**Universidad de Costa Rica**
**Escuela de Ingeniería Eléctrica**
**IE-0624 Laboratorio de Microcontroladores**

EIE
Escuela de
Ingeniería Eléctrica

# Laboratorio #3

UNIVERSIDAD DE
COSTA RICA

## Grupo 01

**Profesor:** MSc. Marco Villalta Fallas



| | |
|---|---|
| **Estudiante:** Luis Brenes Campos | **Carné:** C21324 |
| **Estudiante:** Elizabeth Matamoros | **Carné:** C04652 |



I SEMESTRE 2025

# Índice

# 1. Introducción/Resumen

El laboratorio tuvo como objetivo el diseño, implementación y prueba de un sistema voltímetro de cuatro canales utilizando un microcontrolador Arduino UNO. Este sistema debía ser capaz de medir señales de voltaje tanto en corriente continua (DC) como en corriente alterna (AC), dentro de un rango de entrada de [–24 V, +24 V]. Para lograrlo, se desarrolló un circuito de acondicionamiento de señal que adapta estos voltajes al rango de operación del convertidor analógico-digital (ADC) del microcontrolador, que opera entre 0 y 5 V. Esta etapa incluyó un sumador no inversor para aplicar un offset de 24V a la señal, una etapa de rectificación con filtro capacitivo para adaptar señales en AC, y un divisor resistivo diseñado para escalar el voltaje al rango operativo del ADC del microcontrolador.

Además del procesamiento analógico, el sistema incorpora una interfaz gráfica mediante una pantalla LCD PCD8544 (Nokia 5110), donde se visualizan los valores medidos. La lógica del programa en Arduino permite seleccionar entre los modos DC y AC mediante un switch, calcular el valor promedio o RMS según corresponda, y activar una alarma visual con un LED por canal cuando se detectan voltajes superiores a ±20 V. También se integró comunicación serial mediante USART, lo que permite transmitir los valores leídos hacia una computadora, donde un script en Python los registra en un archivo CSV para su posterior análisis.

Durante el desarrollo del laboratorio se implementaron con éxito todas las funcionalidades planteadas. Las mediciones obtenidas fueron coherentes con los valores reales, con pequeñas desviaciones atribuibles a tolerancias en los componentes, ruido en la lectura del ADC y limitaciones de la simulación. La visualización en pantalla funcionó correctamente y la transmisión de datos a la PC fue estable y precisa. La lógica de alarmas respondió adecuadamente al superar los umbrales establecidos.

En conclusión, este laboratorio permitió aplicar conocimientos fundamentales de acondicionamiento de señales analógicas y programación en Arduino. El sistema desarrollado logró cumplir con los objetivos establecidos, mostrando un funcionamiento adecuado en simulación y una lógica de operación coherente con los requerimientos del experimento.

# 2. Nota Teórica

## 2.1. Información general MCU

El Arduino UNO es una placa de desarrollo electrónica de código abierto ampliamente utilizada tanto en entornos académicos como en proyectos profesionales. Su diseño se basa en el microcontrolador ATmega328P, que proporciona una arquitectura RISC de 8 bits, memoria Flash de hasta 32 KB, 2 KB de SRAM y 1 KB de EEPROM. Esta arquitectura permite un rendimiento eficiente y bajo consumo energético, ideal para sistemas embebidos como el del laboratorio en cuestión .

Además, el Arduino UNO incluye un segundo microcontrolador, el ATmega16U2, el cual actúa como un puente de comunicación entre el puerto USB y el puerto serial del ATmega328P.

Este chip adicional permite la programación directa desde la computadora sin necesidad de un conversor externo, facilitando el desarrollo y depuración del código.

## Características principales

- **Microcontrolador principal:** ATmega328P
  - Arquitectura AVR de 8 bits
  - Frecuencia de reloj: 16 MHz
  - Memoria Flash: 32 KB (0.5 KB usados por el bootloader)
  - SRAM: 2 KB
  - EEPROM: 1 KB
- **Microcontrolador secundario:** ATmega16U2 (conversor USB a serie)
- **Entradas/Salidas:**
  - 14 pines digitales (6 con salida PWM)
  - 6 entradas analógicas
- **Voltajes:**
  - Voltaje de operación: 5 V
  - Voltaje de entrada recomendado: 7–12 V
  - Voltaje de entrada (límites): 6–20 V
- **Corriente:**
  - Corriente por pin de E/S: 20 mA
  - Corriente para pin de 3.3 V: 50 mA
- **Otros:**
  - Oscilador cerámico de 16 MHz
  - Conexión USB tipo B
  - Encabezado ICSP
  - Botón de reinicio

## Capacidades de comunicación

El Arduino UNO permite múltiples tipos de comunicación:

- **UART:** Comunicación serial TTL en los pines 0 (RX) y 1 (TX)
- **USB a serial:** mediante el microcontrolador ATmega16U2
- **Protocolos adicionales:** Soporta $I^2C$ y SPI

## Programación

El Arduino UNO se programa a través del entorno *Arduino IDE*, que permite escribir, compilar y cargar código fácilmente. El ATmega328P incluye un bootloader que facilita la carga sin hardware adicional.

Aunado a ello, Arduino IDE permite la exportación del código en binario. Esta opción facilita las simulaciones en entornos como SimuIDE.

## 2.2. Diagrama de bloques y de pines

El diagrama de pines es el siguiente [1]:

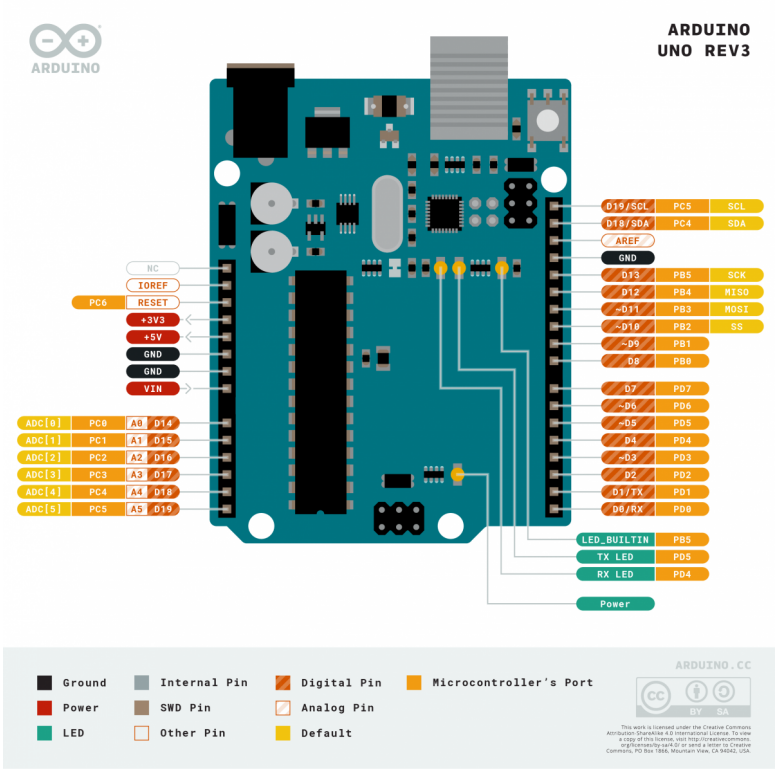


Figura 1: Diagrama de pines de Arduino UNO

El diagrama de bloques (esquemático) es el siguiente. Este se encuentra separado en las 3 partes principales: alimentación, comunicación USB y procesamiento [2]:

