Sistema Automatizado de Medición de Velocidad Angular

Sebastián Castro, Laura Florez, Juan Pablo Rojas
Universidad Nacional de Colombia

9 de diciembre 2025

Resumen

1. Introducción

En el experimento de momento de inercia, la precisión del resultado final depende principalmente de la medición del periodo de rotación. La técnica manual, propensa significativamente a errores de medición humanos, introduce incertidumbres significativas que afectan directamente la confiabilidad de los cálculos derivados, un ejemplo es el tiempo en que las masas pasan por un punto inicial (x_0). Estos desfases temporales no solo generan errores en el valor medido, sino que también limitan la cantidad de datos obtenidos, lo cual amplifica el error estadístico en la determinación del momento de inercia.

Con el objetivo de cuantificar y reducir estas incertidumbres, este trabajo implementa un sistema automatizado de adquisición de datos basado en un circuito electrónico compuesto por un sensor infrarrojo TCRT5000, un Arduino UNO y una cinta reflectante. Este sistema registra cada revolución del disco, enviando la información a un código que calcula en tiempo real el periodo y la frecuencia de rotación, y posteriormente el momento de inercia del cuerpo. Esto permite mejorar la precisión de la medición y capturar un volumen mayor de datos, facilitando un análisis más fundamentado.

El propósito de este estudio es comparar directamente los resultados obtenidos con el método tradicional analógico y el sistema digital propuesto, evaluando la reducción en los errores de medición y la mejora en la precisión experimental. Se espera que la implementación de este circuito optimice significativamente la confiabilidad de las mediciones en prácticas de laboratorio relacionadas con dinámica rotacional.

2. Marco teórico

2.1. Sensorización Infrarroja para Medición de Tiempo

El sensor TCRT5000 es un dispositivo optoelectrónico compuesto por un emisor de luz infrarroja y un fototransistor receptor. Su principio de operación se basa en la reflectividad: cuando la superficie reflectante (cinta adhesiva) pasa frente al sensor, la luz infrarroja emitida se refleja y es detectada por el fototransistor, generando un cambio abrupto en su señal de salida [1].

- **Modo de operación**: Se configura en modo digital, donde la salida cambia de un estado ALTO a BAJO (o viceversa) al superar un umbral de reflectividad predefinido.
- **Calibración**: La distancia óptima entre sensor y superficie reflectante (≈ 2 cm) y el ajuste del umbral son cruciales para evitar falsas detecciones causadas por vibraciones o irregularidades superficiales.
- **Aplicación**: En este sistema, cada pulso generado corresponde a una revolución completa del disco, permitiendo medir el periodo de rotación (T) como el intervalo entre pulsos consecutivos.

2.2. Teoría de Adquisición de Datos Digitales

La automatización de la medición requiere convertir una señal física (el paso de la cinta reflectante) en datos numéricos procesables. Esto implica:

- **Digitalización de la señal**: El sensor entrega una señal digital limpia de 0V o 5V al Arduino.
- **Muestreo y captura de tiempo**: Se utiliza una interrupción externa en el pin digital del Arduino para detectar el flanco de bajada de la señal. Las interrupciones permiten una respuesta inmediata y precisa, independiente del ciclo principal del programa.
- **Cálculo de periodo y frecuencia**:

$$T = t_i - t_{i-1}, \quad f = \frac{1}{T}$$

  donde t_i es el tiempo de detección del pulso i-ésimo.

2.3. Filtrado Digital de Señales y Anti-Rebote

Las señales provenientes de sensores mecánicos suelen presentar rebotes, que son transitorios rápidos e indeseados que pueden registrarse como múltiples pulsos. Para eliminarlos:

- Se implementa un filtro de anti-rebote por software: tras una detección, se ignora cualquier cambio en la señal durante un intervalo breve. Esto garantiza que solo se registre un pulso válido por evento.
- Además, el código incluye un timeout por inactividad: si no se detecta un pulso en un tiempo predefinido, se asume que el sistema está detenido y se muestran valores nulos.

