Universidad de Costa Rica

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

IE-0624 Laboratorio de Microcontroladores

Laboratorio #3

Pablo Durán Segura B62416

Sergio Garino Vargas B73157

Profesor: Marco Villalta Fallas

6 de octubre de 2024

1. Introducción

En este proyecto se diseñará un voltímetro de 4 canales utilizando la placa Arduino UNO, capaz de medir voltajes en el rango de [-24, 24]V, tanto en corriente continua (DC) como en corriente alterna (AC). El sistema será capaz de recibir y visualizar en tiempo real los valores de los 4 canales de entrada en una pantalla LCD PCD8544. Además, se integrará una comunicación entre el programa SimulIDE y la computadora, para registrar los datos medidos en un archivo CSV utilizando el puerto serial.

Para convertir las tensiones de [-24, 24]V, a un rango de 0 a 5V adecuado para el Arduino, se utilizó un amplificador sumador inversor.

2. Nota teórica

2.1. Información general del Arduino

El Arduino Uno es una tarjeta de desarrollo que utiliza un microcontrolador llamado ATmega328P, fabricado por Atmel. El ATmega328P es un microcontrolador RISC de 8 bits con 32kB de Flash, 2kB de SRAM y 1kB EEPROM. Funciona a una velocidad de 16 MHz. Dispone de 23 pines de GPIO, de los cuales 6 pueden generar señales PWM. Estos pines pueden configurarse como entradas o salidas. El ATmega328P cuenta con 6 canales de entrada analógica que pueden ser utilizados para leer sensores analógicos. Estos canales están conectados a un conversor Analógico-Digital de 10 bits, que permite convertir señales de voltaje analógicas en valores digitales. Esto resultará muy útil para la lectura de las tensiones del voltímetro. El ATmega328P también cuenta con interfaces de comunicación como USART, SPI y I2C, además de 3 temporizadores e interrupciones. A continuación se muestra el diagrama de bloques del ATmega328P.

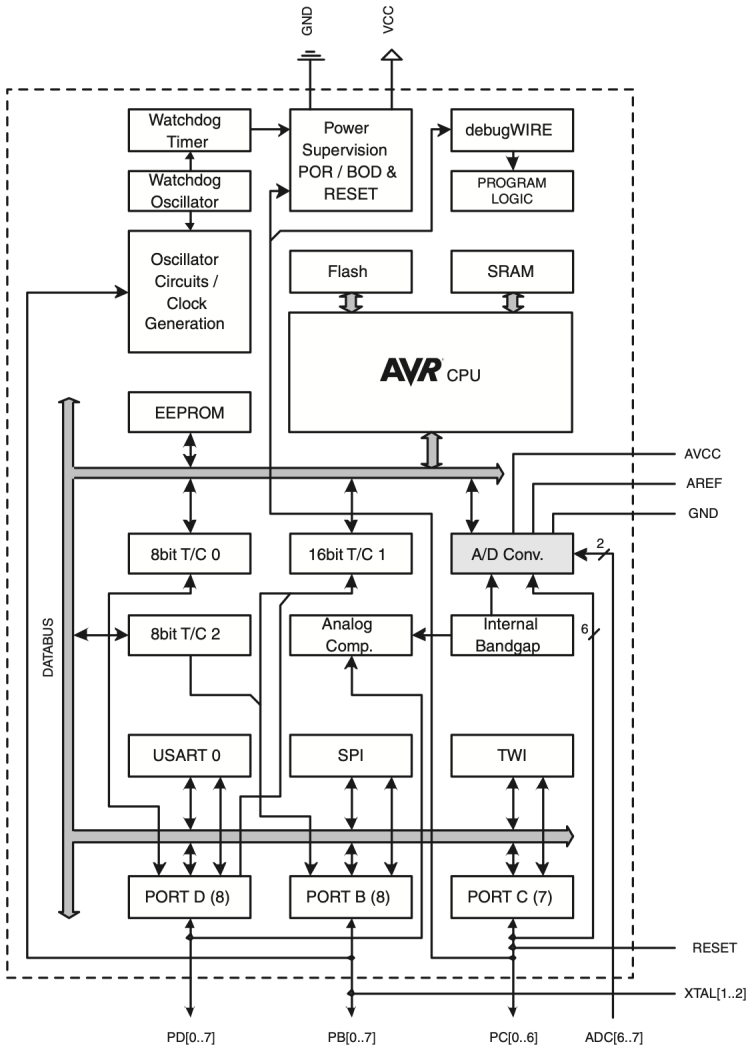

Figura 1: Diagrama de bloques del ATmega328P.[1]

