben 3 voltios menos de el voltaje ingresado, incluye un regulador que nos puede dar una salida de 5v, si lo alimentamos entre 5 a 12v.

---

\* Laboratorio de Sistemas de Control

\*\* e-mail: 3006922500101@ingenieria.usac.edu.gt

\*\*\* e-mail: 3136032510901@ingenieria.usac.edu.gt

\*\*\*\* e-mail: 2997291250101@ingenieria.usac.edu.gt

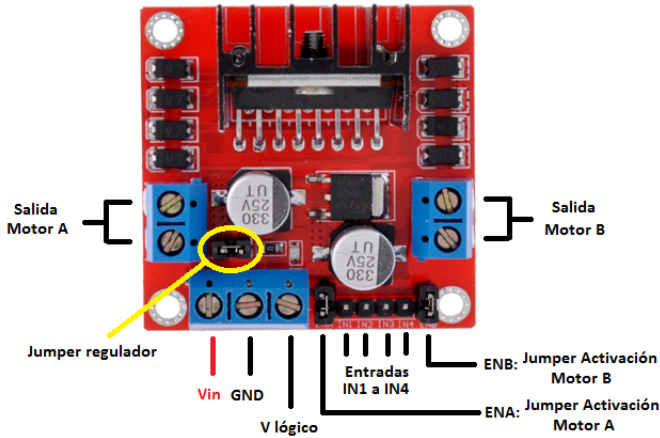


Figura 2: Pines Controlador de Motores L298N

- \* Los pines IN1 y IN2: Sirven para controlar el sentido del motor A.
- \* Los pines IN3 y IN4: Sirven para controlar el sentido del motor B.

Entradas			Función
ENA/ENB	IN1\IN3	IN2\IN4	
H	H	L	Adelante
	L	H	Atrás
	IN1\IN3 = IN2\IN4		Parada rápida del motor
L	X	X	Motor Parado

Figura 3: Tabla funcionamiento Controlador de Motores L298N

- **Motorreductor**  
Máquina muy compacta que combina un reductor de velocidad y un motor. Estos van unidos en una sola pieza y se usa para reducir la velocidad de un equipo de forma automática.



Figura 4: Motorreductor 6v

C. Sensores

- **Encoder de Cuadratura**  
Un Encoder de Cuadratura es un tipo de encoder rotativo incremental que tiene la capacidad de indicar tanto la posición, la velocidad y la dirección de movimiento del eje del motor.

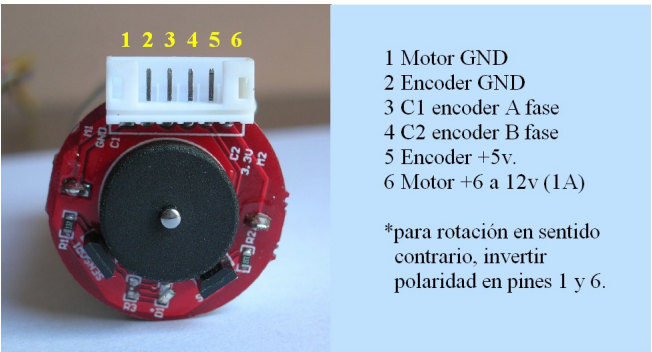


Figura 5: Encoder de Cuadratura

Este encoder tiene dos sensores de efecto Hall que generan dos señales de pulsos digitales desfasada en 90º eléctricos o en cuadratura gracias a un disco magnético giratorio, montado en el eje trasero del motor. A estas señales de salida, se les llama comúnmente A y B. Estos dos sensores se van activando y desactivando en una secuencia que nos permite saber la dirección y el número de desplazamientos que han ocurrido en el encoder.

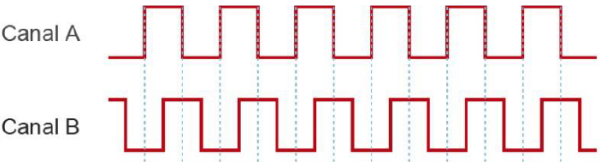
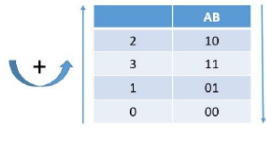


Figura 6: Pulsos digitales desfasada en 90º

### \*Sentido giro

Para la determinar el sentido de giro (signo), se toman como referencia el desfase que existe entre las señales AB. El hecho de trabajar con dos canales, nos lleva a tener 4 estados, estos mismos son la combinación de A y B.