2.4. Protocolos de Comunicación Serial

La transferencia de datos desde el Arduino hacia un computador para su posterior análisis se realiza mediante comunicación serial UART:

- **Configuración**: Se establece una tasa de transmisión de 115,200 baudios.
- **Formato de datos**: Los datos se envían en formato CSV, con la estructura:

  ```
  tiempo_ms, Periodo_ms, Frecuencia_Hz
  ```

  Este formato es universalmente compatible con software de análisis como Python, Excel o MATLAB. En este caso Excel.
- **Control remoto**: Mediante comandos simples ('R' para iniciar registro, 'S' para detener) enviados desde el monitor serial, se controla el flujo de adquisición sin necesidad de reiniciar el sistema.

2.5. Procesamiento de Datos en Python

Una vez capturados, los datos requieren un procesamiento estadístico para extraer valores representativos:

- **Limpieza de datos**: Se filtran las filas con frecuencia mayor que cero, descartando así los periodos de inactividad.
- **Cálculo de promedios**: Se obtiene el periodo y la frecuencia promedio de todas las mediciones válidas, reduciendo el error aleatorio.

3. Descripción del circuito

Para el montaje analógico se utilizó el sistema tradicional de medición de momento de inercia, compuesto por un tambor giratorio montado sobre un soporte fijo. El sistema incluye una base estable, un plato circula y una cuerda enrollada alrededor del soporte. La medición del tiempo se realizó manualmente con un cronómetro, registrando el intervalo en el que la masa completaba un ciclo de rotación al pasar por un punto de referencia fijo x_0, de manera simultanea se realizaron distintas mediciones usando el circuito de arudino.

La cinta reflectante se adhirió al borde del disco giratorio, asegurando una detección por vuelta completa. El sensor se fijó en un soporte ajustable, manteniendo una distancia constante de 2 cm y alineado perpendicularmente a la trayectoria de la cinta, como se muestra en la Figura 1.

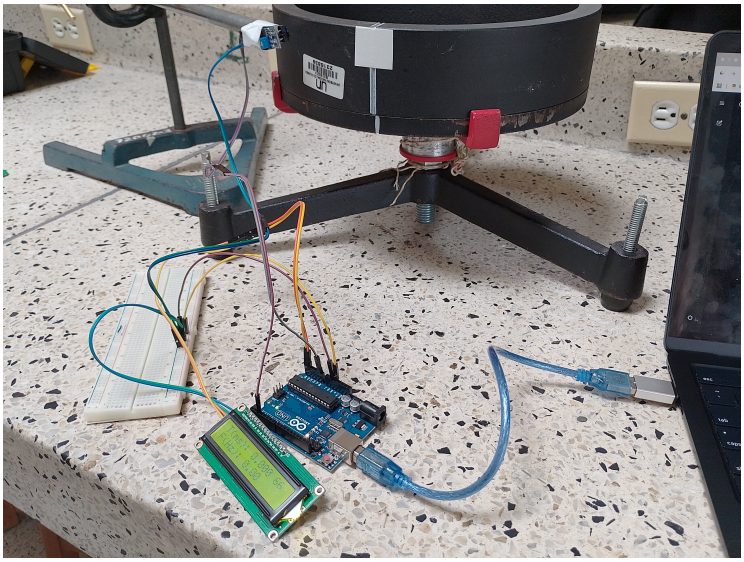

Figura 1: Montaje experimental

Este montaje permite tomar medidas usando 2 métodos distintos para poder realizar una comparación entre las mismas.

3.1. Montaje digital

El montaje digital se implementó para automatizar la adquisición de datos y reducir los errores asociados a la medición manual. El sistema consta de los siguientes componentes en la configuración que se puede ver en la figura en la Figura 2.

