Laboratorio 3: Arduino: GPIO, ADC y comunicaciones

Mauricio Rodríguez Obando/B96694, Katharina Alfaro Solís/B80251

***Resumen*—En el presente laboratorio se hace uso del Arduino UNO y del microcontrolador que este contiene que es el ATmega328 en su tarjeta de desarrollo, con este se hace un voltímetro de cuatro canales. El circuito se diseñó de manera que puede recibir cuatro tensiones en el rango de [-24, 24] V DC o AC, además se visualizan los valores en una pantalla LCD PCD8544. Para lograr la comunicación con la PC se utiliza la comunicación serial y de esta forma se guardan los datos en un archivo csv.**

I. Nota teórica

I-A. Información del microcontrolador

En este laboratorio se utilizará Arduino UNO, el cual es una placa electrónica de Hardware que tiene como microcontrolador el ATmega328P. Las características generales de este microcontrolador son las siguientes:

- Microcontrolador AVR de 8 bits
- Arquitectura RISC/Harvard
- 4/8/16/64 Kb Flash
- 512b/ 1/ 2k bytes de SRAM
- 256/ 512/ 1k bytes de EEPROM
- 23 GPIOS
- Dos timer/counters de 8 y 16 bits
- Interrupciones
- 8 canales PWM y comparador analógico
- 6 canales 10-bit ADC
- USART
- Maestro/Esclavo SPI (USARTSPI)
- TWI (2-wire), DGB, BTLDR

El diagrama de bloques del microcontrolador es el siguiente:

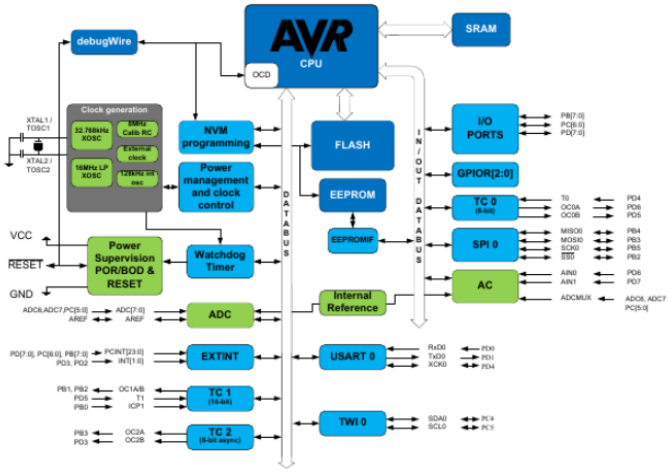

Figura 1. Diagrama de bloques del ATMega328

Para el microcontrolador ATmega328P con 28 pines, el diagrama de pines es el siguiente:

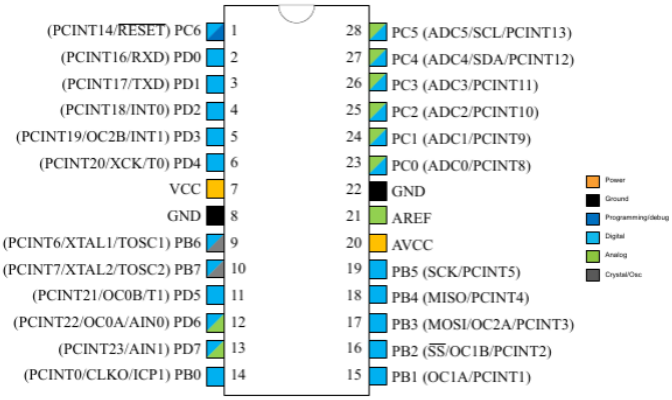

Figura 2. Diagrama de pines del ATMega328

Las caraterísticas eléctricas del microcontrolador son las siguientes de acuerdo a la hoja de datos [1]:

- Temperatura de funcionamiento: -55 °C a +125 °C
- Temperatura de almacenamiento: -65 °C a +150 °C
- Tensión en cualquier Pin excepto RESET con respecto a Tierra: -0.5V a VCC+0.5V
- Tensión en RESET con respecto a Tierra: -0.5V a +13.0V
- Tensión de funcionamiento máximo: 6.0V
- Corriente CC por pin de I/O: 40.0 mA
- Pines VCC y GND de corriente CC: 200.0 mA
- Inyeccción de corriente a VCC = 0V $\pm 5mV$
- Inyeccción de corriente a VCC = 5V $\pm 1mV$

Se va a hacer uso del Arduino UNO para accesar al microcontrolador expuesto anteriormente, por lo que es importante ver el diagrama de pines de este.