2.2. Periféricos

Para el desarrollo del voltímetro se utilizaron varios periféricos, principalmente para comunicar el Arduino con la pantala LCD.

- Entradas Analógicas (ADC):

  El ATmega328P tiene un Conversor Analógico-Digital (ADC) de 10 bits, es decir, que la señal analógica se divide en $2^{10} = 1024$ niveles, desde 0 a 1023. Por ejemplo, si se va a medir una tensión de 5V, el nivel sería 1023. La conversión de la tensión de la fuente DC al nivel que el Arduino acepta se realizó por medio del software con la siguiente relación.

$$\frac{V_{entrada}}{Valor_{analogico}} = \frac{5V}{1023} \tag{1}$$

Para el desarrollo del voltímetro se utilizaron 5 pines de entrada analógica, los pines A5, A4, A3 y A2 para leer los voltajes de la fuente y el pin. El pin A2 se utilizo como entrada seleccionadora del modo de tensión a leer AC o DC.

- Puertos de Entrada/Salida Digital (GPIO)

  Los pines GPIO (8, 9, 10 y 11) se utilizaron para controlar los LEDs de las alarmas . Se diseñó el circuito de tal manera que, en caso de que los voltajes de entrada superen los ±20V, el sistema encienda un LED de advertencia correspondiente al canal afectado. Esta alarma se implementó mediante los pines digitales del microcontrolador.

- Interfaz SPI (Serial Peripheral Interface)

  La comunicación entre el Arduino y la pantalla LCD PCD8544 se implementó mediante el protocolo de comunicación SPI. La librería Adafruit_PCD8544.h fue incluida en el proyecto para facilitar el control de la pantalla LCD. Esta librería maneja automáticamente la comunicación SPI, permitiendo enviar los datos a la pantalla sin tener que escribir código complejo para manejar el protocolo SPI directamente. Las conexiones de la pantalla al Arduino se realizaron de la siguiente manera: [2]

  El pin 3 se conectó al pin RST (para resetear la pantalla cuando sea necesario). Pin 4 se conectó al pin CS (selección del chip). Pin 5 se conectó al pin DC (indica si se está enviando un comando o datos). Pin 6 del Arduino se conectó al pin DIN de la pantalla (datos de entrada). Pin 7 se conectó al pin CLK (señal de reloj para la comunicación SPI).

  Estas conexiones permiten que la pantalla reciba comandos e información desde el Arduino utilizando la comunicación SPI (Serial Peripheral Interface).

- Puerto Serial USART (Universal Synchronous/Asynchronous Receiver Transmitter)

  El puerto USART (Universal Synchronous/Asynchronous Receiver/Transmitter) es un periférico muy importante en el Arduino. Se utiliza para la comunicación serial. Este tipo de comunicación permite que el microcontrolador envíe y reciba datos un bit a la vez a través de un solo cable, lo que es ideal para conectar dispositivos como computadoras, módulos de comunicación

o periféricos. Para el desarrollo de este proyecto, se simuló la conexión serial creando un enlace virtual entre dos puertos seriales, /tmp/ttyS0 y /tmp/ttyS1, usando el comando socat. De esta forma la computadora puede recibir los datos que son medidos por el microcontrolador en el simulador.

A continuación se muestra el diagrama de pines del ATmega328P.