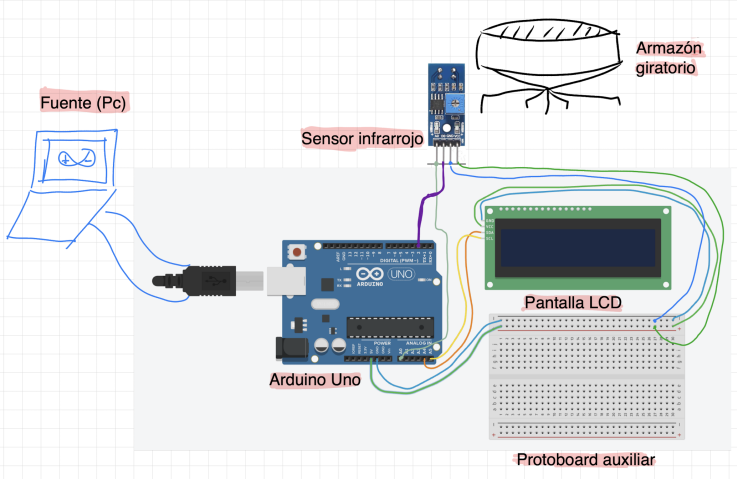

Figura 2: Esquema del circuito

- Arduino UNO [2]: Encargado de procesar las señales del sensor, calcular el periodo y la frecuencia de rotación, y gestionar la visualización en pantalla.
- Sensor infrarrojo TCRT5000: Ubicado frente al disco giratorio, calibrado para detectar la cinta reflectante a una distancia de aproximadamente 2 cm. Genera un pulso digital cada vez que la cinta pasa frente a él.
- Pantalla LCD: Muestra en tiempo real el periodo (en ms) y la frecuencia (en Hz) calculados, junto con el estado de registro de datos.
- Protoboard auxiliar: Utilizada para distribuir las conexiones de alimentación y tierra entre los componentes.
- Computador portátil: Alimenta el Arduino vía USB y ejecuta el entorno Arduino IDE para la programación y el monitoreo serial, además de almacenar los datos registrados en formato CSV.

3.1.1. Diagrama de conexiones

Las conexiones eléctricas se realizaron de la siguiente manera:

Sensor TCRT5000:

- Vcc a la línea positiva de la protoboard.
- GND a la línea negativa de la protoboard.
- Salida digital D0 al pin digital 2 del Arduino.
- Salida analógica A0 al pin análogo A0 del Arduino.

Pantalla LCD

- Vcc a la línea positiva de la protoboard.
- GND a la línea negativa de la protoboard.
- SDA al pin análogo A4 del Arduino.
- SCL al pin análogo A5 del Arduino.

Tanto el sensor como la pantalla se alimentaron desde las líneas de la protoboard, las cuales a su vez recibieron energía del Arduino conectado al puerto USB del computador.

4. Códigos fuente

Para el procesamiento de datos se utilizaron dos códigos principales: uno en Arduino IDE para la adquisición de datos en tiempo real, ademas de un código de Python para el análisis posterior de los datos registrados.

4.1. Código Arduino

El programa implementado en el Arduino UNO (disponible en el Apéndice B) se diseñó para medir el periodo de rotación del disco a partir de las señales generadas por el sensor infrarrojo TCRT5000. Su funcionamiento se basa en una interrupción externa asociada al pin digital 2, cuyo valor es 1 cada vez que el sensor detecta el paso de la cinta reflectante. Para evitar lecturas falsas causadas por vibraciones o rebotes mecánicos, se incorporó un filtro que garantiza que solo se registre un pulso válido por cada detección, ignorando señales invalidas en un pequeño intervalo posterior.

El código calcula automáticamente el periodo (en milisegundos) y la frecuencia (en Hz) a partir del tiempo transcurrido entre pulsos consecutivos. Si no se detecta actividad durante un tiempo límite predefinido, el sistema asume que no hay rotación y muestra valores nulos. Los resultados se visualizan en una pantalla LCD conectada mediante comunicación serial, y también pueden enviarse al puerto serie del computador en formato CSV para su almacenamiento. El registro de datos se controla mediante comandos simples enviados desde el monitor serial: R"para iniciar y "S"para detener.