	AB
2	10
3	11
1	01
0	00

	Anterior	0	1	2	3
Actual	AB	00	01	10	11
0	00	0	-1	+1	E
1	01	+1	0	E	-1
2	10	-1	E	0	+1
3	11	E	+1	-1	0

Figura 7: Tabla para determinar el sentido de giro

Como se puede observar en la figura anterior, la Tabla de la derecha muestra una matriz de incrementos donde podemos comparar el estado anterior con el actual y saber cómo fue el movimiento. Si el estado anterior fue 01 y el estado actual del encoder es 11 podemos saber que el motor está girando en sentido anti-horario (+) ya que tenemos un incremento que va de 1 a 3 según la tabla de la izquierda. Esta matriz es muy útil para saber en pocos pasos que ocurre con el encoder. Y en total tenemos 4 incrementos(+1).

**Respecto a la precisión, tenemos más de una opción.**

- Precisión simple, registrando un único flanco (subida o bajada) en un único canal.
- Precisión doble, registrando ambos flancos en un único canal.
- Precisión cuádruple, registrando ambos flancos en ambos canales.

### \*Precisión cuádruple

Para aplicaciones de robótica se utiliza la precisión cuádruple, esta precisión es la que permite detectar el sentido de giro del eje del motor. Por lo tanto, la resolución del encoder para una precisión cuádruple (R) se calcula de manera diferente al Holzer resolution.

$$R = mH * s * r \quad (1)$$

Donde:

R: Es la resolución del encoder para una precisión cuádruple.

mH: Es el número de cuentas por revolución del eje del motor (Motor Holzer). En nuestro caso 11.

s : Es el número de estados generado por los canales AB. En nuestro caso 4.

r : Es la relación de reducción de la caja reductora (ratio).

### ■ Sensor Ultrasónico

Los sensores ultrasónicos miden la distancia mediante el uso de ondas ultrasónicas. El cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto. Los sensores ultrasónicos miden la distancia al objeto contando el tiempo entre la emisión y la recepción.

Un sensor óptico tiene un transmisor y receptor, mientras que un sensor ultrasónico utiliza un elemento ultrasónico único, tanto para la emisión como la recepción. En un sensor ultrasónico de modelo reflectivo, un solo oscilador emite y recibe las ondas ultrasónicas, alternativamente. Esto permite la miniaturización del cabezal del sensor.



Figura 8: Sensor HC SR04

Fuente: Elaboración propia.

Características eléctricas:

- Voltaje de trabajo: 5V.
- Corriente de trabajo: 15mA.
- Frecuencia de trabajo: 40KHz
- Rango de funcionamiento: 2 a 500 cm
- Ángulo de detección: 15 a 20 grados.

### D. Materiales

\* Arduino Nano

\* Driver Controlador de Motores L298N

\* Motorreductor de 120 RPM con encoder a 6V 25GA370-100

\* Sensor Ultrasonico HC -SR04

### III. COSTE DEL PROYECTO

En la siguiente tabla se da la lista de materiales con su respectivo precio y el monto total para la realizacion del proyecto:

No	Producto	Precio
2	Motorreductor(25GA370-100)	Q376.00
1	Sensor ultrasónico HC-SR04	Q29.00
2	Mini-protoboard de 170 punto	Q22.00
1	Buzzer magnético activo	Q5.00
1	Módulo puente H L298N	Q45.00
2	Alambre para protoboard	Q5.50
1	MÓDULO BLUETOOTH HC-05	Q70.00
2	RUEDA PLASTICA LLANTA	Q38.00
1	Impresion 3D Carro	Q30.00
1	ARDUINO NANO V3.0 FT232RL	Q68.00
	Total	Q688.5

Cuadro I: Presupuesto

### IV. RESULTADOS

#### A. Diseño de Estructura

El diseño de la estructura fue realizado en el software Tinkercad, que permite crear modelos 3D, para su posterior Impresión 3D

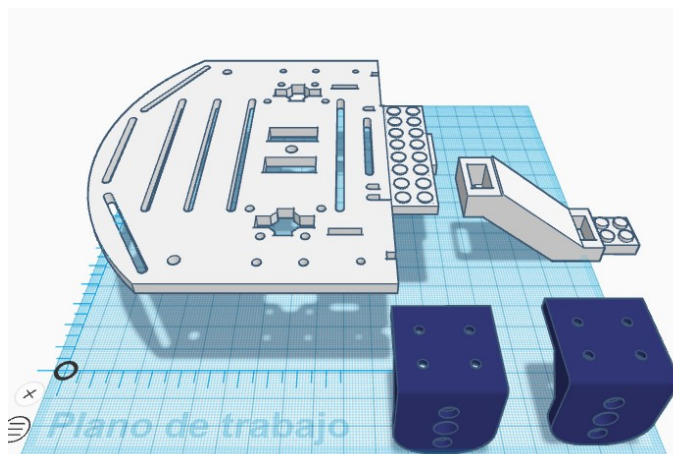


Figura 9: Diseño de estructura para la impresión 3D  
Fuente: Elaboración propia 2021

#### B. Calculo Resolución del encoder

La resolución de un encoder es una medida en pulsos por revolución (PPR), es decir el número de conteo (cuentas) que genera el encoder en cada vuelta. Este dato es proporcionado por el fabricante, pero en nuestro caso el modelo de motor 25GA370-100 no posee una descripción específica para la relación de reducción de la caja reductora (ratio). Numero de cuentas por revolución.