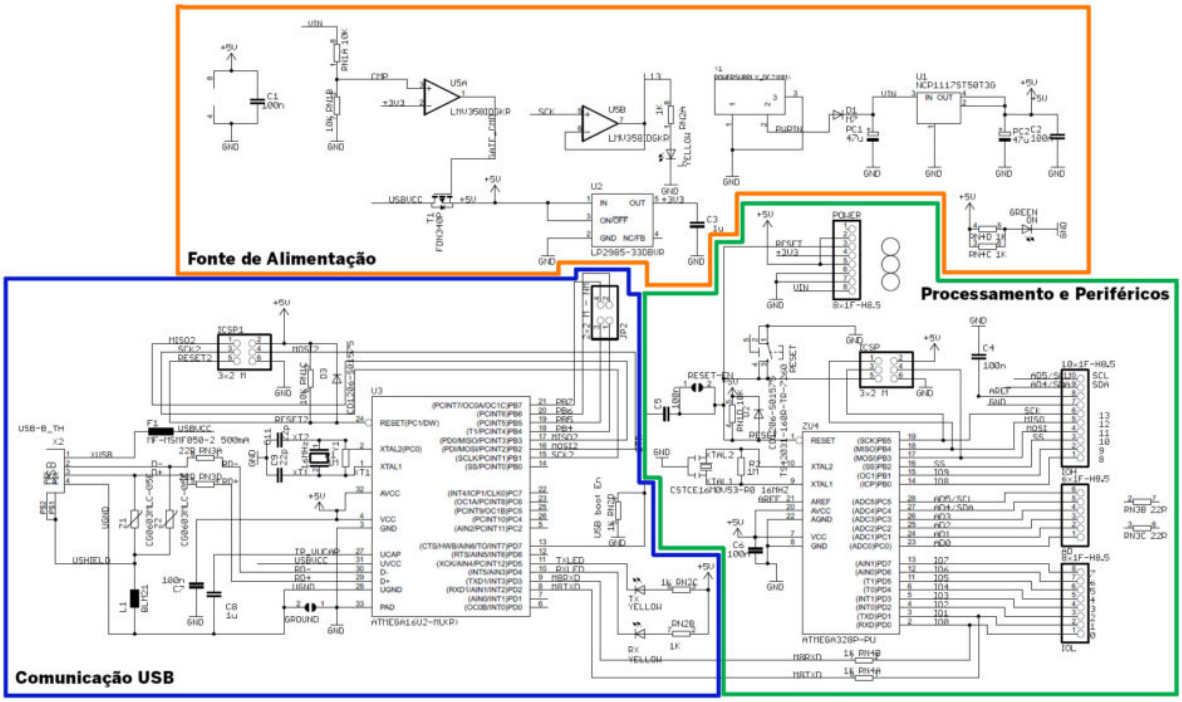


Figura 2: Esquemático de Arduino UNO

## 2.3. Periférico:

### 2.3.1. Pantalla LCD PCD8544 (Nokia 5110)

El LCD PCD8544, también conocido como Nokia 5110, es una pantalla gráfica monocromática de 48×84 píxeles, originalmente utilizada en teléfonos móviles Nokia. Esta pantalla es controlada por el chip controlador/driver PCD8544 de Philips, el cual permite su fácil integración con microcontroladores mediante una interfaz serial tipo SPI. Es ampliamente utilizada en proyectos de electrónica y sistemas embebidos debido a su bajo costo, bajo consumo energético y facilidad de uso [3]. Para este laboratorio, la pantalla se encargará de la visualización de las tensiones.

Sus especificaciones técnicas son las siguientes [3]:

| **Característica** | **Valor** |
|---|---|
| Resolución | 48 × 84 píxeles |
| Controlador | PCD8544 (Philips) |
| Voltaje de operación | 2.7V a 3.3V |
| Consumo de corriente | Aproximadamente 6 mA |
| Interfaz | SPI (Serial Peripheral Interface) |
| Retroiluminación | 4 LEDs (rojo, verde, azul, blanco) |

Tabla 1: Especificaciones técnicas del LCD PCD8544

A continuación, se detallan los pines utilizados por el módulo LCD PCD8544:

- **RST**: Pin de reinicio.
- **CE**: Chip Enable (habilita la comunicación con el controlador).
- **DC**: Data/Command (selección de datos o comandos).
- **DIN**: Data In (entrada de datos).
- **CLK**: Clock (señal de reloj).
- **BL**: Backlight (retroiluminación).
- **VCC**: Alimentación (3.3V).
- **GND**: Tierra (0V).

Para más detalles sobre la conexión y uso del LCD PCD8544 con Arduino, puedes consultar el tutorial completo en [3].

## 2.4. Registros

## 2.5. Diseño del circuito

El sistema desarrollado requiere medir señales de voltaje dentro del rango de [-24V,24V], lo cual excede los límites de entrada del conversor analógico-digital (ADC) del Arduino UNO, que trabaja en el rango de [0V,5V]. Por esta razón, se diseñó un circuito de acondicionamiento de señal que permite reducir y adaptar la señal de entrada a un rango seguro y proporcional.

### Circuito de Acondicionamiento de Señal

La figura muestra el circuito utilizado para adaptar la señal de entrada. Este se basa en dos etapas de amplificación con operacionales configurados como amplificadores no inversores con retroalimentación y referencia a un punto medio de 2.5V. La configuración permite centrar la señal de entrada en 2.5V y reducir su amplitud mediante una red resistiva tipo divisor.