4.2. Código Python

El script en Python[3] (disponible en el Apéndice A) tiene como objetivo procesar de manera eficiente los archivos CSV generados por el sistema Arduino. Su función principal es leer los datos registrados, filtrar las mediciones válidas (aquellas con frecuencia mayor que cero) y calcular los promedios del periodo y la frecuencia. Este filtrado permite descartar automáticamente los intervalos sin detección, los cuales aparecen como valores nulos en el archivo.

El código utiliza la librería pandas para manejar los datos de forma estructurada y proporciona como salida los valores promedio junto con el número de mediciones útiles consideradas. Estos promedios se emplean posteriormente en la fórmula experimental del momento de inercia, sustituyendo el tiempo t por el periodo promedio convertido a segundos. De esta manera, se obtiene una estimación más robusta y menos susceptible a fluctuaciones aleatorias en las mediciones individuales.

5. Resultados y discusión

Obtuvimos los siguientes resultados al promediar el periodo y la frecuencia usando ambos métodos, véase la Tabla 1.

# Repetición	Promedio Periodo Arduino [s]	Promedio Periodo manual [s]	Diferencia porcentual
1	1060.727	1580.90	49 %
2	1420.6	1236.97	13 %
3	242.75	1146.00	372 %
4	3860.98	1408.57	64 %

Cuadro 1: Periodos calculados

Y por otro lado respecto a la frecuencia en 2:

# Repetición	Promedio Frecuencia Arduino [Hz]	Promedio Frecuencia manual [Hz]	Diferencia porcentual
1	5.88	1.10	81.3 %
2	1.97	0.86	13.0 %
3	4.60	1.00	53.3 %
4	3.16	0.84	73.4 %

Cuadro 2: Frecuencias calculadas

Y calculando la velocidad angular asociada a cada frecuencia:

# Repetición	ω_{Arduino} [rad/s]	ω_{Manual} [rad/s]	Diferencia porcentual
1	36.94	6.91	81.3 %
2	12.38	5.40	13.0 %
3	28.90	6.28	53.3 %
4	19.85	5.28	73.4 %

Cuadro 3: Velocidades angulares calculadas a partir de las frecuencias medidas.

Cantidades que se obtuvieron a partir de la relación:

$$\omega = \frac{2\pi}{T} \tag{1}$$

Análisis de datos

Tras obtener las primeras mediciones con los códigos desarrollados, el análisis de los resultados reveló discrepancias significativas entre los datos adquiridos de forma manual y los registrados por el prototipo automatizado. Ante esta inconsistencia, se procedió a revisar minuciosamente el código, el equipo electrónico y el montaje experimental. Se identificó que el grosor de la cinta reflectante —utilizada como punto de referencia— provocaba que el sensor TCRT5000 registrara

hasta tres lecturas idénticas en intervalos de tiempo distintos, lo cual distorsionaba los promedios calculados. Como contramedida, se implementó en el código un filtro que descarta valores repetidos detectados en ventanas temporales muy reducidas.

Al comparar los valores promedio de velocidad angular obtenidos a partir de los archivos CSV con los resultados manuales, se mantuvo una diferencia considerable. No obstante, se advirtió que la escala temporal de los datos automatizados era aproximadamente diez veces menor que la de las mediciones manuales. Tras verificar la corrección de los cálculos en el firmware de Arduino, se determinó que esta discrepancia se originaba en dos factores principales:

- Falsas detecciones múltiples: El tiempo de respuesta del sensor era demasiado breve, lo que, sumado al efecto del grosor de la cinta, generaba series de hasta tres o más registros consecutivos por cada paso real de la cinta.
- Interferencias del montaje mecánico: Se observaron imperfecciones superficiales en la estructura giratoria (marcas o irregularidades) que el sensor podía interpretar como señales reflectantes adicionales, introduciendo así datos espurios en el archivo de salida.