Para determinar los PPR se plante lo siguiente:

En base a la figura (tabla para determinar el sentido de giro) anterior se debe crear un código que permita comparar el estado anterior y el estado actual de los pines C1 (canal A) y C2 (canal B) del encoder.

En primer lugar, se debe detectar los flancos de subida y bajada de ambos canales AB. Para esta tarea lo más recomendable es usar interrupciones. Entonces, definimos los pines y las variables que se van a utilizar, configuramos los pines como entradas y activamos las interrupciones para detectar ambos flancos.

```
const int C1 = 3; // Entrada de la señal A del encoder.
const int C2 = 2; // Entrada de la señal B del encoder.
//Las variables de tipo interrupcion se recomienda que sean de tipo volatile
volatile int n = 0;
volatile byte ant = 0; // Variable para el estado anterior
volatile byte act = 0; // Variable estado actual, segun la tabla vista
void setup()
{
  pinMode(C1, INPUT);
  pinMode(C2, INPUT);
  attachInterrupt(digitalPinToInterrupt(C1), encoder, CHANGE);
  attachInterrupt(digitalPinToInterrupt(C2), encoder, CHANGE);
}
```

En la función encoder vamos a comparar el estado anterior y actual e incrementar o decrementar un contador según la tabla.

Para esta tarea, vamos a leer un puerto completo que dependiendo en donde se encuentre conectado los pines puede variar.

Los chips utilizados en la placa Arduino (ATmega8 y ATmega168) tienen tres puertos:

- B (pines digitales de 8 a 13) PINB
- C (pines de entrada analógica) PINC
- D (pines digitales de 0 a 7) PIND

En este caso estamos utilizando los pines 2 y 3 es decir el puerto D, por lo tanto, para leer el puerto completo usaremos PIND.

Adicionalmente, debemos filtrar solo los pines de interés este caso el 2 y 3. Entonces considerando que los puertos de Arduino (ATmega8 y ATmega168) son de 8 bits se puede aplicar una operación and entre el valor de



PIND y 00001100 (12 DECIMAL).

0	0	0	0	1	1	0	0
7	6	5	4	3	2	1	0

Entonces en base a tabla debemos hacer la comparación sin olvidar que los valores de la tabla también cambian en base a la conexión de los pines, es decir, si hay un flanco en el canal A (pin 3) y nada en el canal B (pin 2) el valor binario es 10 (2 DECIMAL) se debe cambiar a 00001000 (8 DECIMAL) y así en todas las combinaciones 00, 01 y 11.

0	0	0	0	1	0	0	0
7	6	5	4	3	2	1	0

Código para calcular los incrementos y decrementos del motor:

```
// Encoder precisión cuádruple.
void encoder(void)
{
  ant=act;
  act=PIND & 12;
  if (ant==0 && act== 4) n++;
  if (ant==4 && act==12) n++;
  if (ant==8 && act== 0) n++;
  if (ant==12 && act== 8) n++;
  if (ant==0 && act==8) n--;
  if (ant==4 && act==0) n--;
  if (ant==8 && act==12) n--;
  if (ant==12 && act==4) n--;
}
```

Entonces, si cargamos el siguiente sketch en Arduino al dar una vuelta completa al eje del motor se debe mostrar en el monitor serial el valor de la **resolución del encoder para una precisión cuádruple R**.



Figura 10: Calibración de encoder.

Fuente: Elaboración propia 2021

Realizando la comprobación de manera práctica, por medio del monitor serial de Arduino podemos ver cuántos pulsos muestra por una revolución. Que es la Resolución del Encoder para una precisión cuádruple **(R)= 1890 ±10. Dando una relación de reducción de la caja reductora de aproximadamente 43.**

Usando la función `millis()` que nos cuenta el tiempo desde que inició a ejecutarse el programa, calculamos el tiempo de las muestras cada 100ms. Cada tiempo de muestreo debemos resetear el valor del contador para obtener una nueva lectura en el siguiente instante de muestreo. El código es el siguiente.

```
if (millis() - lastTime >= sampleTime)
{
  // Se actualiza cada tiempo de muestreo
  // N = (n*60.0*1000.0)/((millis()-lastTime)*R); // Velocidad en RPM
  lastTime = millis(); // Almacenamos el tiempo actual.
  n = 0; // Inicializamos los pulsos.
}
```

### C. Programación de la velocidad Angular en RPM

Una vez calculado las revoluciones del encoder para una precisión cuádruple se puede medir las **revoluciones por minuto**, para ello vamos a utilizar la siguiente fórmula.

$$N = \frac{n * 60}{t * R} \quad (2)$$

Donde:

N: Es la velocidad de rotación en Revoluciones Por Minuto (RPM).

n: Es el número de conteo generado en determinado tiempo t

t : Es el tiempo de generación de los pulsos en segundos

(s).

