# Reporte de prácticas de laboratorio

Facultad de Instrumentación Electrónica

Ingeniería en Instrumentación Electrónica

EE: Tópicos Selectos de IIE: Laboratorio de Automatización

Profesor: Mtro. Daniel Sandria Flores

Fecha: 17 de junio de 2024

## Estudiante


| | |
|---|---|
| Víctor Joshua Hernández Ramón | zS20015585@estudiantes.uv.mx |
| Rodrigo Granillo Murrieta | zS20015589@estudiantes.uv.mx |
| Jesús Emanuel García Utrera | zS20015653@estudiantes.uv.mx |
| Del Ángel Alvino Henry | zS20020851@estudiantes.uv.mx |


## Nombre de la práctica

Elaboración del sistema de control de velocidad para el motor DC con Arduino.

## Introducción

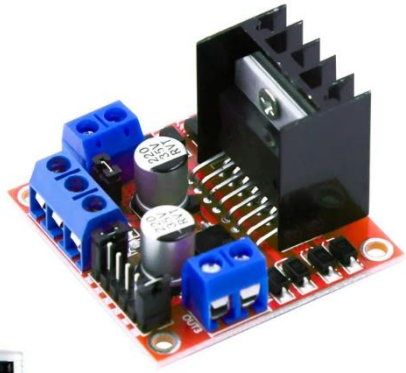
En esta práctica se procederá a la implementación de un sistema de control PID utilizando un microcontrolador Arduino, específicamente el modelo Uno ya que se contaba con este, con el objetivo de gestionar la velocidad de un motor DC realizando un PID digital adaptando las etapas correspondientes.

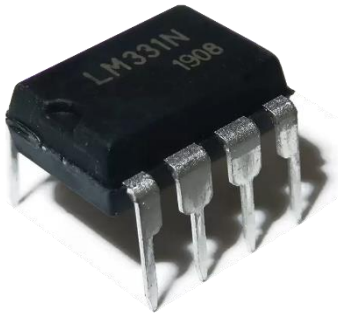
Además, entender que en la automatización y el control preciso de sistemas dinámicos es un componente esencial en la ingeniería moderna.

## Materiales utilizados o componentes

**Conversor frecuencia voltaje**

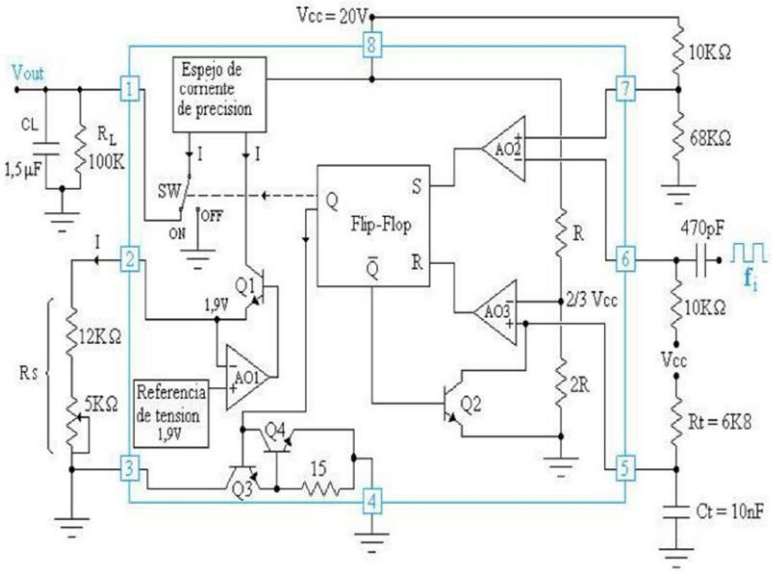
- 1 Arduino Uno
- 1 módulo HC-020k
- 1 Lm331
- 1 potenciómetro de 5 kΩ
- 2 resistencias de 10 kΩ
- 1 resistencia de 100 kΩ
- 1 resistencia de 12 kΩ
- 1 resistencia de 68 kΩ
- 1 resistencia de 6.8 kΩ
- 1 capacitor de 10 nF
- 1 capacitor 470 pF
- 1 capacitor 1.5 uF