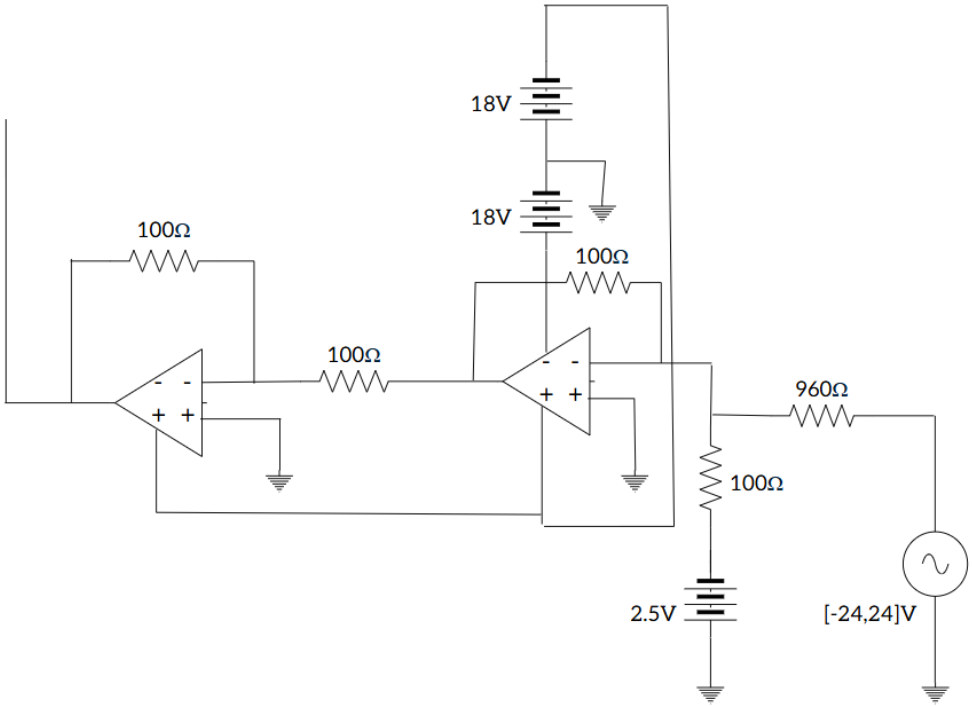


Figura 3: Circuito de Acondicionamiento de Señal

Este diseño logra comprimir el rango de entrada original de 48Vpp(±24V). a un rango de 5Vpp centrado en 2.5V, usando una relación de atenuación de aproximadamente $\frac{5}{48}$. Esta relación es coherente con el factor de escala aplicado luego en software para recuperar el valor real:

$$V_{real} = V_{ADC} \cdot \frac{5}{48} - 24 \tag{1}$$

## Diagrama de la conexión con el Arduino y dispositivos periféricos

La figura muestra un diagrama de la conexión entre los módulos del sistema: Arduino UNO, pantalla LCD PCD8544, LED de alarma, switches de control y el circuito de medición.

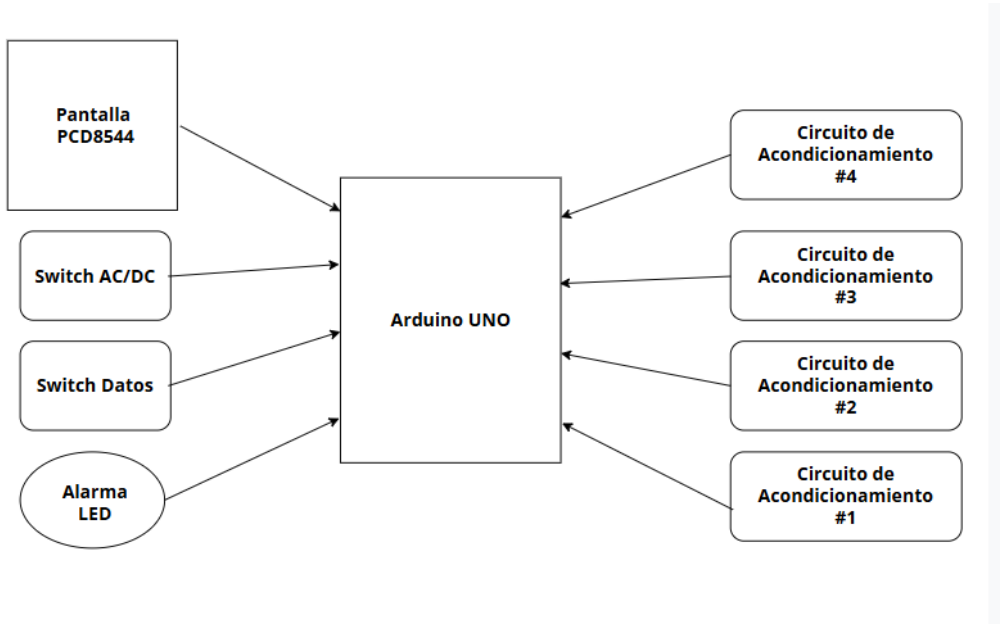


Figura 4: Diagrama de la conexión con el Arduino y dispositivos periféricos

## Circuito completo implementado en SimulIDE

La figura del diagrama completo implementado en SimulIDE es la siguiente:

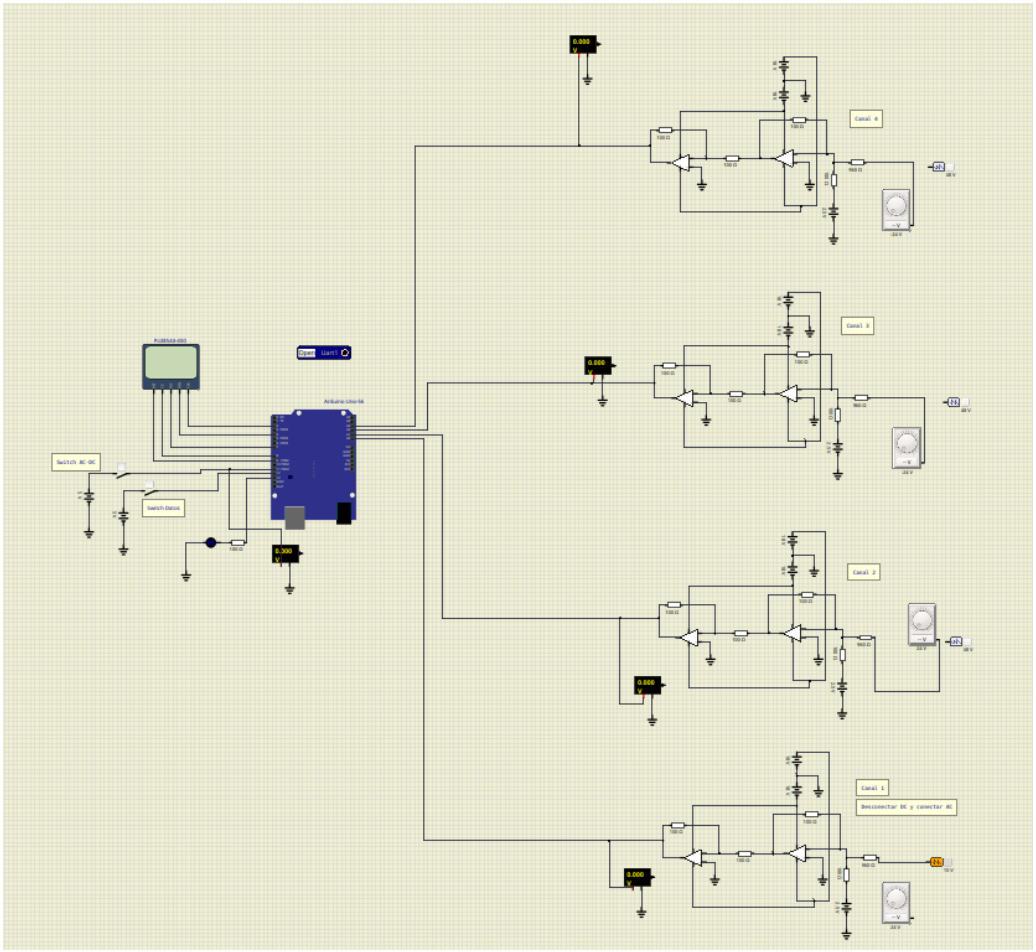

Figura 5: Circuito implementados en SimulIDE

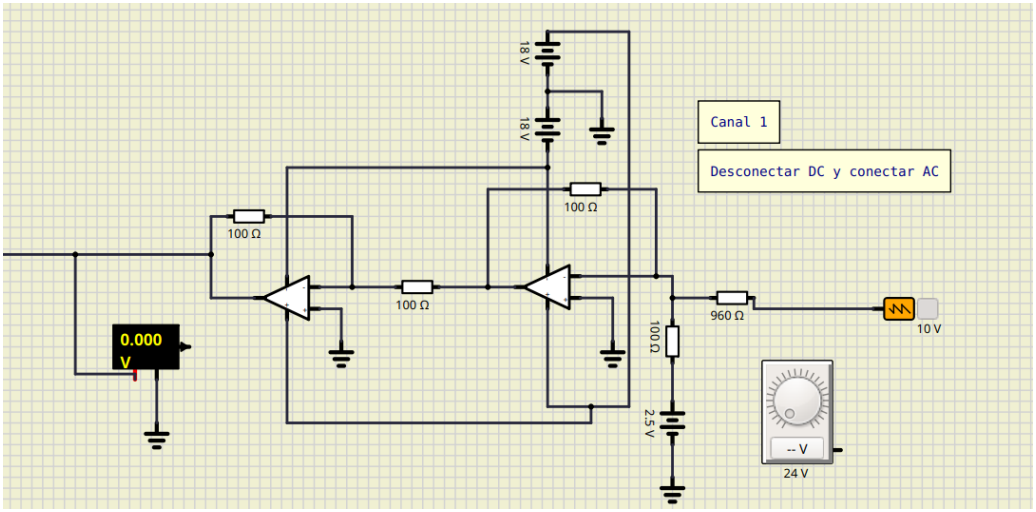


Figura 6: Circuito de Acondicionamiento de Señal

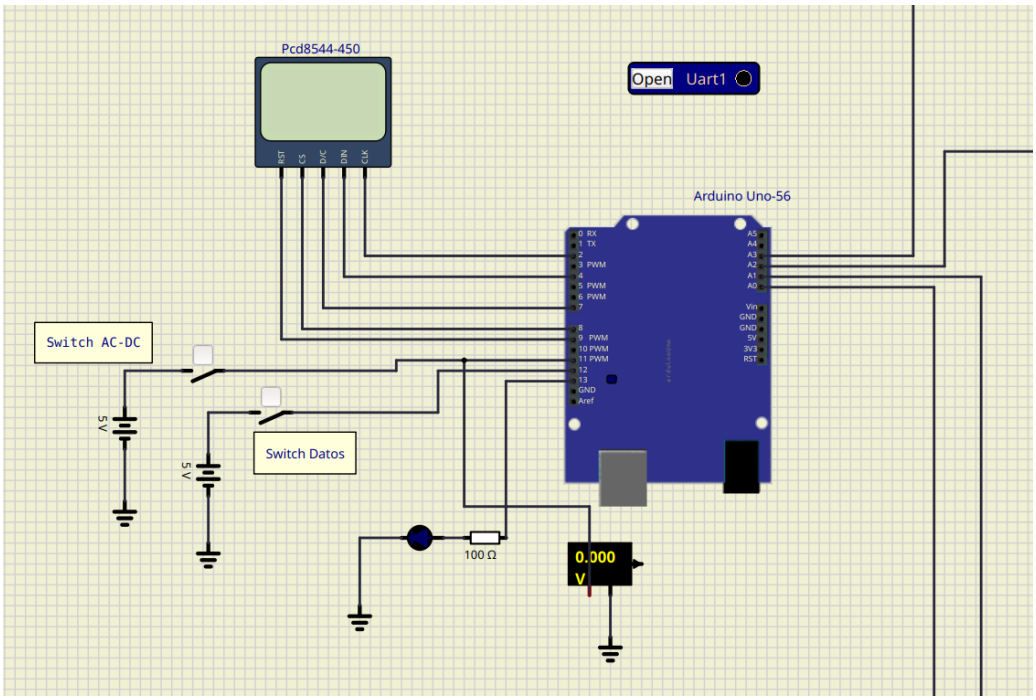


Figura 7: Circuito Conexiones

Están integradas todas las conexiones entre los modulos y dispositivos. El voltaje acondicionado ingresa al ADC del microcontrolador, y los pines digitales gestionan las funciones de visualización, transmisión y alarma.

El display PCD8544 se comunica mediante protocolo SPI con el Arduino (pines 2, 4, 7, 8, 9). Se emplean switches físicos conectados a los pines 11 y 12 para seleccionar el modo de medición (AC/DC) y habilitar/deshabilitar la transmisión de datos por USART.

El pin digital 13 controla un LED que se enciende si alguno de los canales excede el rango permitido de operación. Además, la salida serial (TX) está conectada al módulo UART del simulador o al puerto USB para transmisión hacia la PC.

## 2.6. Lista de componentes y precios

### Tabla de componentes

| Componente | Cantidad | Precio Total |
|---|---|---|
| Amplificadores Operacionales (LM358P) | 8 | $8 |
| LED rojo | 0.12 | $1.00 |
| Switch | 2 | $2.56 |
| Resistencia de 100 Ω | 17 | $1 |
| Resistencia de 960Ω | 4 | $0.54 |
| Pcd8544 | 1 | $11.42 |
| Arduino UNO | 1 | $14.77 |

Tabla 2: Lista de componentes y costos

### Fuentes de adquisición

Los siguientes componentes pueden adquirirse en las siguientes tiendas o plataformas en línea:

- **LED rojo** – Disponible en `https://www.steren.cr/catalogsearch/result/?q=LED+ROJO`.
- **Resistencia de 100Ω** – Disponible en `https://www.steren.cr/catalogsearch/result/?q=R100`.
- **Resistencia de 960Ω** – Disponible en `https://www.steren.cr/catalogsearch/result/?q=R1k`.
- **Amplificadores operacionales LM358P** – Disponible en `https://electrocr.tech/tienda/%f0%9f%94%b2-lm358p/`.
- **PCD8544** – Disponible en `https://www.steren.cr/display-para-raspberry.html?srsltid=AfmBOoq5Gn7LE-moV4TCQWllacYHhwSLx61rK4MfWAu872BBSRehPcTy`.
- **Arduino UNO** – Disponible en `https://electrocr.tech/tienda/arduino-uno-r3-gen/`.
- **Switch** – Disponible en `https://electrocr.tech/tienda/switch-palanca/`.

## Conceptos del laboratorio

A continuación se listan los conceptos extraídos de la presentación teórica proporcionada por el profesor, los cuales son relevantes para la implementación del laboratorio:

- **GPIO (E/S digitales)**:
  - Voltajes de operación: 0 y 5V.
  - Corriente máxima por pin: 20 mA (100 mA total).

  - Funciones: `pinMode()`, `digitalWrite()`, `digitalRead()`, `pulseIn()`.
  - Uso de `INPUT_PULLUP` para entradas digitales.

- **E/S Analógicas (ADC)**:
  - Rango: 0 a 5V, resolución de 10 bits (0–1023).
  - Paso de cuantización: 5 mV.
  - Función: `analogRead()`.
  - Tiempo de conversión: 100 µs.
  - Cambio de referencia con `analogReference()`.