R : Es la resolución del encoder para una precisión cuádruple. En nuestro caso 1890 cuentas por revolución.

Ya que el tiempo  $t$  es en segundos, sin embargo, en el microcontroladores se trabaja en el orden de los microsegundos o milisegundos. Para trabajar en milisegundos se debe hacer la siguiente conversión  $1s = 1000ms$  y aplicarlo a la fórmula de la siguiente manera

```
void computeRpm(void)
{
  N = (n*60*1000.0)/((millis()-lastTime)*R);
  lastTime = millis();
  n =0;
}
```

Resultado de la velocidad angular RPM en el monitor serial

```
COM7
11:03:07.518 -> RPM: -149.70
11:03:07.657 -> RPM: -151.52
11:03:07.750 -> RPM: -151.52
11:03:07.843 -> RPM: -149.70
11:03:07.937 -> RPM: -150.91
11:03:08.030 -> RPM: -150.00
11:03:08.122 -> RPM: -151.52
11:03:08.215 -> RPM: -151.21
11:03:08.355 -> RPM: -149.39
11:03:08.449 -> RPM: -151.21
11:03:08.542 -> RPM: -150.91
11:03:08.637 -> RPM: -149.39
```

Obteniendo una media de 150.2 RPM.

#### D. Recepción de la velocidad angular

A 255, 255 muestran una velocidad angular de la llanta izquierda de 16.5 y de la llanta derecha de 16.7 aproximadamente. Con el propósito de saber cual es nuestra velocidad máxima para ambos motores.

Cv (25-191)	wL	wR	
25	0.0	0.0	Zona Muerta
30	1.0	1.8	
35	2.0	3.0	
45	4.1	5.2	50 %
55	6.0	7.1	
65	7.7	8.7	75 %
75	9.1	9.9	
85	10.3	10.8	100 %
95	11.4	11.7	
105	12.0	12.5	
115	12.7	13.0	

Cuadro II: Calibración de velocidad

```
COM8
01:45:26.305 -> 16.528 16.373
01:45:26.305 -> 16.528 16.373
01:45:26.305 -> 16.843 16.716
01:45:26.305 -> 16.843 16.716
01:45:26.350 -> 16.843 16.716
01:45:26.350 -> 16.843 16.716
01:45:26.391 -> 16.843 16.716
01:45:26.391 -> 16.843 16.716
```

Figura 11: Calibración de encoder

Fuente: Elaboración propia 2021

Realizamos una tabla que contiene la velocidad y la relación del temporizado PWM del arduino nano con el fin de establece una medida de velocidad angular máxima.

Zona Muerta:

Mínima señal de entrada necesaria para que el sistema responda.

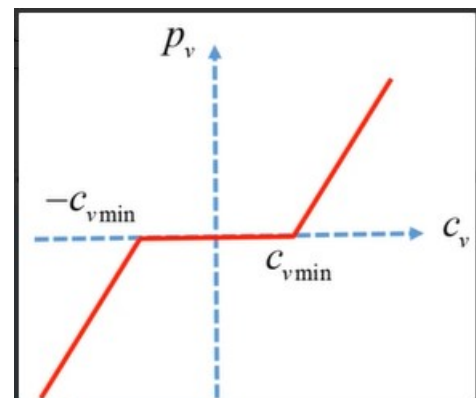


Figura 12: Zona muerta

Fuente: Elaboración propia 2021

## E. Código Arduino

Necesitamos la librería PinChangeInterrupt.h ya que estamos utilizando dos motores y necesitamos mas pines para interrupciones.

```
#include "PinChangeInterrupt.h"
```

Asignación de pines tanto del encoder como del modulo de potencia para el control de los motores.

```
//////////MOTOR DERECHO///
```

```
const byte    C1_1 = 3;
const byte    C2_1 = 2;
const byte    in1  = 6;
const byte    in2  = 7;
const byte    enA  = 10;
```

```
volatile long contador_1 = 0;
volatile byte ant_1      = 0;
volatile byte act_1      = 0;
```

Distribución de pines para el motor Izquierdo

```
//////////MOTOR IZQUIERDO//////////
```

```
const byte    C1_2 = 5;           // Entrada de la señal A del encoder.
const byte    C2_2 = 4;           // Entrada de la señal B del encoder.
const byte    in3  = 8;
const byte    in4  = 9;
const byte    enB  = 11;
```

```
volatile long contador_2 = 0;
volatile byte ant_2      = 0;
volatile byte act_2      = 0;
```