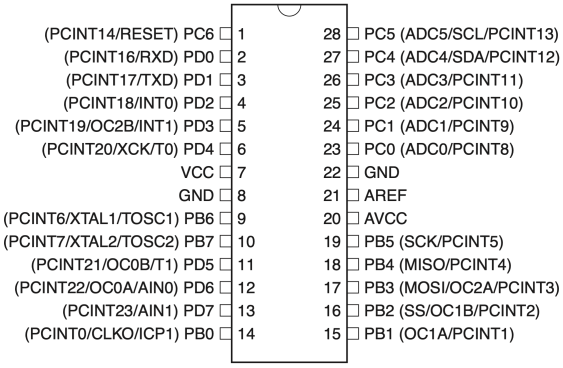

Figura 2: Diagrama de pines del ATmega328P.[1]

2.3. Diseño de circuito

Para adaptar los voltajes de entrada, que se encuentran en el rango de [-24, 24]V, al rango que el Arduino puede manejar (0 a 5V), se diseñó un amplificador sumador inversor. Este circuito nos permite escalar y desplazar los voltajes de entrada para que estén dentro del rango aceptable para el microcontrolador. A continuación se muestra el circuito.

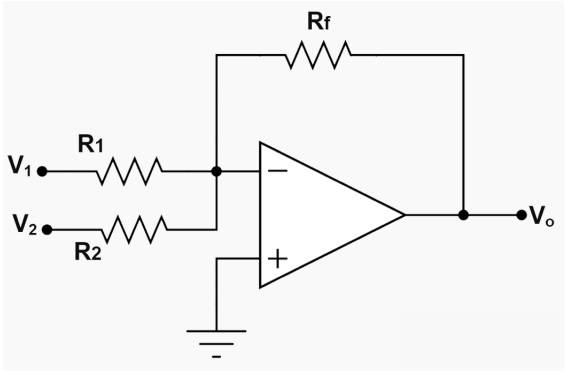

Figura 3: Amplificador sumador inversor.[3]

La tensión de salida de este amplificador está definida por la siguiente ecuación.

$$V_o = -R_f \left(\frac{V_1}{R_1} + \frac{V_2}{R_2} \right) \tag{2}$$

De acuerdo con esta ecuación, se podría elegir algunas de las tensiones V_1 o V_2, empezaremos con V_1 para que sirva como referencia y de esta forma poder mantener la tensión de salida acotada entre 0 y 5V. Lo siguiente es elegir un valor para alguna de las resistencias, en este caso iniciamos eligiendo la resistencia $R_f = 1k\Omega$. Seguidamente se elige la resistencia R_1, también con un valor de $1k\Omega$, para mantener la referencia en 2.5V. La ecuación entonces se convierte en lo siguientes.

$$V_o = -1k\Omega \left(\frac{-2{,}5V}{1k\Omega} + \frac{V_2}{R_2} \right) \tag{3}$$

$$V_o = 2{,}5V - 1k\Omega \cdot \frac{V_2}{R_2} \tag{4}$$

A partir de esta ecuación se debe elegir una resistencia R_2 que permita escalar las tensiones de entrada, que en este caso sería la tensión V_2. Si se escoge una resistencia $R_2 = 10k\Omega$, podemos verificar que las tensión de salida no sobrepasa los 5V. Primero se verifica el caso donde V_2 es -24V.

$$V_o = 2{,}5V - 1k\Omega \cdot \frac{-24V}{10k\Omega} = 4{,}9V \tag{5}$$

Ahora se comprueba la tensión de salida cuando V_2 es 24V.

$$V_o = 2{,}5V - 1k\Omega \cdot \frac{24V}{10k\Omega} = 0{,}1V \tag{6}$$

Como se puede observar, la tensión de entrada de [-24, 24]V queda acotada entre 0 y 5V. Para mapear esto en el microcontrolador se obtuvo una función que luego se codificó en el IDE de Arduino. Esta función se puede obtener sabiendo que tenemos cuatro puntos que corresponden a (4.9V, -24V) y (0.1V, 24V). Al calcular la ecuación de la recta se tiene la siguiente pendiente.

$$m = \frac{24V - (-24V)}{0{,}1V - 4{,}9V} = -10 \tag{7}$$

Luego se calcula el valor de b.

$$-24 = (-10) \cdot 4{,}9 + b \tag{8}$$

$$b = -24 + 49 = 25 \tag{9}$$

La ecuación resultante es:

$$V_{entrada} = -10 \cdot V_{arduino} + 25 \tag{10}$$

Donde $V_{arduino}$ puede ser cualquier valor entre 0.1V y 4.9V.