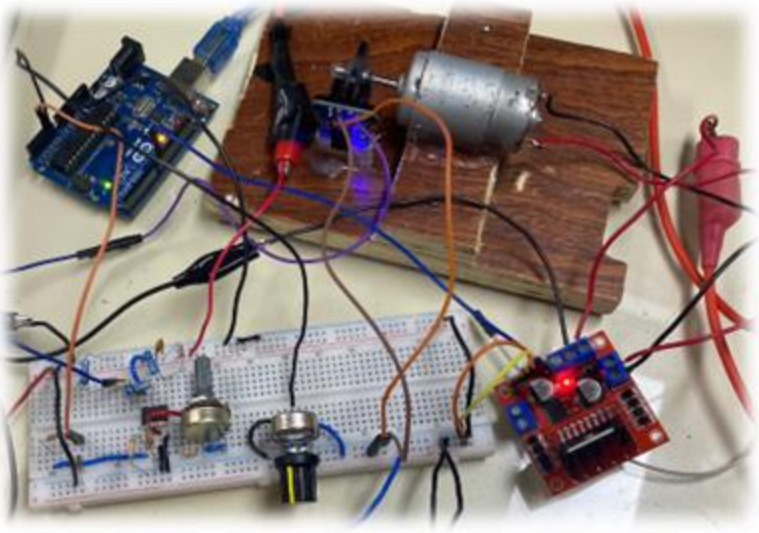




## Procedimiento

Primeramente, se realizar el armado del módulo de conversión de frecuencia a voltaje con el LM331 con el circuito presentado.



Además, se implementó una etapa de a condicionamiento de la señal que consiste en adaptar las señales del sistema para que sean compatibles con el microcontrolador y los demás componentes del circuito. El cual del control de un motor DC con un controlador PID, esto implica convertir las señales de entrada (como el setpoint) y las señales de retroalimentación (como la salida del sensor de velocidad) en formas que el Arduino Uno pueda procesar correctamente.



*Implementación Del Setpoint y Retroalimentación. FIGURA 1*

Implementación de la setpoint.

El setpoint se establece utilizando un potenciómetro conectado al pin analógico A0 del Arduino. La señal del potenciómetro se convierte en un valor digital utilizando el convertidor analógico-digital (ADC) del Arduino. Este tiene un voltaje de referencia de 5 V y GND.

Implementación de la Retroalimentación

La retroalimentación se obtiene a partir de un sensor de velocidad (frecuencímetro) que mide la velocidad del motor. Este sensor genera una señal que fue acondicionada en el cual se añadió un filtro pasa bajas con Fc =2kHz, antes de ser leída por el Arduino.

## Desarrollo matemático

Las operaciones necesarias para diseñar un filtro pasan bajas con una frecuencia de corte de 2 kHz se determinan obteniendo los valores de ( $C$ y $R$ ) utilizando la siguiente fórmula:

$$fc = \frac{1}{2\pi RC}$$

$$Se\ desea\ una\ frecuencia\ de\ corte\ de\ 100Hz$$

$$fc = 100\ Hz$$

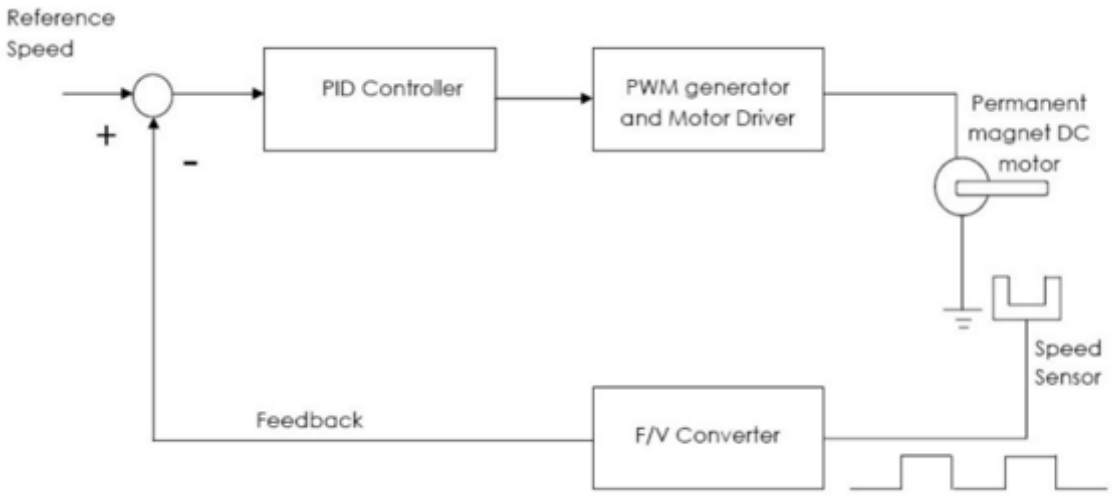
$$100 = \frac{1}{2\pi RC}$$

$$Se\ propone\ el\ Valor\ estándar\ de\ C = 100\ nF\ (C\ 104)$$

$$R = \frac{1}{2\pi * 100\ Hz * 100\ nF} = 15915.49 \approx 16\ k\Omega$$

## Simulación

Para el control digital del motor, el proceso se estructura siguiendo un diagrama de bloques. El cual muestra a continuación los pasos principales del sistema:



Código utilizado en Arduino Uno:

```
int setpointPin = A0; // Pin para el potenciómetro (entrada de referencia)
int sensorPin = A1; // Pin para el conversor (retroalimentación)
int pwmPin = 9; // Pin para la salida PWM (control del motor)
float setpoint = 0;
float sensorrpm = 0;

float x = 0;
float x1 = 0;
float x2 = 0;
float y = 0;
float y1 = 0;

float Kp = 10.0; // Ganancia proporcional
float Kd = 0.1; // Ganancia derivativa
float Ki = 0.1; // Ganancia integral
float a;
float b;
float c;

void setup() {
  pinMode(pwmPin, OUTPUT); // Configurar el pin de salida PWM

  x1 = 0;
  x2 = 0;
  y1 = 0;

  // Calcular constantes de control PID
  a = Kp + Ki + Kd;
  b = Kp + 2 * Kd;
  c = Kd;
}

void loop() {
  // Leer el setpoint del potenciómetro
  setpoint = analogRead(setpointPin);

  // Leer la retroalimentación del sensor de RPM
  sensorrpm = analogRead(sensorPin);

  // Calcular el error
  x = (setpoint - sensorrpm) / 1023.0; // Normalizar la lectura (0 a 1)

  // Calcular la salida del PID
  y = (a * x) - (b * x1) + (c * x2) + y1;

  // Actualizar variables
  y1 = y;
  x2 = x1;
  x1 = x;
```

```
  // Calcular el valor de PWM
  int PWM = int(127.5 + ((y * 255) / 2)); // Ajustar escala para 8 bits (0
a 255)

  // Limitar el valor de PWM para estar dentro del rango válido
  PWM = constrain(PWM, 0, 255);

  // Aplicar el valor de PWM al pin de salida
  analogWrite(pwmPin, PWM);

  // Pequeño retraso para permitir que el bucle se ejecute a una tasa
controlable
  delay(10); // Ajustar según sea necesario
}
```

En esta parte del código se establece los pines de entrada del Setpoint y el Convertidor de F a V, implementamos la salida PWM del módulo HC-020k.
Además de agregar los parámetros de Kp, Ki, Kd, ya obtenidos anteriormente.

```
int setpointPin = A0; // Pin para el potenciómetro (entrada de referencia)
int sensorPin = A1; // Pin para el conversor (retroalimentación)
int pwmPin = 9; // Pin para la salida PWM (control del motor)

float setpoint = 0;
float sensorrpm = 0;
float x = 0;
float x1 = 0;
float x2 = 0;
float y = 0;
float y1 = 0;

float Kp = 10.0; // Ganancia proporcional
float Kd = 0.1; // Ganancia derivativa
float Ki = 0.1; // Ganancia integral

float a;
float b;
float c;

void setup() {
  pinMode(pwmPin, OUTPUT); // Configurar el pin de salida PWM

  x1 = 0;
  x2 = 0;
  y1 = 0;

  // Calcular constantes de control PID
  a = Kp + Ki + Kd;
  b = Kp + 2 * Kd;
  c = Kd;
}
```

En esta parte se determina el error como la diferencia entre el setpoint y la retroalimentación, El setpoint es el valor de referencia, establecido mediante un potenciómetro y la retroalimentación es la señal de salida de un sensor de velocidad.

```
void loop() {
  // Leer el setpoint del potenciómetro
  setpoint = analogRead(setpointPin);

  // Leer la retroalimentación del sensor de RPM
  sensorrpm = analogRead(sensorPin);

  // Calcular el error
  x = (setpoint - sensorrpm) / 1023.0; // Normalizar la lectura (0 a 1)
```

El error calculado se introduce en un controlador PID, El PID procesa el error y genera un voltaje de salida.