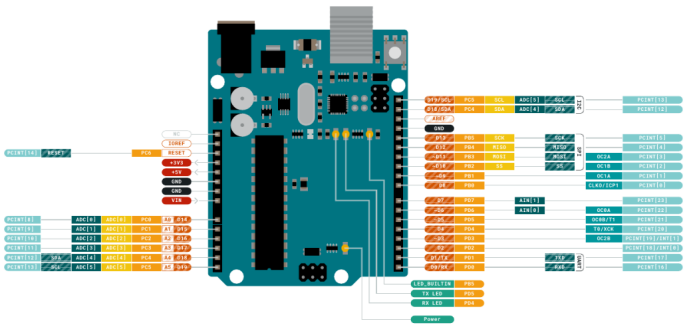

Figura 3. Diagrama de ArduinoUNO

I-B. Componentes electrónicos complementarios

Para la realización del voltímetro se hizo uso del Arduino UNO, adicionalmente de la pantalla PCD8544, estos componentes se explican de mejor forma más abajo en este documento. Además para realizar la medición de la salida de tensión se colocó un osciloscopio. Cabe destacar que adicionalmente se usaron leds, resistores, capacitores y fuentes de tensión como se muestra en la figura 5.

I-C. Diseño del circuito

I-C1. Divisor de tensión: Se utiliza un divisor de tensión con dos fuentes para transformar el rango de tensión de entrada, ya que según las indicaciones el voltímetro puede medir tensiones entre [-24,24]V , sin embargo, los pines del Arduino UNO solo pueden leer entre 0V a 5V. Este divisor de tensión se ve de la siguiente forma como forma de ejemplo:

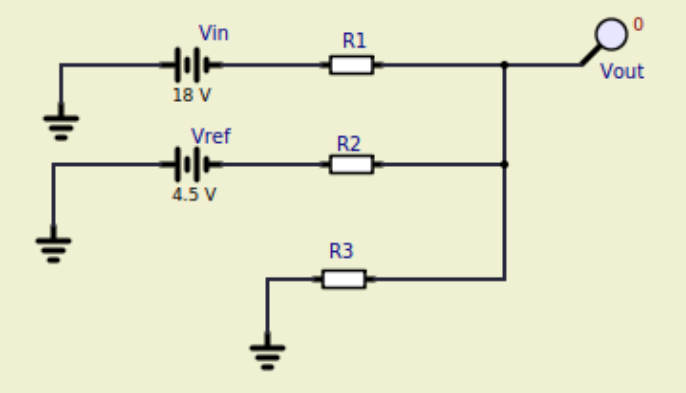

Figura 4. Diagrama del divisor de tensión

Para encontrar una ecuación que modele la salida V_{out} se utiliza el teorema de superposición, de forma que se apaga una fuente independiente primero y luego la otra.

· Apagando la fuente V_{in}:

$$V_{out} = \frac{R_1||R_3}{R_1||R_3 + R_2} \cdot V_{ref} \tag{1}$$

· Apagando la fuente V_{ref}:

$$V_{out} = \frac{R_2||R_3}{R_2||R_3 + R_1} \cdot V_{in} \tag{2}$$

Sumando ambas de las expresiones anteriores:

$$V_{out} = \frac{R_1||R_3}{R_1||R_3 + R_2} \cdot V_{ref} + \frac{R_2||R_3}{R_2||R_3 + R_1} \cdot V_{in} \tag{3}$$

