

Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

IE-0624: Laboratorio de Microcontroladores

Reporte de laboratorio #3: Arduino: GPIO,ADC y  
comunicaciones

Jose Mario Navarro Bejarano

B75398

II-2024

## Contenido

|  |    |
|--|----|
| Introducción.....                            | 3  |
| Nota teórica.....                            | 4  |
| Información del Arduino .....                | 4  |
| Desarrollo .....                             | 5  |
| Circuito implementado .....                  | 5  |
| Procesamiento de las tensiones medidas. .... | 6  |
| Código implementado .....                    | 7  |
| Muestra de funcionamiento .....              | 9  |
| Envío de datos a archivo CSV.....            | 11 |
| Conclusiones y recomendaciones.....          | 13 |

# Introducción

Los Arduinos son tarjetas de desarrollo ampliamente utilizadas por su sencillez y el nivel de abstracción que contiene, lo que permite que cualquier persona con apenas conocer sobre programación, pueda realizar proyectos directamente programados sobre un microcontrolador sin la necesidad de modificar registros, timers, interrupciones o alguna otra de las características más complejas que un dispositivo de este tipo puede poseer. Por ello en este laboratorio se desarrolla un voltímetro de 4 canales, con capacidad de medir tensión tanto en DC como en AC, además de mostrar un mensaje de advertencia cuando la tensión aumenta los 20 V en DC o AC (para la tensión pico). Todo esto haciendo uso de funciones como `serial.print()` o `analogRead()`, funciones que permiten una manipulación mas sencilla de las señales.

En este informe se muestran las características de el dispositivo desarrollado, la forma en la que se construyó y además la forma en la que diseñó el programa para poder interpretar las señales de tensión medidas.

El repositorio con los archivos base se puede encontrar en:  
[https://github.com/marionabe/Lab\\_micros](https://github.com/marionabe/Lab_micros)

# Nota teórica

## Información del Arduino

El Arduino UNO es una tarjeta de desarrollo que incorpora un microcontrolador ATmega 328P. Además, la tarjeta incluye un oscilador de cristal de 16MHz, 14 pines digitales y 6 entradas analógicas.

El Arduino Uno incluye también un controlador ADC de 10 bits, lo que permite cuantizar las entradas de tensión de entre 0V y 5V en valores de 0 a 1023.

En la figura 1 se muestra un diagrama de todos los pines del Arduino uno y del microcontrolador que posee.