Aunque la relación cuantitativa entre ambos conjuntos de datos no es exacta, al analizar los órdenes de magnitud se encontró que los valores automatizados presentan una coherencia física más clara con las magnitudes esperadas en el experimento, lo que sugiere que el sistema posee el potencial de ofrecer mediciones más fiables que el método manual.

6. Dificultades

Ahora bien se mencionarán algunos obstaculos encontrados al desarrollar este experimento, tanto en la parte digital, análoga y en el análisis de resultados.

Calibración del equipo

Para poder usar el sensor infrarojo TCRT5000 de manera adecuada, fue necesario llevar a cabo un proceso de calibración que conllevó diversas complicaciones, debido a la sensibilidad que este tiene respecto al umbral de luz en el cual el valor resultante es positivo. Este problema se pudo solucionar adquiriendo otro sensor, dado que al realizar diversas pruebas se concluyó que el que se había adquirido previamente no era apto para los propósitos del proyecto.
Al adquirir un nuevo sensor también se realizó un proceso de calibración, sin embargo en este caso el proceso fue más sencillo y se pudo completar sin mayor complicación.

Toma de datos

Cómo se mencionaba en la sección anterior respecto a la toma de datos, la discrepancia entre los datos en ambas modalidades no se logró solventar de la mejor manera, debido a cuestiones de tiempo y disponibilidad. Sin embargo para próximas oportunidades, se propone realizar un mejor control de variables en el código fuente, y tratar de operar con instrumentos homogéneos, para no obtener datos notablemente desproporcionados que pueden ser desconcertantes.

7. Conclusiones

El desarrollo e implementación de un sistema automatizado de adquisición de datos para la medición de velocidad angular y periodo de rotación permitió evaluar de manera crítica las ventajas y limitaciones de la instrumentación digital frente al método manual tradicional. A partir de los resultados obtenidos y el análisis de las dificultades encontradas, se pueden establecer las siguientes conclusiones:

- El sistema basado en Arduino y el sensor TCRT5000 demostró ser conceptualmente viable para automatizar la medición del periodo de rotación. La capacidad de capturar un volumen de datos significativamente mayor que el método manual y de procesarlos en tiempo real representa una clara ventaja en términos de potencial para reducir el error estadístico y agilizar la experimentación.

- El experimento puso en evidencia que la precisión del sistema digital no depende únicamente de la electrónica y el código, sino también de factores mecánicos y de calibración. La principal fuente de discrepancia con los datos manuales fue la detección múltiple causada por el grosor de la cinta reflectante y las imperfecciones superficiales del disco giratorio, las cuales fueron interpretadas por el sensor como pulsos válidos adicionales.
- Se pudo incorporar un prótotipo que es capaz de detectar señales de forma digital y análoga a partir de la variación de ciertos parámetros.
- Se identificó que la volatidad de los datos y la discrepancia notable es producto tanto de cuestiones de montaje y de factores que se pueden corregir, por lo que es implementable con la variación de distintos parámetros.

Referencias

[1] "Módulo tcrt5000 sensor infrarrojo seguidor de línea." `https://www.ferretronica.com/products/modulo-tcrt5000-sensor-infrarrojo-seguidor-de-linea`. Ferretrónica, s.f.

[2] "Arduino Uno," Nov. 2025. Page Version ID: 1324473596.

[3] Python Core Team, *Python: A dynamic, open source programming language*. Python Software Foundation, 2019. Python version 3.7.