Por factores de diseño, escogemos el valor del resistor $R_1 = 1200\Omega$ y $V_{ref} = 4{,}5V$. Sabemos que la salida debe ser 5V cuando $V_{in} = 24$ y 0V cuando $V_{in} = -24$, entonces:

$$5 = \frac{1200||R_3}{1200||R_3 + R_2} \cdot 4{,}5 + \frac{R_2||R_3}{R_2||R_3 + 1200} \cdot 24 \tag{4}$$

$$0 = \frac{1200||R_3}{1200||R_3 + R_2} \cdot 4{,}5 + \frac{R_2||R_3}{R_2||R_3 + 1200} \cdot -24 \tag{5}$$

Al resolver este sistema de ecuaciones se tiene que: $R_2 = 225\Omega$ y $R_3 = 357{,}62\Omega$

Se buscó los valores anteriores en resistores comerciales con resistencia más cercana a estas, por lo que: $R_2 = 220\Omega$ y $R_3 = 390\Omega$

I-C2. Switches: En este caso se utilizan dos switches, uno para manejar AC/DC y el otro para la transmisión serial o no serial. Al igual que los laboratorios anteriores, se utiliza un filtro RC para controlar el efecto rebote. Por lo que elegimos un capacitor comercial de $10\mu F$ y un resistor de 100Ω.

I-C3. Resistores de protección para los LEDS: Igual es importante recordar que se pone un resistor antes de conectar cada LED para proteger el mismo, en este caso utilizamos resistores de 100Ω.

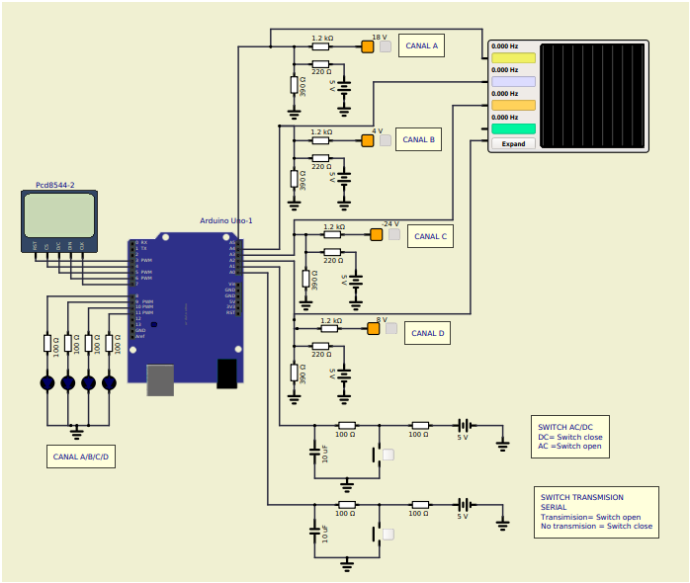

Figura 5. Circuito voltímetro

Cabe resaltar que se utilizó el Arduino UNO, el osciloscopio y la pantalla PCD8544.

I-D. Periféricos utilizados

I-D1. Pantalla LCD PCD8544: La pantalla LCD PCD8544 es una pantalla gráfica monocromática de matriz de puntos de baja resolución fabricada por Philips. Esta pantalla es muy popular en proyectos de electrónica debido a su bajo costo, facilidad de uso y compatibilidad con el microcontrolador Arduino. La pantalla LCD PCD8544 tiene una resolución de 84x48 píxeles y utiliza una interfaz serial para la comunicación con el microcontrolador. [2]

En el código del programa se llama como *Adafruit_PCD8544()*, donde recibe los puertos a los que se conecta esta con el Arduino UNO que son el 7,6,5,4 y 3, como se muestran en la siguiente figura.