En la etapa previa a acondicionar la señal que mide el arduino, se diseño utilizó un relay que permite escoger entre una fuente AC y una fuente DC, el relay es alimentado por la misma señal de tensión que llega al Arduino y permite escoger entre la forma de medición AC o DC.

La medición DC consisten en tomar el resultado de la ecuación (10), en cuanto a la medición de la tensión AC el procedimiento varía levemente, ya que si se pretende tomar el mismo resultado la medición va a oscilar entre los valores pico de la señal sinusoidal, por lo que se desea tomar el valor efectivo o valor rms de la señal.

El RMS se una señal en forma matemática representa la raíz cuadrada del promedio de los valores de la señal elevado a la dos, en teoría eléctrica se puede interpretar la una señal de corriente como el valor de corriente directa que disiparía la misma potencia en un resistor. [4]

Para cualquier señal periódica y continua en el tiempo el RMS se calcula con la siguiente fórmula.

$$\mathrm{V_{RMS}} = \sqrt{\frac{1}{\mathrm{T}} \int_0^{\mathrm{T}} \mathrm{v(t)^2 dt}} \tag{11}$$

Donde T representa el periodo de la señal y v(t) es la señal periódica. En nuestro caso, al estar trabajando con un microcontrolador calcular el RMS para una señal discreta, lo cual es aún más sencillo ya que solo hay que tomar una N cantidad de muestras por periodo, elevarlas al cuadrado y sumarlas, ese dato se divide entre el total de muestras y sacándole la raíz cuadrada se obtiene el RMS. Matemáticamente se expresa de la siguiente forma:

$$\mathrm{V_{RMS}} = \sqrt{\frac{1}{\mathrm{N}} \sum_0^{\mathrm{N}} \mathrm{v_i}} \tag{12}$$

Donde N es el número de muestras y v_i el valor de la señal cuando se tomó la muestra.

2.4. Lista de componentes y precios

Componente	**Precio/unidad**	**Cantidad**
Arduino UNO	19200	1
Display PCD8544	5800	1
LED rojo	60	1
LED azul	100	1
LED verde	100	1
LED amarillo	125	1
100 Ω res	25	4
1 kΩ res	70	4
2 kΩ res	70	4
10 kΩ res	70	4
50 kΩ res	70	4
1 F cap	320	6
switch	65	6
Amplificador operacional	400	4
Relay de tiro doble THD-1201L	780	4
Potenciómetro 2 kΩ	350	4
Potenciómetro 50 kΩ	350	4
Total	36435	

Tabla 1: Tabla de componentes

3. Desarrollo y análisis de resultados

Al igual que laboratorios anteriores, en este se adoptó una estrategia de dividir el problema en sus partes e ir paso a paso, lo primero que se hizo fue aprender a utilizar la pantalla, las conexiones se hicieron tal como se indica en la nota teórica, las pruebas iniciales se hicieron imprimiendo palabras

y luego asumiendo que ya se tenía la lectura de la tensión y mostrando el resultado en la pantalla. El código es bastante sencillo ya que con la librería solo se debe indicar la posición del dato y el valor.

```
display.setCursor(0, 0);
display.print("Prueba");
```

El resultado obtenido es;

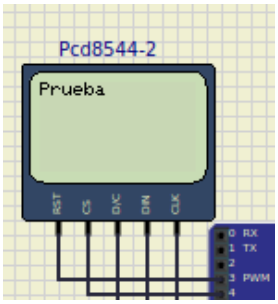

Figura 4: Prueba con la pantalla

Para poner varios datos en la pantalla se procede de forma similar, solo se debe llamar varias veces el método para manipular la pantalla.

```
display.setCursor(0, 0);
display.print("Prueba");
display.setCursor(0, 10);
display.print("Voltaje");
display.setCursor(0, 20);
display.print("Voltaje2");
```