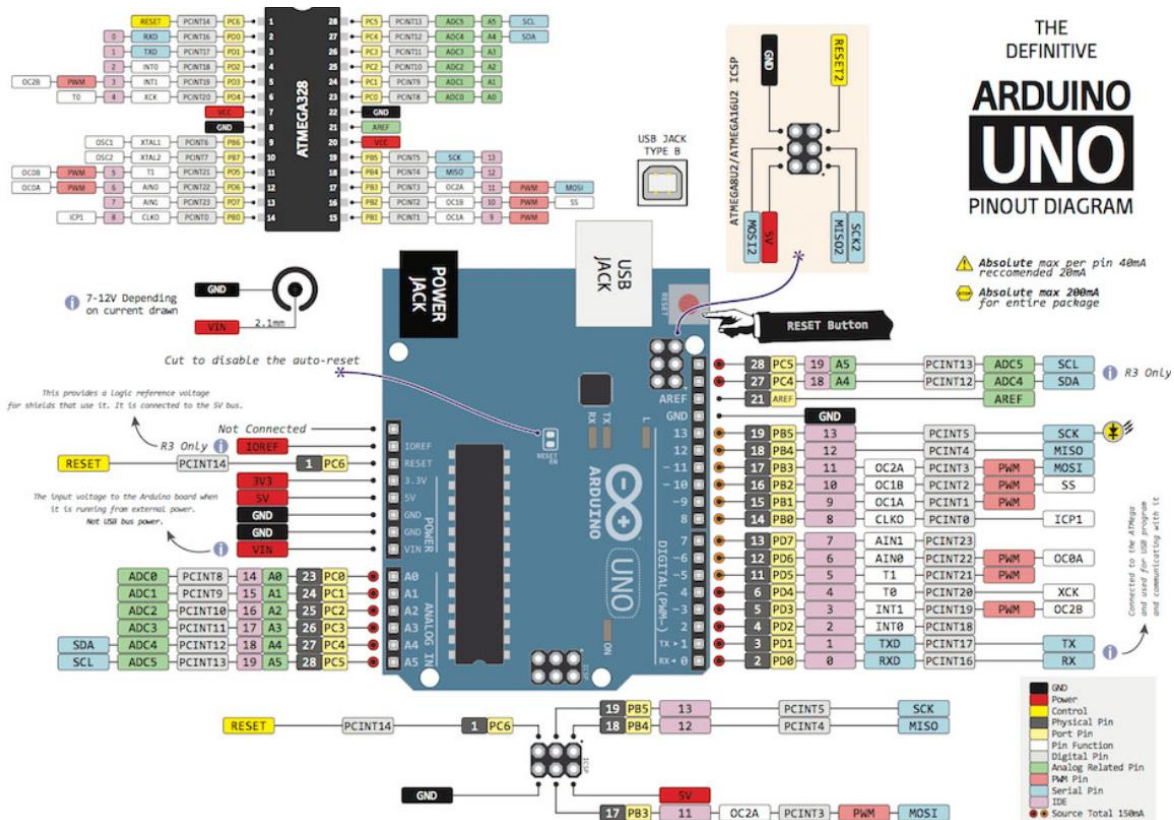


Figura 1: Diagrama de pines y funciones del Arduino Uno.

# Desarrollo

## Circuito implementado

Para desarrollar la implementación del circuito del voltímetro, lo que se hizo fue realizar dos etapas de divisores de tensión. Una primera etapa permite reducir la tensión de entrada o de medición en un factor de 5, lo que resulta en una tensión de entre -5V a 5V aproximadamente. Esta tensión se suma con una fuente de 5V a modo de off-set, lo que permite tener los valores entre 0V y 10V, luego de lo cual se aplica un segundo divisor de tensión que divide la tensión entre 2, lo que permite a la salida de esta etapa tener tensiones de entre 0V y 5V. A continuación, se muestra una fotografía del circuito implementado, en donde se han señalado las secciones más importantes, las cuales se describen a continuación.