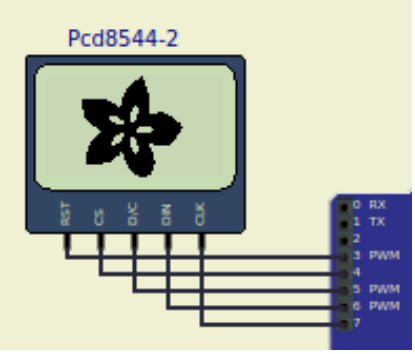

Figura 6. Pantalla LCD PCD8544 en la simulación

En cuanto a las características eléctricas de la pantalla, la pantalla LCD PCD8544 tiene una tensión de alimentación de 2.7 a 3.3V y consume aproximadamente 0.5mA. El pin de acceso de la pantalla es el pin 7, que se utiliza para la transmisión de datos entre la pantalla y el microcontrolador a través de una interfaz serial de 3 pines. También hay pines de control adicionales para la selección de la línea y la columna, así como pines para la alimentación y la conexión a tierra. [2]
I-D2. Funciones manejo de entradas y salidas digitales: Según la guía de Arduino [3] se tiene que:

- pinMode(pin,mode): se utiliza para configurar el modo de un pin digital. El primer argumento es el número del pin, mientras que el segundo argumento es el modo deseado (`INPUT`, `OUTPUT` o `INPUT_PULLUP`). Esta función se utiliza para establecer si un pin se utilizará como entrada o salida digital y configurar la resistencia pull-up interna
- digitalWrite(pin,state): se utiliza para establecer el estado de un pin digital. El primer argumento es el número del pin, mientras que el segundo argumento es el estado deseado (HIGH o LOW). Esta función se utiliza para controlar dispositivos como LEDs, relés y motores, entre otros.
- digitalRead(pin): se utiliza para leer el estado de un pin digital. El argumento es el número del pin que se desea leer. Esta función se utiliza para detectar la presencia o ausencia de señales en dispositivos como sensores, botones y interruptores.
- pulseIn(pin,state,time): se utiliza para medir la duración de un pulso en un pin digital. El primer argumento es el número del pin, mientras que el segundo argumento es el estado deseado (HIGH o LOW) del pulso a medir. El tercer argumento es el tiempo máximo en microsegundos que se esperará para que llegue el pulso antes de que la función devuelva un valor cero.

I-D3. Funciones manejo de entradas analógicas: Según la guía de Arduino [3] se tiene que:

- analogWrite(pin,value): se utiliza para generar una señal de salida analógica en un pin de salida analógica. El primer argumento es el número del pin que se desea escribir. El segundo argumento es el valor de la señal analógica deseada, que debe estar en una escala de 0 a 255.
- analogRead(pin): esta función se utiliza para leer el valor de una señal analógica en un pin de entrada analógica. El argumento es el número del pin que se desea leer. Esta función devuelve un valor entero que representa la amplitud de la señal analógica en una escala de 0 a 1023.
- analogReference(tipo): función para establecer el voltaje de referencia utilizado por el convertidor analógico-digital (ADC) del Arduino UNO. El argumento es el tipo de referencia deseado, que puede ser DEFAULT, INTERNAL o EXTERNAL. La referencia predeterminada es el voltaje de alimentación del Arduino UNO (generalmente 5V), mientras que las otras opciones permiten utilizar una referencia interna (1.1V) o una fuente de referencia externa conectada a través del pin AREF.

I-D4. Funciones para comunicación entre Arduino y PC: Envió de datos

Según la guía de Arduino [3] se tiene que:

- Serial.begin(rate): función para inicializar la comunicación serial entre el Arduino UNO y un dispositivo externo, como una computadora o un módulo de comunicación. El argumento es la velocidad de baudios deseada para la comunicación
- Serial.print(data): función para enviar datos a través del puerto serie del Arduino UNO en formato ASCII legible. El argumento es el dato que se desea enviar, que puede ser una cadena de texto o un valor numérico. Comúnmente utilizada para la depuración y monitoreo de datos en tiempo real.
- Serial.flush(): función para poner en espera la ejecución y esperar a que se completen todas las transmisiones en serie pendientes antes de continuar el programa. Es de gran utilidad para asegurarse de que no se acumulen datos en el buffer de salida.
- Serial.println(data): función similar a Serial.print() pero agrega un carácter de cambio de linea al final de los datos enviados.
- Serial.write(data): función para enviar datos a través del puerto serial en formato binario .
- Serial.end(): función para detener la comunicación en serie y liberar recursos

Recepción de datos

Según la guía de Arduino [3] se tiene que:

- Serial.available(): función para determinar si hay datos disponibles para ser leídos en el buffer de entrada del puerto serial, devolviendo así la cantidad de bytes disponibles para ser leídos.
- Serial.read(): función para leer el siguiente byte disponible en el buffer de entrada del puerto serial, devuelve el valor leído o un -1 si no hay bytes disponibles
- Serial.peek(): función para leer el siguiente byte disponible en el buffer de entrada del puerto serial sin eliminarlo de este nada mas devolviendo su valor.
- Serial.find(): función para buscar un patrón especifico de bytes en el buffer de entrada del puerto serial. Devuelve true o false.
- Serial.readBytes(): función para leer una cantidad específica de bytes del buffer de entrada del puerto serie y almacenarlos en un arreglo de bytes. El primer argumento es el arreglo de bytes en el que se almacenarán los datos leídos, y el segundo argumento es el número de bytes que se desea leer.

I-D5. Protocolos de comunicaciones:

- SPI: de sus siglas en ingles Serial Peripheral Interface, es un protocolo de comunicación en serie sincrónico utilizado para interconectar dispositivos electrónicos digitales. Se utiliza para la transferencia de datos entre un microcontrolador y dispositivos periféricos, como sensores, pantallas, convertidores analógico-digital, entre otros. [4]
  Se basa en una conexión punto a punto entre el microcontrolador y los dispositivos periféricos, y utiliza un bus de 4 señales: SCLK (reloj), MOSI (datos de salida del maestro), MISO (datos de entrada del maestro) y SS (señal de selección de esclavo). El maestro controla la velocidad de transferencia de datos a través de la señal de reloj, y utiliza la señal de selección de esclavo para comunicarse con uno de los dispositivos periféricos conectados. La transmisión de datos se realiza en ambos sentidos, ya que el dispositivo periférico también puede enviar datos al microcontrolador. [4]
- USART: de sus siglas en ingles (Universal Synchronous/Asynchronous Receiver/Transmitter) es un protocolo de comunicación utilizado en sistemas de comunicación serie síncrona o asíncrona, que permite la transmisión de datos en ambas direcciones a través de un solo canal de comunicación. Es utilizado en sistemas embebidos para la comunicación entre microcontroladores, periféricos y dispositivos externos. Soporta velocidades de transmisión de datos desde pocos bits hasta megabits y también permite la configuración de parámetros como la longitud de palabra, la paridad y el control de flujo, para adaptarse a las necesidades específicas de la aplicación. [5]

I-E. Lista de componentes y costo

Cuadro I
COSTO PROMEDIO DE LOS COMPONENTES UTILIZADOS EN EL DISEÑO

Componente	Cantidad	Costo por unidad promedio
Arduino UNO	1	$28.5
Resistencia de 1.2KΩ	4	$6
Resistencia de 220Ω	4	$5.3
Resistencia de 390Ω	4	$8
Capacitor de 10uF	2	$0.20
LED sencillo	4	$0.40
Pantalla PCD8544	1	$8.95
Switch	2	$1.00
Total	$58.35	

II. DESARROLLO/ANÁLISIS

Para la realización del voltímetro, se realizó un diagrama de flujo del voltímetro para entender mejor su funcionamiento y facilitar la programación del mismo.