- **PWM (Modulación por ancho de pulso)**:
  - Control de dispositivos analógicos mediante salida digital.
  - Función: `analogWrite()` (valor entre 0 y 255).
  - Cálculo de voltaje medio: $V_{\text{medio}} = \frac{DC}{100} \cdot V_{\text{alto}}$.

- **Funciones de utilidad**:
  - `map(value, fromLow, fromHigh, toLow, toHigh)`: mapeo lineal entre rangos.
  - `millis()`: milisegundos desde el inicio del programa.

- **Comunicación Serial (USART)**:
  - Protocolos: UART (asíncrono), USART (sincrónico y asíncrono).
  - Pines del Arduino UNO: `TX (1)` y `RX (0)`.
  - Funciones de transmisión: `Serial.begin()`, `Serial.print()`, `Serial.println()`, `Serial.write()`, `Serial.end()`.
  - Funciones de recepción: `Serial.available()`, `Serial.read()`, `Serial.peek()`, `Serial.readB`

- **Pantalla PCD8544 (Nokia 5110)**:
  - Pines: `RST, SCE, D/C, DN, SCLK, VCC, LED, GND`.
  - Voltaje de operación: 2.7–3.3V.
  - Comunicación SPI.
  - Uso de la librería `PCD8544.h`.
  - Funciones: `lcd.begin()`, `lcd.setCursor()`, `lcd.print()`, `lcd.clear()`.

# 3. Desarrollo/Análisis de resultados

## 3.1. Análisis programa

El programa desarrollado cumple con todos los requerimientos funcionales del laboratorio, implementando un sistema de medición multicanal con procesamiento de señales analógicas, visualización en pantalla y transmisión de datos hacia un computador vía USART. A continuación se presenta un diagrama de bloques que sigue el proceso que se ejecutó en el programa y un análisis detallado de cada bloque funcional del sistema::

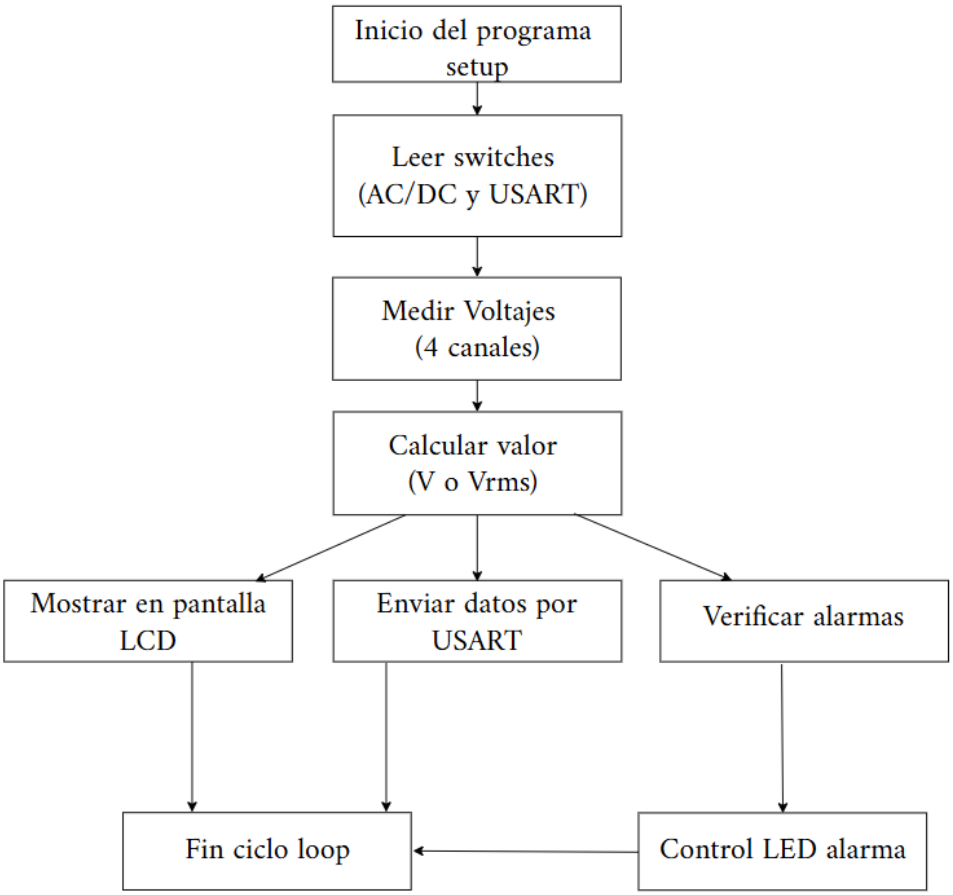


Figura 8: Diagrama de bloques del programa

### Inicialización del sistema

La función `setup()` se encarga de la configuración inicial del hardware y periféricos involucrados, incluyendo:

- Inicialización de la pantalla LCD PCD8544 con `lcd.begin(84, 48)` y ajuste de su contraste con `lcd.setContrast(60)`.
- Configuración de la comunicación serial a 9600 baudios mediante `Serial.begin(9600)`.
- Se definen los pines de entrada y salida del sistema:

```
            pinMode(ALARM_LED_PIN, OUTPUT);
            pinMode(SERIAL_ENABLE_PIN, INPUT);
            pinMode(MODE_SWITCH_PIN, INPUT);

```

Este bloque asegura que todos los dispositivos estén listos antes de iniciar el ciclo principal.

## Lectura del modo de operación y control USART

Dentro del ciclo `loop()`, se determina si el sistema debe operar en modo DC o AC mediante un switch conectado al pin digital:

```
        bool isDC = (digitalRead(MODE\_SWITCH\_PIN) == LOW);
        bool sendCSV = (digitalRead(SERIAL\_ENABLE\_PIN) == LOW);

```

Estas señales controlan el tipo de procesamiento de las señales de entrada y si se debe realizar o no la transmisión por USART.

## Adquisición de señales analógicas

El sistema utiliza cuatro canales analógicos del Arduino UNO (A0 a A3) para medir simultáneamente cuatro señales de voltaje. El acondicionamiento de señal previo convierte el rango de entrada de [-24V,24V] a [0V,5V] para que pueda ser muestreado por el ADC interno del microcontrolador.

**Modo DC**: Se utiliza la función `leerCanalDC()` para cada canal, la cual convierte el valor leído por el ADC a un voltaje real escalado:

```
        float raw = analogRead(pinCanal) * (5.0 / 1023.0);
        salidaVar = raw * (48.0 / 5.0) - 24.0;

```

Esta fórmula aplica un factor de escalamiento correspondiente al diseño del circuito de acondicionamiento: una ganancia de $\frac{48}{5}$ seguida de un corrimiento de -24V.

**Modo AC**: Se emplea la función `leerCanalAC()`, que estima el valor pico de la señal en un ciclo de lectura y lo convierte en valor eficaz (RMS) utilizando la fórmula:

```
        salidaVar = (raw * (48.0 / 5.0) - 24.0) / sqrt(2.0);

```

Esto asume una señal sinusoidal pura, y el número de muestras (por defecto 1000) asegura una buena resolución temporal para captar el valor pico.