El resultado es:

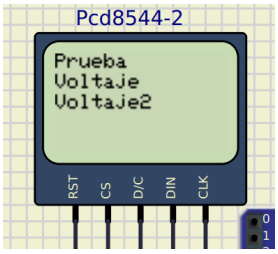

Figura 5: Segunda prueba con la pantalla

Una vez comprendida la forma en la que se utiliza la pantalla se procede con la red DC, la parte para acondicionar la señal se explica en detalle en la nota teórica, para el generador se optó por usar dos fuentes en cada canal, una tensión positiva 0V a 24V y otra de tensión negativa de 0V a -24V. Mediante un switch de tiro doble se puede escoger entre una y otra.

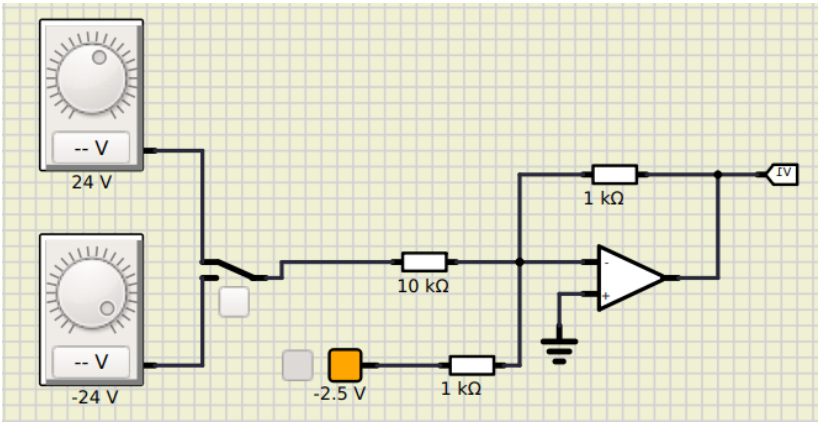

Figura 6: Circuito DC

Ese circuito se utiliza cuatro veces, uno por canal, de manera que la señal a la salida del OPAMP se lleva a la entrada analógica del Arduino que se designó para dicho canal. Lo siguiente fue agregar la parte AC, la forma más sencilla de hacer pasar ambas señales por la etapa que acondiciona la señal fue con un relevador de doble tiro, el cual selecciona no solo la fuente AC o DC sino que se utilizó para indicarle al Arduino la forma en que tiene que medir la señal en su entrada, si es AC calcula el RMS, si es DC solo mantiene el dato.

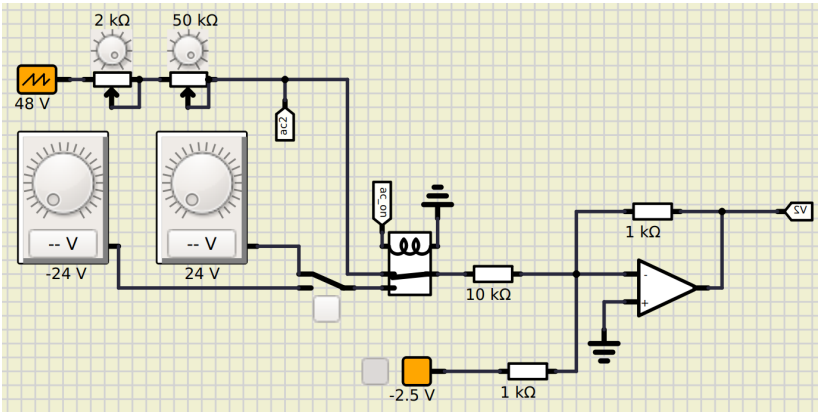

Figura 7: Circuito con ambas fuentes

Simulide no ofrece una fuente de tensión AC a la cual se le pueda variar la amplitud de la señal,

por lo que se optó por tener una fuente fija de 24V pico a pico, la amplitud de la señal se varía usando dos potenciómetros conectados en forma de divisor de tensiones. Usar un solo potenciómetro no es opción ya que si la resistencia máxima es pequeña no se puede disminuir la tensión hasta un valor cercano a 0 y si se usa uno con resistividad grande con variar poco la perilla la tensión cae mucho y se hace imposible regualr el valor deseado, por lo que se usan dos potenciómetros.