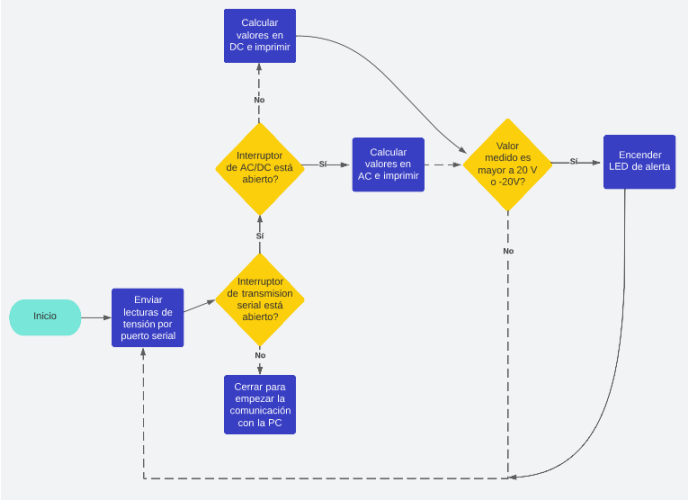

Figura 7. Diagrama de flujo voltímetro

Para el desarrolló de este voltímetro de 24V a -24 V se realizó como fue mencionado anteriormente por divisor de tensión reducir la misma a un rango de 0V a 5V, por lo tanto es importante mencionar que para recuperar el valor real de nuevo y que sea desplegado de forma correcta en la pantalla LCD, se utilizó un factor de conversión de 9,6. Utilizando la siguiente ecuación:

$$Valor_{Real} = (((Valor_{Leido}/1023) * 9,6) - 24) \quad (6)$$

De manera que pasamos de un rango [0, 5] V a [0,48] V, el factor de conversión escala la señal recibida (el valor leído) y por lo tanto se le resta 24 V para lograr obtener de nuevo el rango de [-24, 24] V en la salida, visto desde la pantalla LCD.

Con respecto a los resultados obtenidos se tiene que dentro del SimulIDE al correr el circuito diseñado se obtiene que para

ambos funcionamientos cuando esta el switch de transmisión esta abierto se observa lo siguiente:

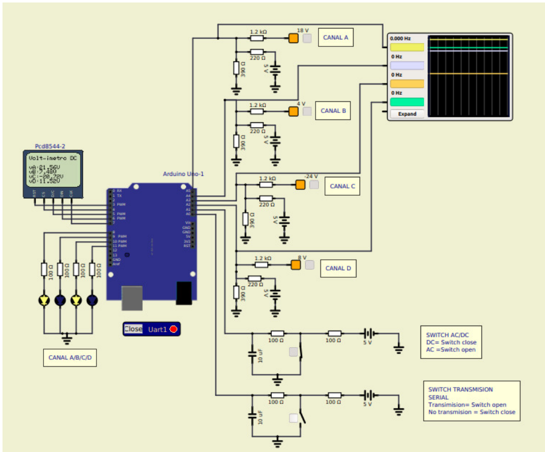

Figura 8. Modo dc funcionando

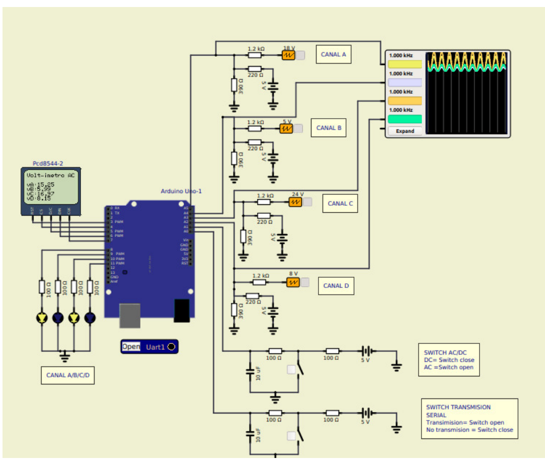

Figura 9. Modo ac funcionando

De donde se puede verificar que para ambos funcionamientos AC/DC se logra medir el voltaje de los 4 canales a la vez, condicionar los voltajes de entrada de modo que se pueden interpretar a el rango de valores del Arduino, el funcionamiento del switch para configurar el modo de medición, el encendido de los leds de protección en donde para casos como los del CANAL A y C se observa que al estar midiendo valores superior a 20V e inferior a -20 y se muestra el funcionamiento de la pantalla LCD en donde se observa que se indica el tipo de medición realizada y el nombre de cada canal con el valor medido respectivamente. También se puede observar que en el modo AC, no solo cambian a señales AC como entradas tal y como se observa en el osciloscopio, sucede también que aunque no se este estrictamente sobre el rango de precaución, como es el caso del CANAL A y C en donde se están midiendo 15V y 16V respectivamente, se mantienen encendidos los leds al igual que en el modo DC por su conversión a rms.