## Visualización en pantalla LCD

Los valores medidos se presentan en una pantalla Nokia 5110 (PCD8544) en tiempo real. La pantalla se actualiza cada segundo con el siguiente bloque de código:

```
        lcd.setCursor(0, 1);
        lcd.print("V1: "); lcd.print(V1); lcd.print(" V");

```

Esto se repite para los cuatro canales y en ambos modos (DC y AC), usando etiquetas diferentes para distinguir el tipo de medición ("V en DC", "V en AC").

## Transmisión de datos por USART

Cuando el switch de transmisión está activo, los datos se envían hacia una computadora vía USART. Se implementa una salida por consola usando el puerto serial:

```
1         Serial.print("V1: "); Serial.println(V1);
2
```

Esto permite que un script en Python lea la información y la almacene como un archivo CSV, facilitando el análisis posterior o la visualización remota de los datos.

## Sistema de alarmas

El sistema evalúa continuamente si alguno de los voltajes medidos está fuera del rango permitido [-20V,20V]. En ese caso, se activa un LED de alarma:

```
1         if ((V1 > 20.0) || (V1 < -20.0) || ...) {
2         digitalWrite(ALARM_LED_PIN, HIGH);
3         }
4         else {
5         digitalWrite(ALARM_LED_PIN, LOW);
6         }
7
```

Esta función permite detectar condiciones potencialmente peligrosas o errores en el circuito conectado al canal.

## Ciclo continuo y retardo

Finalmente, el ciclo del programa se completa con una pausa controlada por `delay(1000)`, permitiendo una frecuencia de actualización de 1 Hz que es adecuada para una lectura de voltajes relativamente estables.

## Validación del sistema y funcionamiento general

Durante las pruebas del sistema en el simulador y en la implementación física:

- Se verificó que los cuatro canales muestran valores consistentes con las señales aplicadas, tanto en AC como en DC.
- El cálculo RMS funciona correctamente para señales sinusoidales simuladas.
- La activación del LED de alarma fue exitosa ante señales mayores a ±20V.
- Los datos enviados por el puerto serial fueron recibidos en el script en Python y guardados correctamente en archivos CSV.
- La pantalla LCD mostró un rendimiento óptimo y buena legibilidad en pruebas prolongadas.

## 3.2. Análisis electrónico

En esta sección se presenta la validación funcional del sistema diseñado, verificando que cada módulo cumple con los requerimientos establecidos: adquisición de señales AC y DC, procesamiento digital, visualización en pantalla, activación de alarmas y transmisión de datos a través del puerto serial.

### Verificación del funcionamiento en modo DC

Se aplicaron señales de voltaje continuo a cada uno de los canales, dentro del rango permitido [-24V,+24V]. El sistema procesó correctamente las señales mediante la función `leerCanalDC()`, realizando el escalamiento y corrimiento apropiado para mostrar el valor real en la pantalla LCD.

En la Figura 9, se observa que los valores mostrados en la pantalla corresponden con los voltajes aplicados con solo una diferencia de 0.05 V, validando el funcionamiento del circuito de acondicionamiento, el correcto uso del ADC y el procesamiento digital.

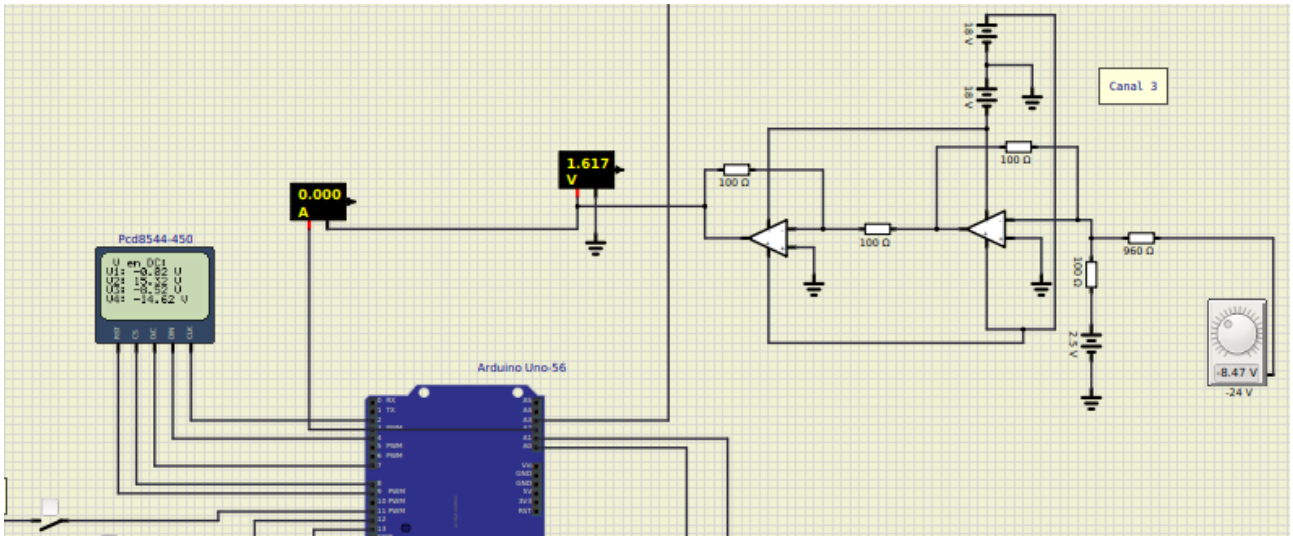


Figura 9: Verificación de la tensión DC

Además, en la figura 10 podemos observar que las tensiones en entran al Arduino son de entre 0 V a 5 V y la corriente es pequeña, evitando dañar el microcontrolador.

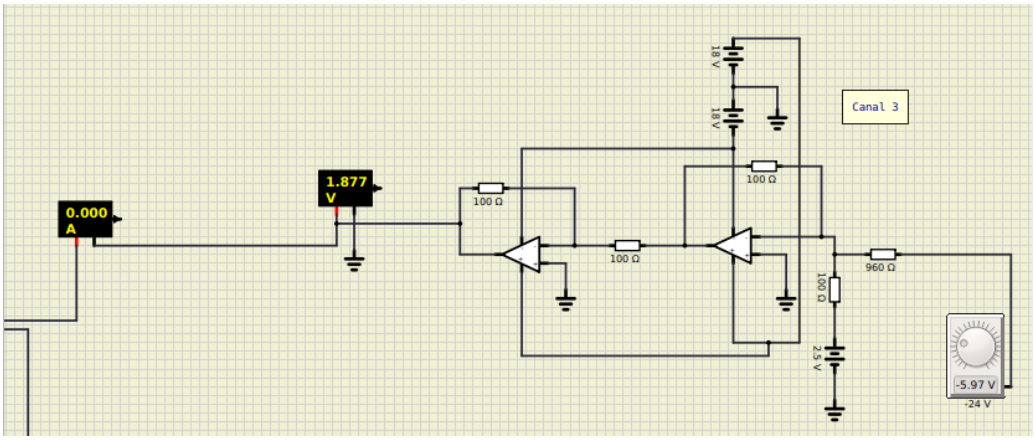


Figura 10: Tensiones y corrientes en DC

## Verificación del funcionamiento en modo AC

Se aplicaron señales sinusoidales de amplitud y frecuencia conocidas a los canales de entrada. El sistema, en modo AC, midió el valor pico y calculó el voltaje eficaz (RMS) usando la relación $V_{rms} = \frac{V_{pico}}{\sqrt{2}}$. La función `leerCanalAC()` implementa esta lógica, asegurando una estimación válida para señales senoidales.