Así se ve el circuito del lado del Arduino, la etiqueta ac_on conduce la señal de 5V que llega al relay para seleccionar el modo.

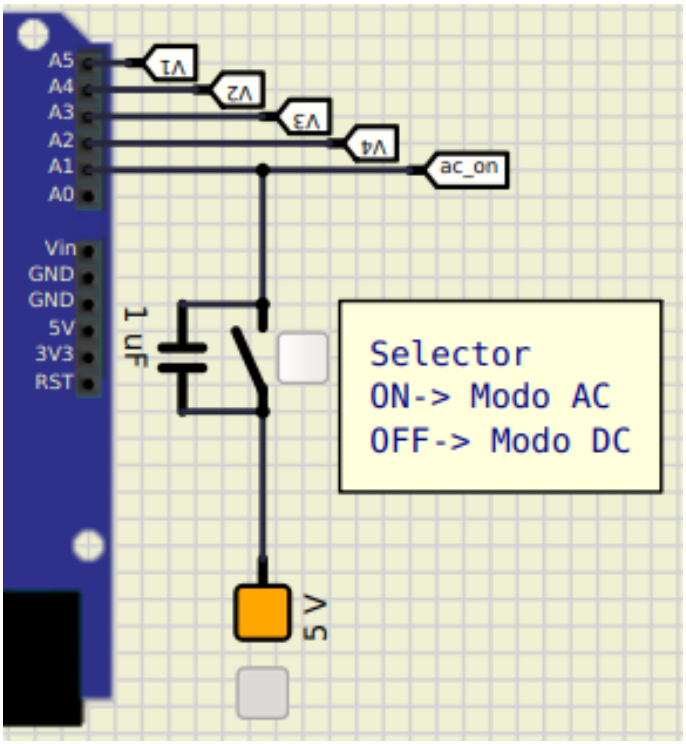

Figura 8: Arduino

En cuanto a la selección del modo en que se miden los datos se usa la señal de 5V, esta se lee por el pin A1, el cual devuelve un dato entre 0 y 1023, de manera que se implementó una función que indica que la medición se hará para AC si se detectan 5V (un dato mayor a 512), y la lectura es DC si se detectan 0V a la entrada (menos de 512). La función devuelve un valor booleano.

```
bool voltageACModeDetector() {
  // Leer el pin
  int voltageMode = analogRead(voltageModePin);
```

```
  // Si la lectura es mayor que la mitad del rango la lectura es AC,
  // caso contrario es DC
  bool voltageIsAC = voltageMode > 512;
  return voltageIsAC;
}
```

Dicha función se usa dentro de otro lógica para así escoger el modo de medición y devolver el dato leído si es DC o calcular el RMS si es AC.

```
  if (voltageACModeDetector()) {
    int samples = 200;
    float sumVoltageSquared;

    for (int i=0; i<samples; i++){
      // Leer el valor analógico (0 a 1023)
      analogValue = analogRead(pin);
      // Convertir el valor analógico (0 a 1023) al voltaje de
      //entrada original (-24V a 24V)
      // Relación: 0 a 1023 -> 0V a 5V (en el Arduino)
      voltage = analogValue * (5.0 / 1023.0);
      // Convertir el rango 0-5V al rango -34V a 34V
      convertedVoltage = (-48/4.79) * voltage + 25.00;
      // Sumatoria de las tensiones al cuadrado
      sumVoltageSquared += convertedVoltage * convertedVoltage;
      // Frecuencia de muestreo
      delayMicroseconds(83);
    }

    // Con 200 muestras se calcula el rms
    rmsVoltage = sqrt(sumVoltageSquared / samples);

    return rmsVoltage;
```

```
} else { // Cuando es DC
  analogValue = analogRead(pin);
  voltage = analogValue * (5.0 / 1023.0);
  convertedVoltage = (-48/4.79) * voltage + 25.00;
  return convertedVoltage;
}
```