Tenemos nuestro tiempo de muestreo, que nos será de utilidad para el control PID de los motores

```
unsigned long lastTime = 0,    sampleTime = 0;
```

Los siguientes valores como alfa = 0.7 es una constante que va a permitir realizar el filtro y el Pulsos Por Revolución que ya se calculo 1890. La constValues es determinada por la siguiente ecuación:

$$\omega = \frac{2 * \pi * n * 1000}{t * R} \quad (3)$$

En donde nuestra variables controladas son n y t. Se calcula para ambos motores.

```
unsigned int ppr = 1890; // Número de muescas que tiene el disco del encoder.
const double alfa = 0.95;
double constValue = 3.1733 ; // (1000*2*pi)/R ---> R = 1980 Resolución encoder cuadruple
//const double valueConst1 = 0.10631, wheelDistance = 0.195;
const double ConstValue2 = 3.15;
```

Uso de la librería para configurar como interrupciones los datos recibidos del motor derecho.

Ya que hay pocos pines para interrupciones en el arduino nano se utiliza la función de la librería PinChangeInterrupt.h:

```
attachInterrupt(digitalPinToInterrupt(C1_1), encoder_1, CHANGE);
attachInterrupt(digitalPinToInterrupt(C2_1), encoder_1, CHANGE);

attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(C1_2), encoder_2, CHANGE);
attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(C2_2), encoder_2, CHANGE);
```

## F. Código Arduino control de distancia Sensor Ultrasonico

Control de velocidad angular del Motor Izquierdo por medio del sensor ultrasónico y encendido de led-buffer.

```
if (distancia <= 100 && distancia >= 51){
    w1Ref = 65;//75%
    //w2Ref = 143;
    digitalWrite(LED, HIGH);    // enciende LED
}
else if(distancia <= 50 && distancia >=26)
{
    w1Ref = 45;//50%
    //w2Ref = 96;
    digitalWrite(BUZ,HIGH);//
}
else if (distancia <= 25){
    w1Ref = 0;
    digitalWrite(BUZ,LOW);
    digitalWrite(LED, LOW);
}
else {
    w1Ref = 85;//100%
    digitalWrite(LED, LOW);
    digitalWrite(BUZ,LOW);// apagar
}
```

## G. Código de Control PID

En la siguiente sección tenemos una función que se ejecuta cada 100 ms. La función ComputePID(); nos permite calcular la velocidad de la llanta izquierda y control la velocidad de la llanta derecha.

```
ComputePID(); //CALCULOS
```

Dentro de la función Compute(); tenemos todas las fórmulas con las que vamos a poder calcular la velocidad angular de ambos motores basado en la fórmula anterior.

$$\omega = \frac{2 * \pi * n * 1000}{t * R} \quad (4)$$

El cálculo del PID debe estar dentro de una función que tome muestra cada cierto tiempo como la función millis(). La primera parte de la función computePID() debería ser la de determinar el tiempo transcurrido.

El uso de la función `millis()` nos permite tomar muestra cada 100ms sin la necesidad de parar el código, realizando la tarea de toma de datos de los motores de forma semi paralela al código de programación.

```
if (millis() - lastTime >= sampleTime) {
  noInterrupts(); // Desconectamos la interrupción para que no actué en esta parte del programa.
  wL = constValue*contador_1/(millis()-lastTime);
  wR = ConstValue2*contador_2/(millis()-lastTime);
  lastTime = millis();
  Setpoint1 = wL;
  contador_1 = 0;
  contador_2 = 0;
  interrupts(); // Reiniciamos la interrupción
```

## H. Control PID

Para implementar un controlador PID en un código o un programa de Arduino, se deben conocer cinco parámetros: constantes proporcionales, integrales, derivadas, valor de entrada y valor de Referencia.

Primero aplicamos un filtro a nuestra señal de referencia que es la velocidad del motor izquierdo, el filtro se denomina EMA. La media móvil exponencial (EMA), también conocido como promedio móvil ponderado exponencialmente (EWMA), es un filtro pasa-bajo de primer orden cuyo objetivo es atenuar el ruido presente en las señales.

$$S(t) = \begin{cases} Y(0) & t = 0 \\ \alpha Y(t) + (1-\alpha)S(t-1) & t > 0 \end{cases}$$

- El coeficiente  $\alpha$  es un factor de suavizado entre 0 y 1.
- $Y(t)$  es el valor de la señal a filtrar en un período de tiempo  $t$ .
- $S(t)$  es el valor de la EMA en cualquier período de tiempo  $t$ .
- $S(t-1)$  es el valor de la EMA en el período de tiempo  $t-1$  (pasado).

A continuación determinamos el Error, en donde la integral del error corresponde al error acumulativo en el tiempo y la derivada del error hace referencia a la tasa de cambio del error.