En la Figura 11, se muestra una medición en modo AC donde los voltajes calculados coinciden con los valores teóricos esperados. El sistema distingue correctamente entre señales con y sin sobrevoltaje.

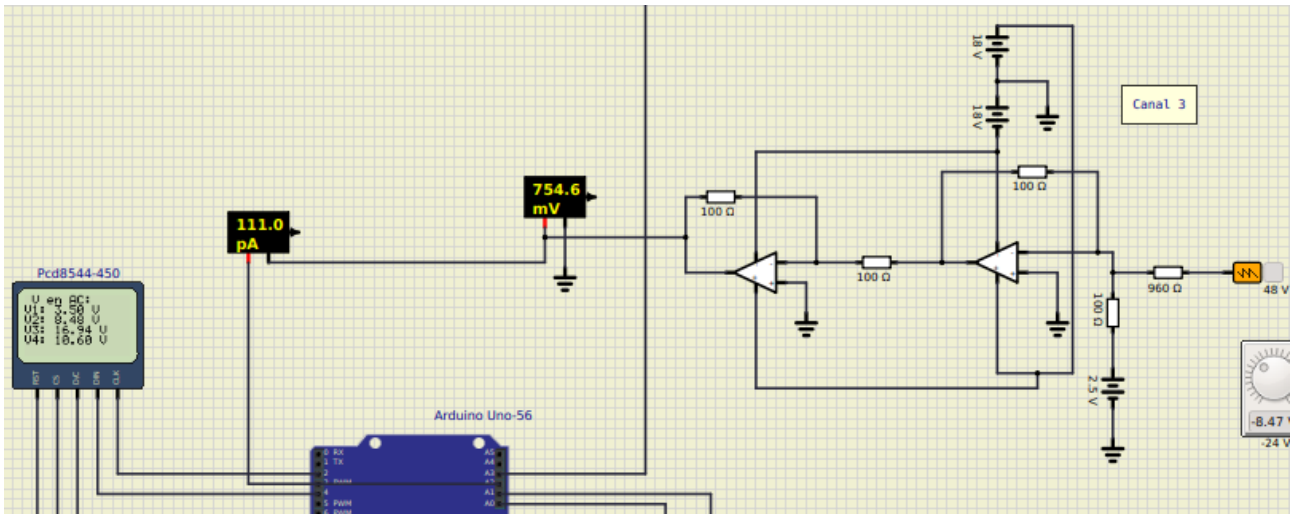


Figura 11: Verificación de la tensión RMS en AC

Además, en la figura 12 podemos observar que las tensiones en entran al Arduino son de entre 0 V a 5 V y la corriente es pequeña, evitando dañar el microcontrolador.

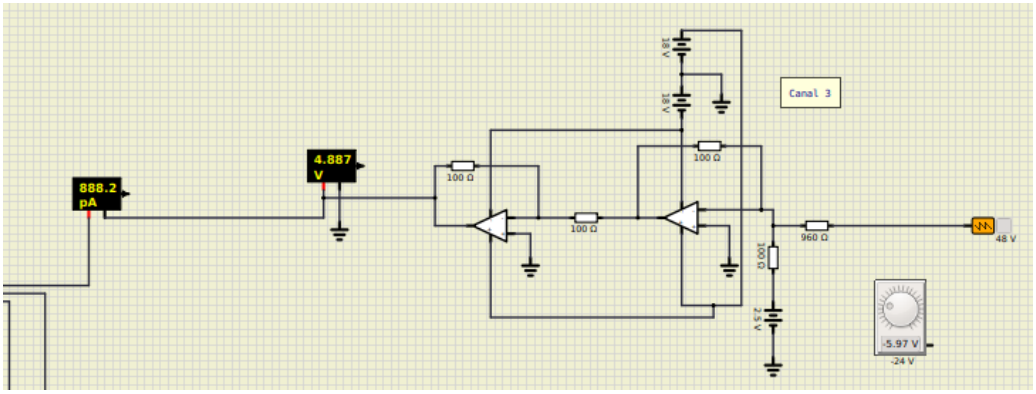


Figura 12: Tensiones y corrientes en AC

## Verificación del sistema de alarmas

El sistema activa un LED indicador cuando uno o más canales presentan un voltaje fuera del rango ±20V. Esta función fue evaluada aplicando señales de prueba entre ±19V y ±21V. El comportamiento observado fue el esperado: el LED solo se encendió cuando alguno de los voltajes superó el umbral permitido.

La Figura 13 muestra el estado del sistema ante una condición de sobrevoltaje. El LED conectado al pin 13 del Arduino se encuentra encendido, indicando la detección de una condición crítica.

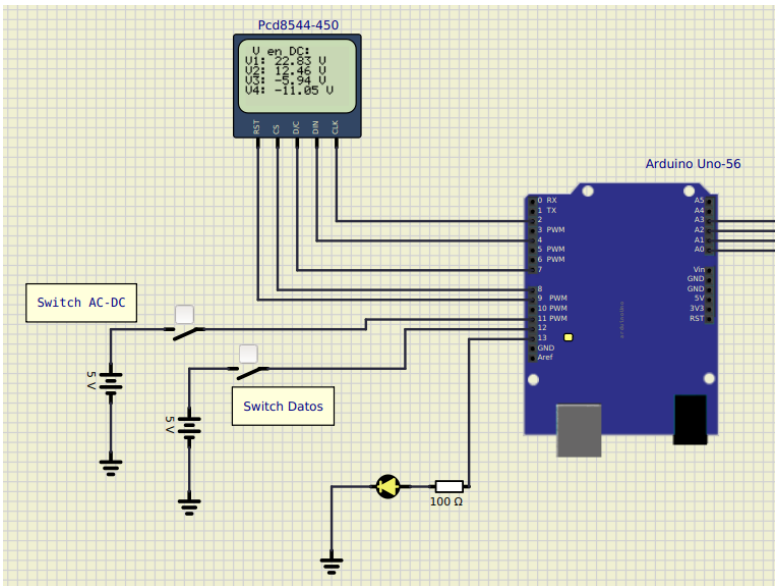


Figura 13: Led de alarma

## 3.3. Verificación de la transmisión serial (USART)

Se verificó que, al activar el switch correspondiente, el sistema envía por el puerto serial los valores leídos en formato legible. El texto transmitido incluye una indicación del modo (AC o DC) y los valores de voltaje para cada canal. Esta salida fue procesada por un script en Python que registro los datos en un archivo CSV.

Las figuras 14 y 15 muestran un ejemplo de la transmisión por el monitor serial. Se observa que los valores enviados coinciden con los mostrados en la pantalla, confirmando la sincronización entre visualización y transmisión. Además, en 16 se observa un guardado correcto de los valores en formato CSV.