8. Apéndice A: Código en python

```python
import pandas as pd
import io

def analizar_datos_csv(nombre_archivo):
    """
    Lee un archivo CSV existente, filtra los datos activos (Frecuencia > 0) y
    calcula el promedio del Periodo y la Frecuencia para esas lecturas válidas.

    Su entrada es un archivo Csv en este caso
    """
    # Lectura
    try:
        # Leer el archivo CSV en pandas
        df = pd.read_csv(nombre_archivo)

        #Mostrar número de filas leídas
        print(f"Total de filas leídas: {len(df)}")
        print(df.head())
        print("-" * 40)

        # Filtración
        # Filtrar las filas donde la frecuencia es mayor que cero para obtener solo las
        #lecturas válidas del sensor
        df_activos = df[df['Frecuencia_Hz'] > 0]

        # Si hay datos distintos de cero, calcular sus promedios
        if not df_activos.empty:
            promedio_periodo_activo = df_activos['Periodo_ms'].mean()
            promedio_frecuencia_activa = df_activos['Frecuencia_Hz'].mean()
        else:
            # Para caso de todo 0
            promedio_periodo_activo = 0.0
            promedio_frecuencia_activa = 0.0

        # Impresión
        print("\nResultados de promedio")
        print("\n Promedios de Datos útiles (Frecuencia > 0)")
        # Cuantas filas habia en el CSV con archivos no nulos
        print(f"Filas con datos activos utilizadas en el promedio: {len(df_activos)}")

        if len(df_activos) > 0:
            print(f"Promedio del Periodo: {promedio_periodo_activo:.3f} ms")
            print(f"Promedio de la Frecuencia: {promedio_frecuencia_activa:.3f} Hz")
        else:
            print("No se encontraron lecturas activas (Frecuencia > 0) para calcular el promedio."

        print("==============================")

    except FileNotFoundError:
        # Mensaje de error por si no se encuentra el archivo
        print(f"\nError: El archivo '{nombre_archivo}' no fue encontrado.")
        print(f"Revisar '{nombre_archivo}' y que esté en esta carpeta.")

# Ejecutar la función principal
if __name__ == "__main__":
    # Se llama al archivo al que se le vayan a hacer los promedios
    analizar_datos_csv("Datos_1.csv")
```

Figura 3: Script de Python para el análisis y promedio de datos CSV.