Finalmente, la salida calculada que es la suma de todos los errores.

```
Input1 = alfa*((wR)/2.0)+(1.0-alfa)*lastInput1;
lastInput1 = Input1; // Se guarda la posición para convertirla en pasado.
//Control Proporcional
error1 = (Setpoint1 - Input1) * kpW;
//Control Integral
ITerm1 += (error1 * kiW);
//Control Derivativo
rateError = (error1 - lastError)*kdW;
lastError= error1;
// Acota el error integral para eliminar el "efecto windup".
if (ITerm1 > outMax) ITerm1 = outMax; else if (ITerm1 < outMin) ITerm1 = outMin;
vW = error1 + ITerm1 + rateError; // Suma todos los errores, es la salida del control PID.
vR = vW;
if (vR > outMax) vR = outMax; else if (vR < outMin) vR = outMin; // Acota la salida para que el
```

Aquí, el  $K_p$ ,  $K_i$  y  $K_d$  son las constantes predeterminadas, y que se calcularon a base de prueba y Error, basado

en las Reglas de sintonía Fina. Entre las observaciones se destaca lo siguiente:

1. Aumentando la ganancia proporcional disminuye la estabilidad.
2. El error decae a mayor velocidad si se disminuye el tiempo de integración.
3. Disminuyendo el tiempo de integración disminuye la estabilidad.
4. Aumentando el tiempo derivativo mejora la estabilidad.

```
sampleTime = 100;
kpW = 16; //15; 16;
kiW = 0.00015; // 1
kdW = 2;//2;
Setpoint1 = 0.0;
```

Mandamos los valores de 0 a 255 de la velocidad angular de referencia para el motor izquierdo y la velocidad modificada por el control PID al motor Derecho.

```
// Guardar Contador de 0 a 255 en variables cvR y cvL
cvR = wLRef;
cvL = vR;

//Movimiento de motores Horaria y antihorario
if (cvR > 0) mov_horario(in2,in1,enA,cvR); else mov_anti_horario(in2,in1,enA,abs(cvR));
if (cvL > 0) mov_anti_horario(in3,in4,enB,cvL); else mov_horario(in3,in4,enB,abs(cvL));

void mov_horario(int pin1, int pin2,int analogPin, int pwm)
{
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, HIGH);
  analogWrite(analogPin,pwm);
}

void mov_anti_horario(int pin1, int pin2,int analogPin, int pwm)
{
  digitalWrite(pin1, HIGH);
  digitalWrite(pin2, LOW);
  analogWrite(analogPin,pwm);
}
```

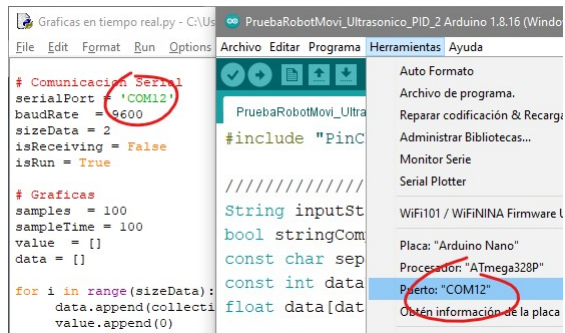
## I. Envío de Datos Por Bluetooth

Se utilizo el modulo Bluetooth HC-05 para el envio de datos hacia una computadora por medio de la comunicaciones serial.

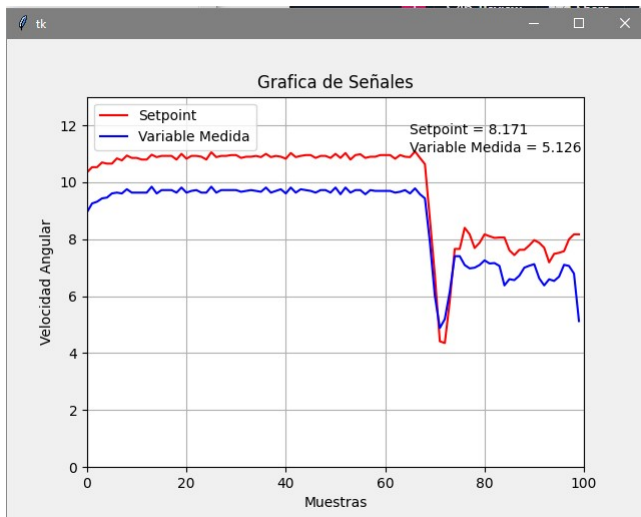
Primero debemos conectar nuestro modulo HC-05 configurado como esclavo y saber la contraseña de modulo. Segundo vamos a Configuraciones>Bluetooth y otros dispositivos>Agregar Bluetooth y otros dispositivos>Seleccionamos nuestro modulo bluetooth.