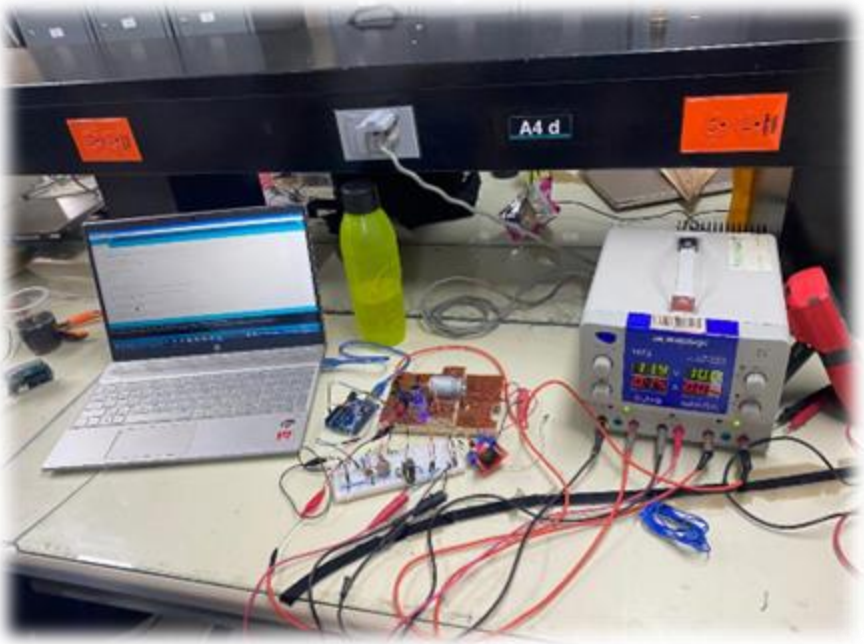
```
  // Calcular la salida del PID
  y = (a * x) - (b * x1) + (c * x2) + y1;

  // Actualizar variables
  y1 = y;
  x2 = x1;
  x1 = x;

  // Calcular el valor de PWM
  int PWM = int(127.5 + ((y * 255) / 2)); // Ajustar escala para 8 bits (0 a 255)

  // Limitar el valor de PWM para estar dentro del rango válido
  PWM = constrain(PWM, 0, 255);

  // Aplicar el valor de PWM al pin de salida
  analogWrite(pwmPin, PWM);

  // Pequeño retraso para permitir que el bucle se ejecute a una tasa controlable
  delay(10); // Ajustar según sea necesario
}
```

Finalmente, la señal de control se convierte en una señal PWM que ajusta la velocidad del motor e interpreta la señal PWM para controlar la dirección y velocidad del motor.

## Pruebas

Implementación del Convertidor de frecuencia a voltaje, el módulo HC-020k y Arduino

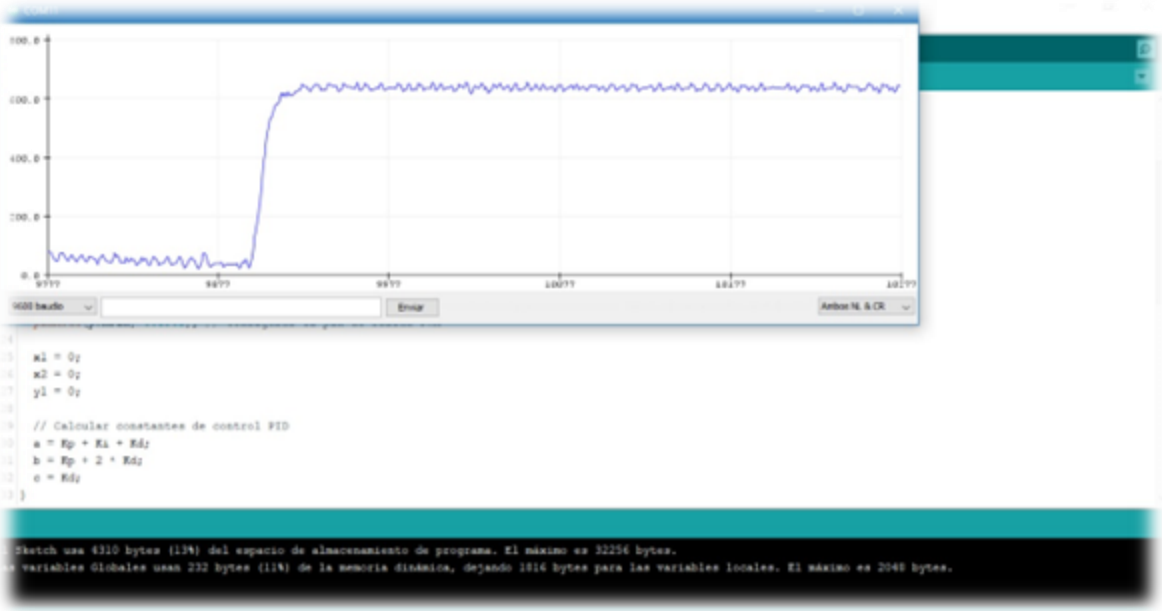
En esta práctica, al no haberse implementado el PID físico ni el PWM, se ajustó esta etapa para que los componentes realizaran su trabajo, reemplazando todo esto por un control digital. Esto incluye la configuración de un PID digital y la adaptación de la señal analógica proveniente del convertidor de frecuencia a voltaje y del setpoint.