Finalmente, se tiene las alarmas que indican si la lectura de tensión es mayor a un valor X, hay un indicar LED para cada canal, tal como se indica en el enunciado, cuando la lectura DC es mayor a 20V se prende el LED de dicho canal. En cuando a la medición AC se optó por disminuir el umbral de alarma a los 14.2V, la razón es simple, cuando se está en modo AC la tensión máxima que se llega a leer es $V_{rms} = \frac{24V}{\sqrt{2}} = 16{,}97V$ de manera que si el umbral se mantiene el indicador no se enciende nunca, de manera que se optó por escalar los 20V pico AC a su valor rms $V_{rms} = \frac{20V}{\sqrt{2}} = 14{,}2V$.

Para activar la comunicación serial entre el Arduino y la computadora simplemente se añadió un interruptor conectado el pin 2 del microcontrolador. En el código se implementa la impresión de los datos en el monitor serial para luego ser enviados a computadora. Para establecer esta conexión primero se debe crear un enlace virtual con el comando $socat PTY, link = /tmp/ttyS0, raw, echo = 0 PTY, link = /tmp/ttyS1, raw, echo = 0$. Luego se debe abrir el puerto Uart en el simulador dándole click al botón Open en el bloque Uart1. Después se puede ejecutar el script de Python que guarda los datos en un archivo .csv, para verificar que los datos fueran correctos, los datos se imprimieron en la terminal.

4. Conclusiones

- Es importante saber cómo manejar los pines analógicos en proyectos donde es necesario medir señales variables, como en este caso, los voltajes de los diferentes canales. Los pines analógicos del Arduino permiten convertir señales continuas, como tensiones eléctricas, en valores digitales que pueden ser procesados.

- El puerto USART puede resultar muy útil si se quieren monitorear datos y almacenarlos en tiempo real. Por medio de python se pueden crear programas que analicen esos datos con algún

fin.

- El uso del amplificador sumador inversor en este proyecto fue fundamental para lograr la conversión de las señales de voltaje en un rango de ±24V a 0-5V, lo cual es necesario para que el Arduino pueda procesar dichas señales sin dañarse. Este tipo de amplificador permite combinar señales de diferentes fuentes, ajustando el voltaje de salida mediante una ganancia que se puede ajustar por medio de resistencias.

- Este proyecto demostró que es posible construir un voltímetro funcional utilizando pocos componentes como amplificadores operacionales, resistencias, relés, etc; y un Arduino, lo que hace que esta solución sea simple, accesible y económica. Con solo un amplificador sumador, algunas resistencias, y la pantalla LCD PCD8544, es factible medir voltajes en un amplio rango, como el caso de ±24V, y mostrar los resultados de manera precisa.

5. Recomendaciones

- Se recomienda mantener el código ordenado, especialmente en la parte de mapear los valores de las tensiones, para mejorar la eficiencia del sistema y reducir posibles retardos en la visualización de los datos.

- Se recomienda utilizar circuitos anti-rebote en los interruptores para evitar lecturas incorrectas en los pines del Arduino. En un circuito con componentes reales esto puede ayudar a que el sistema sea más preciso.

- A la hora de generar datos y enviarlos a algún ya sea para almacenarlos o procesarlos, es recomendable añadir etiquetas o texto que identifique correctamente cada dato. Esto ayuda a que la lectura de los datos sea mas amigable y eficiente.

- Para aumentar las capacidades del voltímetro se podría implementar una función para medir corriente o incluso tensiones más altas. A pesar de que el Arduino solo permite hasta 5V, hay muchos circuitos que capaces de reducir una tensión alta en una tensión baja.

Referencias

[1] M. Technology. megaavr® data sheet. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf

[2] Components101. Nokia5110 lcd. [Online]. Available: https://components101.com/displays/nokia-5110-lcd

[3] S. Ingenieril. Amplificador sumador inversor. [Online]. Available: https://solucioningenieril.com/amplificadores_operacionales/amplificador_sumador_inversor

[4] Wikipedia. Root mean square. [Online]. Available: https://en.wikipedia.org/wiki/Root_mean_square