Tercero vamos a Configuraciones>Mas opciones de Bluetooth>Puertos COM Y elegimos el modulo bluetooth con la dirección saliente. Acontinuacion se debe configurar le IDE de arduino y elCodigo de python para que se conecten al COM asignado por la computadora que esta conectado al modulo bluetooth.





Se utilizo la libreria Pyserial de Python para recibir los datos por medio de la comunicacion serial enviado por el mudulo de Bluetooth de arduino. A continuacion se presenta la velocidad angular de ambos motores, la velocidad angular del motor izquierdo es el setpoint y la velocidad angular del motor derecho es nuestra variable medida:



#### J. Sistema de control de lazo cerrado

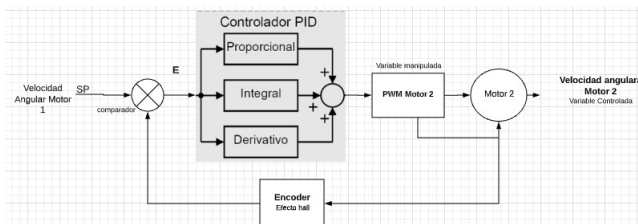


Figura 13: Sistema de control  
Fuente: Elaboración propia 2021

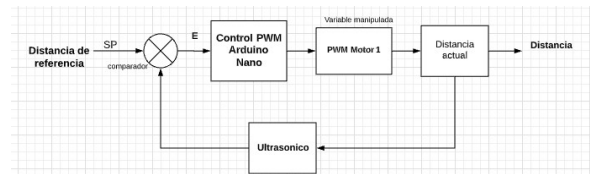


Figura 14: Sistema de control  
Fuente: Elaboración propia 2021

#### K. Circuito de encoder

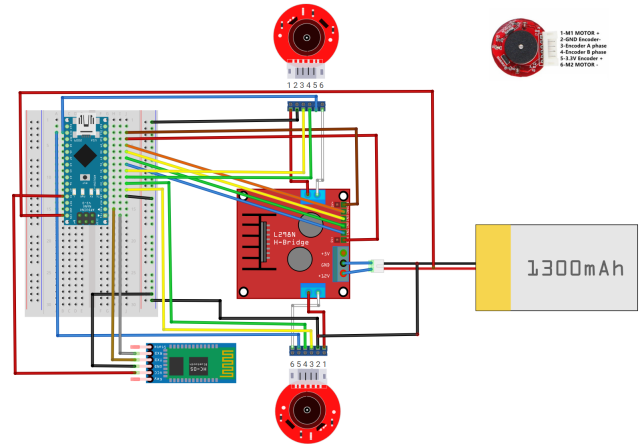


Figura 15: Circuito de control  
Fuente: Elaboración propia 2021

#### L. Estructura y componentes

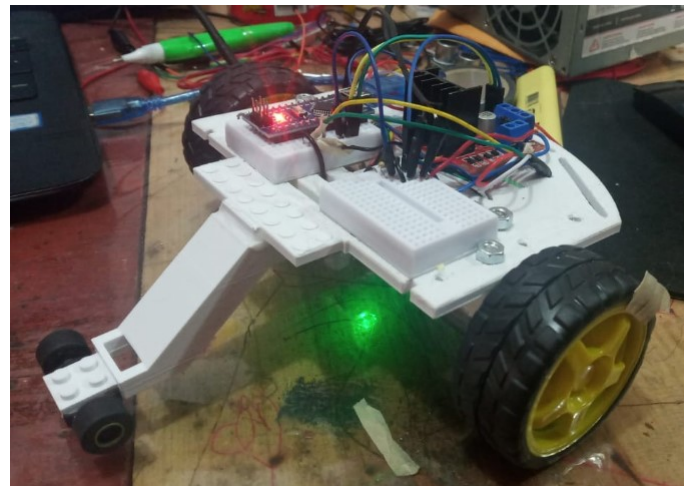


Figura 16: Maqueta montada  
Fuente: Elaboración propia 2021

Motores Con encoders de cuadratura.