Al realizar suficientes pruebas, comprobamos el correcto funcionamiento de nuestro sistema. Además, al variar el setpoint, observamos que la respuesta nos permitía tener un control satisfactorio del motor, concluyendo así las pruebas.

## Resultados

En este momento se determinó graficar la repuesta que se le envía al Módulo HC-020k para ver si funciona como un PWM y ver si este era constate.



Al observar la respuesta, notamos que la señal presentaba cierto ruido, pero mostraba resultados favorables, característicos de un PWM. Este al variar el valor del setpoint, permite disminuir o aumentar, lo que nos permitió obtener el resultado esperado.

## Análisis de resultados

La implementación de un control digital utilizando un Arduino Uno. La transición de un sistema analógico a uno digital implicó varios pasos clave y permitió obtener resultados notablemente superiores.

El uso de un Arduino Uno para el control digital del motor facilitó la implementación. La incorporación de etapas previamente estudiadas, como el convertidor de frecuencia a voltaje y el módulo HC-020k, simplificó el montaje del sistema. Esto se debe a que estos componentes ya habían demostrado su eficacia en la práctica analógica y su integración en el nuevo sistema digital sin complicaciones.

Las pruebas realizadas mostraron que la señal de respuesta, aunque presentaba cierto ruido, era favorable y representativa de un PWM eficiente. Al variar el setpoint, la respuesta del sistema fue conforme a lo esperado, disminuyendo o aumentando según los ajustes realizados. Este comportamiento validó el correcto funcionamiento del control PID digital.

Esto permitió que la señal PWM enviada al motor fuera más precisa y estable, resultando en un control más eficiente del motor.

## Conclusiones

En la práctica anterior, se llevó a cabo el control analógico de un motor. Para esta nueva práctica, se optó por el control digital utilizando un Arduino Uno. La implementación de las etapas previas, que incluían el convertidor de frecuencia a voltaje y el módulo HC-020k, facilitó considerablemente el montaje del sistema. Esto dejó como principal tarea la programación del código necesario para realizar el control PID digital. Este código debía integrar las señales analógicas del conversor y establecer un setpoint para generar una respuesta PWM adecuada.

Al reemplazar los circuitos físicos del sistema analógico por un control digital, se logró un resultado satisfactorio. La respuesta PWM enviada al motor permitió un control preciso, lo que demuestra la eficacia del enfoque digital en la gestión del motor. La transición de un sistema analógico a uno digital no solo simplificó el diseño y la implementación, sino que también mejoró la precisión y la facilidad de ajuste del control del motor. Además, la práctica mostró cómo el uso de un Arduino Uno y la implementación de un PID digital pueden proporcionar un control eficiente y fiable de un motor, optimizando su rendimiento y respuesta.

## Referencias

Bhagat, N. A., & Bhaganagare, M. (2009). *DC Motor Speed Control using PID Controllers*. EE 616 Electronic System Design Course Project, EE Dept, IIT Bombay.

Analog Devices. (2022). LM331 Precision Voltage-to-Frequency Converter (Rev. G). Retrieved from https://www.analog.com/media/en/technical-documentation/data-sheets/LM331.pdf