Revisando por otra parte tanto los resultados en terminal como del arhchivo .csv se tiene que:

```
------------ AC/DC:  DC ------------
CHANNEL A:
21.56
CHANNEL B:
7.48
CHANNEL C:
-20.72
CHANNEL D:
11.52
------------ AC/DC:  AC ------------
CHANNEL A:
15.25
CHANNEL B:
5.29
CHANNEL C:
-14.65
CHANNEL D:
8.15
```

Figura 10. Resultados en terminal

----------- AC/DC: AC -----------	
CHANNEL A:	15.25
CHANNEL B:	5.29
CHANNEL C:	-14.65
CHANNEL D:	8.15
----------- AC/DC: AC -----------	
CHANNEL A:	15.25
CHANNEL B:	5.29
CHANNEL C:	-14.65
CHANNEL D:	8.15
----------- AC/DC: AC -----------	
CHANNEL A:	15.25
CHANNEL B:	5.29
CHANNEL C:	-14.65
CHANNEL D:	8.15
----------- AC/DC: DC -----------	
CHANNEL A:	21.56
CHANNEL B:	7.48
CHANNEL C:	-20.72
CHANNEL D:	11.52
----------- AC/DC: DC -----------	
CHANNEL A:	21.56
CHANNEL B:	7.48
CHANNEL C:	-20.72
CHANNEL D:	11.52
----------- AC/DC: DC -----------	
CHANNEL A:	21.56
CHANNEL B:	7.48
CHANNEL C:	-20.72
CHANNEL D:	11.52
----------- AC/DC: DC -----------	
CHANNEL A:	21.56
CHANNEL B:	7.48
CHANNEL C:	-20.72
CHANNEL D:	11.52

Figura 11. Resultado del csv

De las imágenes anteriores se verifica el funcionamiento de la comunicación serial (USART) entre el arduino y la PC, de modo que se esta realizando el envió de datos por el puerto serial y se puede ver en terminal, donde además con estos datos se crea a la vez un .csv en donde se registran el tipo de medición que se realizó y los valores medidos en los 4 canales.

III. Conclusiones

- Por medio de puertos virtuales se logra establecer una conexión entre el Arduino y el PC, eliminando así la limitación que puede traer no tener este dispositivo en físico y estar utilizándolo desde el SimulIDE. Por lo que es una buena aproximación a lo que seria trabajar con el dispositivo real e incluso puede servir como medio de prueba antes de comenzar una implementación física.

- Gracias a los protocolos de comunicación estudiados se implementa una comunicación serial de la cual mediante un archivo .py se logra observar los resultados registrados en forma de texto por la terminal y como un .csv.
- Se logró diseñar utilizando el Arduino UNO un sistema funcional y útil para medir múltiples voltajes, mantener un registro de ellos y con un sistema de leds de precaución.
- La pantalla LCD PCD8544 es útil para el display de imágenes o textos simples, pero por su características en la actualidad tiene poca aplicación por sus limitaciones.
- A la hora de trabajar con el Arduino es importante mantener la estructura de un sketch bien definida par evitar problemas en el código y mantener el orden.

IV. Link del repositorio de GitHub

Link al repositorio de github:
https://github.com/kathalsol/Laboratorio_Microcontroladores/tree/main/Labo3

Referencias

[1] *8-Bit, ATmega328P ATtiny4313*, Atmel, 2009. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/download/313560/ATMEL/ATmega328P.html

[2] *48 x 84 pixels matrix LCD controller/driver, PCD8544*, NXP Semicpnductors, 2004. [Online]. Available: https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf

[3] *Arduino Uno - Technical Specs*, Arduino, 2023. [Online]. Available: https://store.arduino.cc/arduino-uno-rev3

[4] T. Instruments, "Understanding the spi bus," Tomado de https://www.ti.com/lit/an/slaa068/slaa068.pdf, Disponible en línea, accesado el 05/04/23.

[5] A. Corporation, "Usart universal synchronous/asynchronous receiver/transmitter," Tomado de https://ww1.microchip.com/downloads/en/devicedoc/Atmel-42608-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf, Disponible en línea, accesado el 05/04/23.