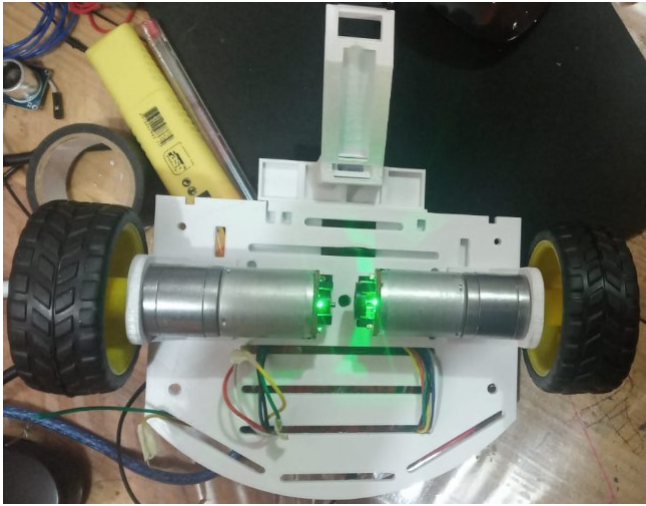


Figura 17: Moto reductor con encoder  
Fuente: Elaboración propia 2021

### M. Justificación del control usado

Se decidió usar un control PID debido a que el control P reacciona directamente proporcional a la velocidad angular de la rueda opuesta para nivelar la velocidad de la rueda sin freno externo. Debido a que aun existe aun un error de estado estable, para que no exista un deslizamiento u desfase de una rueda con otra se implementó el control I para que la velocidad y la fase de la rueda sean exactamente los mismos en ambas ruedas al momento de variar la velocidad de cualquiera de las ruedas. Por ultimo se implemento el control D para corregir el sobrepaso y efecto de "latigazo" debido la reacción rápida causada por el control PI.

Obteniendo así una reacción rápida, una estabilización rápida, y un error de estado estable inexistente

## V. DISCUSIÓN DE RESULTADOS

En la fase final del proyecto de sistemas de control, la silla de ruedas muestra el funcionamiento completo del modelo, con hardware implementado y del código desarrollado para el funcionamiento de este proyecto. Como se logra observar en las secciones anteriores, se presenta la maqueta física en donde se instaló las ruedas y los motores acoplados a sus respectivos encoders, que permiten el control de la velocidad de las ruedas. Adicional, se implemento el posicionamiento del microcontrolador, puente H y sensor ultrasónico que controlan las variables de distan-

cia, corriente y captación de datos respectivamente. En el momento del funcionamiento de la maqueta, el microcontrolador se encarga de transmitir todas las variables medidas, tales como la distancia y la velocidad de los motores, a la computadora, ya que este último elemento permite la visualización de cada variable en tiempo real por medio de una comunicación serial de un emisor bluetooth instalado en el microcontrolador, y usando un graficador en python para su posterior graficación.

## VI. CONCLUSIONES

1. El control de la velocidad de ambas ruedas puede determinarse en función de la distancia medible por el sensor ultrasónico, modificando así el ciclo de trabajo de forma directa a los rangos de distancia.
2. La corrección de las perturbaciones que se presente en una rueda de la silla de ruedas, se puede corregir mediante un Encoder que permite realizar un control PID en la programación de Arduino.
3. La comunicación entre el microcontrolador y el ordenador se obtiene mediante un módulo Wifi que garantiza la obtención de las variables medibles del sensor ultrasónico y encoders.

## VII. CODIGOS UTILIZADOS

- Codigo Encode Cuadratura Posicion  
[https://drive.google.com/drive/folders/1XdGVsh3\\_eQdnA0mrkkMqEPQ9kn4qT1Y0?usp=sharing](https://drive.google.com/drive/folders/1XdGVsh3_eQdnA0mrkkMqEPQ9kn4qT1Y0?usp=sharing)
- Codigo Encode Cuadratura RPM  
[https://drive.google.com/drive/folders/1FJxDs7iIhg9KK\\_WBZ8JVBS3UydPvY1gy?usp=sharing](https://drive.google.com/drive/folders/1FJxDs7iIhg9KK_WBZ8JVBS3UydPvY1gy?usp=sharing)
- Codigo completo  
[https://drive.google.com/drive/folders/1GF4H\\_gts-HRdMdTHBb10agS06jGwaKrB?usp=sharing](https://drive.google.com/drive/folders/1GF4H_gts-HRdMdTHBb10agS06jGwaKrB?usp=sharing)
- Video demostrativo  
[https://drive.google.com/file/d/1-GUdaCjwGEwhTRjvz\\_MQEm5RULvPuNU-/view?usp=sharing](https://drive.google.com/file/d/1-GUdaCjwGEwhTRjvz_MQEm5RULvPuNU-/view?usp=sharing)

- 
- [1] Grossman, S. (Segunda edición). (1987). *Álgebra lineal*. México: Grupo Editorial Iberoamericana.  
[2] Reckdahl, K. (Versión [3.0.1]). (2006). *Using Imported*

- Graphics in LATEX and pdfLATEX*.  
[3] Nahvi, M., & Edminister, J. (Cuarta edición). (2003). *Schaum's outline of Theory and problems of electric cir-*

*cuits*. United States of America: McGraw-Hill.

- [4] Haley, S.(Feb. 1983).*The Thévenin Circuit Theorem and Its Generalization to Linear Algebraic Systems*. Education, IEEE Transactions on, vol.26, no.1, pp.34-36.
- [5] Anónimo. *I-V Characteristic Curves* [En línea][25 de

octubre de 2012]. Disponible en:

<http://www.electronics-tutorials.ws/blog/i-v-characteristic-curves.html>