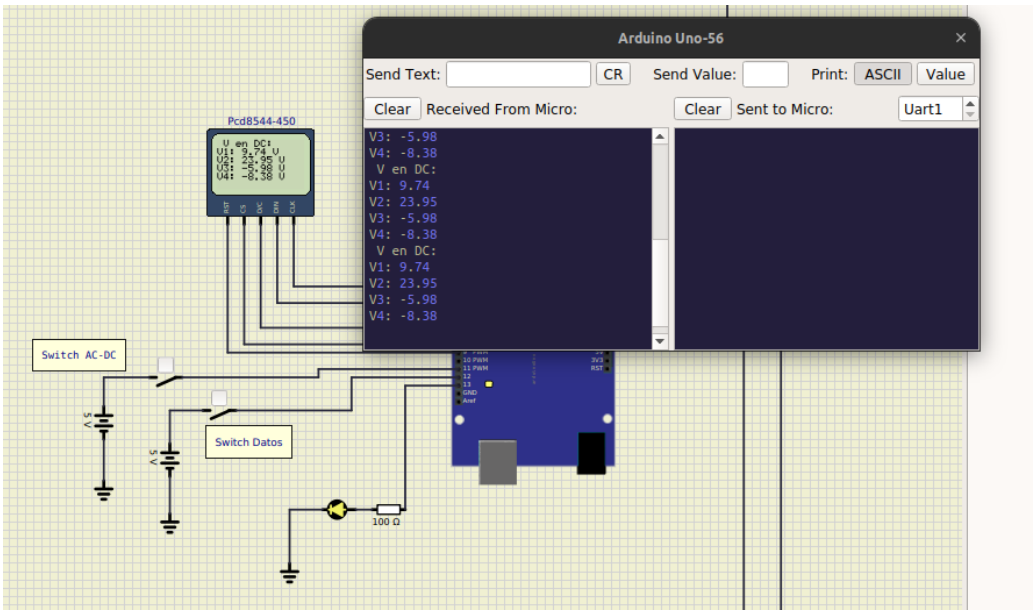


Figura 14: Valores en el monitor del Arduino

```
luis@luis-IdeaPad-3-15IIL05:~/Documents/UCR/Ciclo7/Lab de micro/laboratorio_3$ python3 LecturaPC.py
[2025-05-08 17:17:53] DC | V1=9.74 V2=23.95 V3=-5.98 V4=-8.38
[2025-05-08 17:17:54] DC | V1=9.74 V2=23.95 V3=-5.98 V4=-8.38
[2025-05-08 17:17:55] DC | V1=9.74 V2=23.95 V3=-5.98 V4=-8.38
[2025-05-08 17:17:56] DC | V1=9.74 V2=23.95 V3=-5.98 V4=-8.38
[2025-05-08 17:17:57] DC | V1=9.74 V2=23.95 V3=-5.98 V4=-8.38
[2025-05-08 17:17:58] DC | V1=9.74 V2=23.95 V3=-5.98 V4=-8.38
[2025-05-08 17:17:59] DC | V1=9.74 V2=23.95 V3=-5.98 V4=-8.38
                                                    Ln 1, Col 1   Space
```

Figura 15: Valores en la terminal de Python

laboratorio_3 > Mediciones.csv > data

```
1   Timestamp,Mode,V1,V2,V3,V4
2   2025-05-08 17:17:53,DC,9.74,23.95,-5.98,-8.38
3   2025-05-08 17:17:54,DC,9.74,23.95,-5.98,-8.38
4   2025-05-08 17:17:55,DC,9.74,23.95,-5.98,-8.38
5   2025-05-08 17:17:56,DC,9.74,23.95,-5.98,-8.38
6   2025-05-08 17:17:57,DC,9.74,23.95,-5.98,-8.38
7   2025-05-08 17:17:58,DC,9.74,23.95,-5.98,-8.38
8   2025-05-08 17:17:59,DC,9.74,23.95,-5.98,-8.38
9   2025-05-08 17:18:01,DC,9.74,23.95,-5.98,-8.38
10  2025-05-08 17:18:02,DC,9.74,23.95,-5.98,-8.38
11  2025-05-08 17:18:03,DC,9.74,23.95,-5.98,-8.38
12  2025-05-08 17:18:04,DC,9.74,23.95,-5.98,-8.38
13  2025-05-08 17:18:05,DC,9.74,23.95,-5.98,-8.38
14  2025-05-08 17:18:06,DC,9.74,23.95,-5.98,-8.38
15  2025-05-08 17:18:07,DC,9.74,23.95,-5.98,-8.38
16  2025-05-08 17:18:08,DC,9.74,23.95,-5.98,-8.38
```

Figura 16: Datos guardados en el CSV

# 4. Conclusiones y recomendaciones

### 4.0.1. Conclusiones

- Se logró ejecutar eficazmente la medición de las tensiones generadas, tanto en las tensiones DC como en las RMS en AC, utilizando los pines analógicos del Arduino. Además, es posible observar estas tensiones en una pantalla LCD PCD8544.
- Se logró implementar exitosamente un circuito con amplificadores operacionales que realiza la conversión de un rango de tensiones de -24 V a 24 V a un rango de 0 V a 5 V, permitiendo una lectura más adecuada por el Arduino.
- El laboratorio ha sido capaz de detectar tensiones superiores a un umbral de 20 V, lo que activa el encendido de un LED.
- Se ha logrado una exitosa implementación de la comunicación serial para la conexión entre el microcontrolador y una PC. Además, los datos obtenidos se han almacenado de manera eficiente en un archivo CSV para su posterior análisis.

### 4.0.2. Recomendaciones

- Optimizar la velocidad de medición de las tensiones RMS en señales AC, mejorando el código en Arduino IDE mediante el uso de técnicas como la interrupción de temporizadores o algoritmos de promedio más rápidos.
- Para un análisis más completo, se recomienda que el código Python no solo genere un archivo CSV, sino que también modele y grafique los datos obtenidos, lo que facilitaría la visualización de patrones y tendencias en las mediciones.
- Ampliar el número de tipos de señales AC a medir, incluyendo no solo señales sinusoidales, sino también señales cuadradas y triangulares, para obtener un análisis más completo.

# Referencias

[1] González, Ó. *Guía de modelos Arduino y sus características: Arduino UNO*. BricoGeek Lab. Disponible en: https://lab.bricogeek.com/tutorial/guia-de-modelos-arduino-y-sus-caracteristicas/arduino-uno

[2] Teix, R. *Conhecendo o Esquemático do Arduino Uno – Parte 1*. Ricardo Teix. Disponible en: https://www.ricardoteix.com/conhecendo-o-esquematico-do-arduino-uno-parte-1/

[3] LastMinuteEngineers, *Interface Nokia 5110 Graphic LCD Display with Arduino*, disponible en: https://lastminuteengineers.com/nokia-5110-lcd-arduino-tutorial/

[4] M. Villalta, "Arduino UNO: GPIO, ADC y comunicaciones", Curso IE-0624 Laboratorio de Microcontroladores, Escuela de Ingeniería Eléctrica, Universidad de Costa Rica, I Ciclo 2025.