9. Apéndice B: Código en Arduino

```
/*Medición de Periodo/Frecuencia con interrupción.
 * El registro de datos en formato CSV se activa/desactiva con r de record y s de stop, y con enter

 * HARDWARE:
 * - Sensor TCRT5000 (Salida Digital DO) -> Pin 2
 * - LCD SDA -> Pin A4
 * - LCD SCL -> Pin A5
 */
# include <Wire.h>
# include <LiquidCrystal_I2C.h>

// Configuración de la pantalla LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Definición de pines
const int sensorPin = 2;    // Pin para el sensor

// Variables volátiles
volatile unsigned long tiempoAnterior = 0;
volatile unsigned long periodoActual = 0;
volatile bool nuevoPulso = false;

// Variables globales
unsigned long periodoUltimoValido = 0;
unsigned long ultimaDeteccionGlobal = 0;
// 2 segundos en microsegundos (para forzar 0 Hz si no hay actividad)
const unsigned long timeoutSensor = 2000000;

// Variables para actualización
unsigned long ultimaActualizacionLCD = 0;
const int intervaloLCD = 300; // Actualizar pantalla cada 300ms

// Variables de estado del logging
bool isLogging = false;

// Funciones
void sensorISR();
void mostrarDatosLCD(float t, float f, bool loggingState);
void manejarControlSerial();

void setup() {
  Serial.begin(9600);

  // Instrucciones para el usuario
  Serial.println("Presionar 'R' para iniciar la grabación (CSV) o 'S' para detener");
  // Inicializar LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Iniciando...");
  delay(1000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("T(ms):");
  lcd.setCursor(0,1);
  lcd.print("F(Hz):");
```

```
  mostrarDatosLCD(0.00, 0.00, false);

  // 2. Pines
  // Sensor con PULLUP
  pinMode(sensorPin, INPUT_PULLUP);

  // 3. Inicializar Tiempos
  ultimaDeteccionGlobal = micros();
  tiempoAnterior = micros();

  // 4. Configuración Interrupción del Sensor
  attachInterrupt(digitalPinToInterrupt(sensorPin), sensorISR, FALLING);
}

void loop() {

  // Control serial
  manejarControlSerial();

  // Lectura y manejo
  if (nuevoPulso) {
    noInterrupts();
    periodoUltimoValido = periodoActual;
    nuevoPulso = false;
    interrupts();

    ultimaDeteccionGlobal = micros();
  }

  // Aquí manda el valor a cero si no está midiendo nada en el intervalo propuesto
  if (micros() - ultimaDeteccionGlobal > timeoutSensor) {
    periodoUltimoValido = 0;
  }

  // Cálculo y registro
  // aqui se establece un intervalo para la toma de datos
  if (millis() - ultimaActualizacionLCD >= intervaloLCD) {
    ultimaActualizacionLCD = millis();

    float frecuencia = 0.0;
    float periodoMs = 0.0;

    if (periodoUltimoValido > 0) {
      frecuencia = 1000000.0 / periodoUltimoValido;
      periodoMs = periodoUltimoValido / 1000.0;
    }

    // 1. Impresión de la pantalla
    mostrarDatosLCD(periodoMs, frecuencia, isLogging);

    // 2. Registro de datos CSV (solo si isLogging es TRUE)
    // El Serial.print solo ocurre aquí, asegurando que no se mezclen mensajes.
    if (isLogging) {
      // Formato CSV: Tiempo_ms,Periodo_ms,Frecuencia_Hz
      Serial.print(millis());
      Serial.print(",");
      Serial.print(periodoMs, 3); // 3 decimales
      Serial.print(",");
      Serial.println(frecuencia, 3); // 3 decimales y nueva línea
    }
  }
}

// Control serial
void manejarControlSerial() {
```

```
  if (Serial.available() > 0) {
    char incomingByte = Serial.read();
  // se inicia el registro
    if (incomingByte == 'R' || incomingByte == 'r') {
      if (!isLogging) { // Solo iniciar si no está grabando
        isLogging = true;
        // Marcador de inicio
        Serial.println("\n*** INICIO CSV ***");
        Serial.println("Tiempo_ms,Periodo_ms,Frecuencia_Hz");
      }
  // para parar el registro
    } else if (incomingByte == 'S' || incomingByte == 's') {
      if (isLogging) { // Solo detener si está grabando
        isLogging = false;
        // Marcador de fin de datos CSV
        Serial.println("*** FIN CSV ***\n");
      }
    }
    while (Serial.available() > 0) {
      Serial.read();
    }
  }
}

// Función de Interrupción
void sensorISR() {
  unsigned long tiempoActual = micros();

  // Anti-rebote establecido a 1ms (1000us)
  if (tiempoActual - tiempoAnterior > 1000) {
    periodoActual = tiempoActual - tiempoAnterior;
    tiempoAnterior = tiempoActual;
    nuevoPulso = true;
  }
}

// Función auxiliar para limpiar y escribir en LCD
void mostrarDatosLCD(float t, float f, bool loggingState) {
  // Fila 1: Periodo (T)
  lcd.setCursor(6, 0);
  lcd.print("          "); // Limpia 10 espacios
  lcd.setCursor(6, 0);
  lcd.print(t, 3);     // 3 decimales

  // Fila 2: Frecuencia (F)
  lcd.setCursor(6, 1);
  lcd.print("          "); // Limpia 10 espacios
  lcd.setCursor(6, 1);
  lcd.print(f, 3);     // 3 decimales

  // Estado de registro (Columna 12, Fila 1)
  lcd.setCursor(12, 1);
  if (loggingState) {
    lcd.print(" REC"); // Registrando
  } else {
    lcd.print("STOP"); // Detenido
  }
}
```

Figura 4: Código de Arduino para la medición de Frecuencia/Periodo con interrupción.