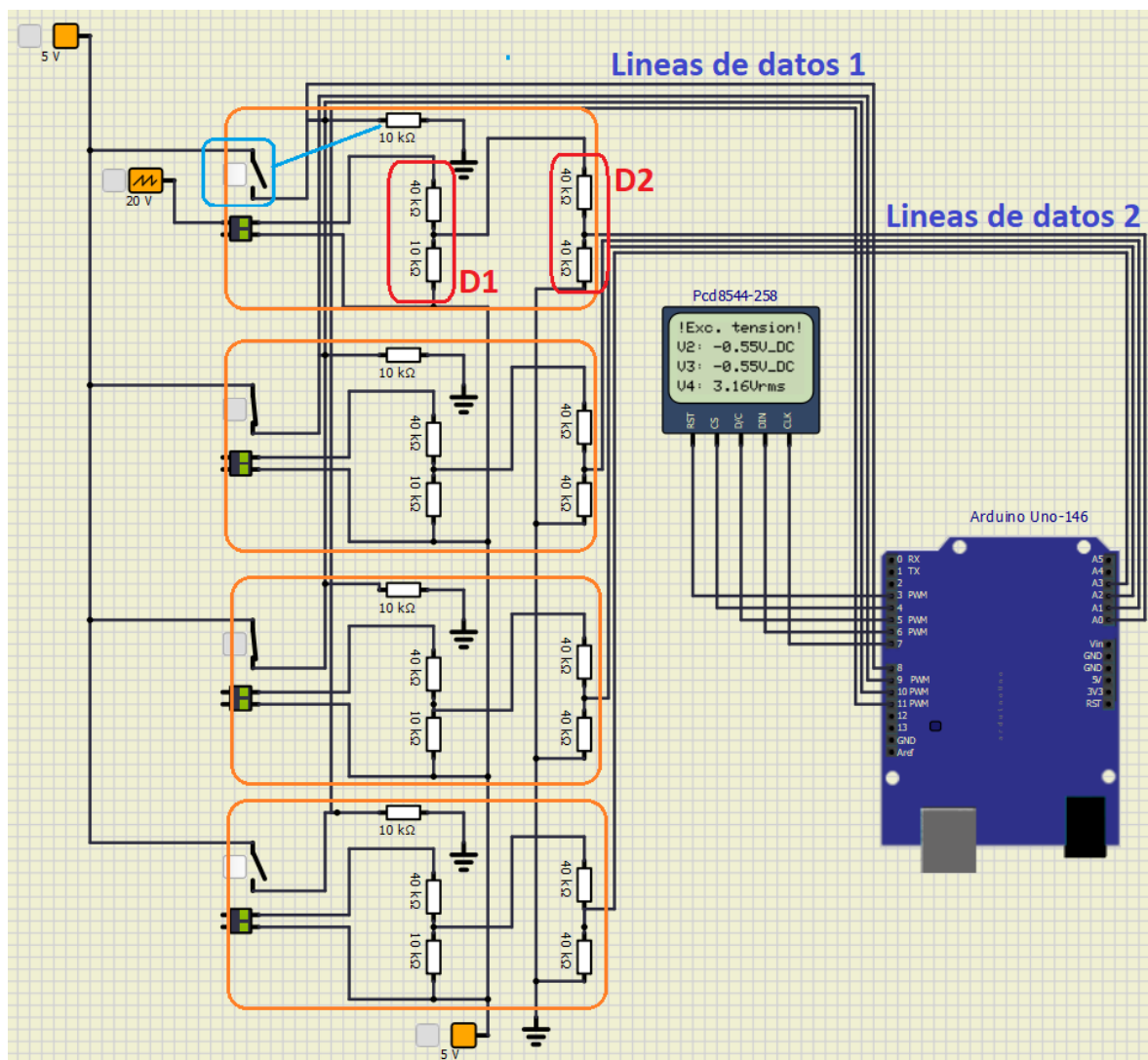


Figura 2: Diagrama del circuito implementado.

La descripción del circuito anterior es la siguiente:

- En los rectángulos anaranjados se muestran los cuatro bloques de cada voltímetro. Todos estos rectángulos son similares, por ello solo se detalló el primer voltímetro.
- En dos bloques rojos se muestran los dos divisores de tensión. Marcado con D1 se señala el primer divisor, que reduce la tensión en aproximadamente un factor de 5. Indicado con D2 se señala el segundo divisor, que se encarga de volver a dividir la tensión (que ahora está entre 0V y 10V) en la mitad, para adaptarla a la tensión que puede medir el pin analógico del Arduino.
- Señalado con un rectángulo de color celeste se muestra el interruptor que se usa para seleccionar si la tensión medida es en DC o AC, esto permite enviar una señal de información al Arduino, por medio de los cables marcados como “Líneas de Datos 1”, para decidir cómo tratar los datos. Esto se profundizará más cuando se explique el código implementado.
- La sección marcada como “Líneas de Datos 2” corresponde a los cables que se envían al Arduino con la señal de tensión medida en cada voltímetro.
- También se muestra la pantalla utilizada, en la cual se despliega información sobre las 4 tensiones medidas, y en caso de que la tensión medida sea superior a 20 V, se muestra un mensaje de alerta.

## Procesamiento de las tensiones medidas.

En los pines analógicos del Arduino, se realiza la medición de la tensión luego de la etapa de ajuste de la tensión. Aunque sería posible con base en los valores de las resistencias utilizadas poder calcular directamente la tensión medida con base en la tensión que recibe el Arduino, en la práctica esto generó algunos errores, ya que, debido a posible imperfecciones o valores no contemplados, resultaba un poco difícil estimar la tensión solo con base en estos valores. Por ello, se hizo uso de una hoja de datos de Excel, en la cual se realizó un escribió el valor cuantizado para cada valor de tensión aplicado en las terminales del voltímetro. Esto se realizó para los 49 valores entre -24V y 24V. Con estos datos y haciendo uso de la herramienta “curva de mejor ajuste” se pudo obtener un par de ecuaciones que permiten estimar el valor de tensión en la entrada del circuito con base en el valor cuantizado. Ambas ecuaciones se muestran a continuación:

$$V_{DC\_estimada} = 0,0537 * (valor\_cuantizado) - 24,926$$

$$V_{AC\_estimada} = 0,0538 * (valor\_cuantizado) - 9,9656$$

Esta técnica también podría permitir, en una implementación real, obtener una ecuación que tome en cuenta las variaciones en los valores de las resistencias comerciales, así como comportamientos no lineales.

## Código implementado

A continuación, se detalla el código utilizado para realizar este voltímetro. Para cada voltímetro se utilizó una copia del mismo código, por lo que igualmente se muestra el código correspondiente solo al voltímetro 1.

```
127 //Decidir si la tensión que se tiene es DC o AC
128 if (dc_2_en==1){
129     //Calcular tensión con base en la fórmula obtenida
130     valorMultimetro2=(0.0537*(valorMultimetro2)-24.926);
131
132     //Desplegar info DC
133     display.setCursor(0,1*espacioReglones);
134     texto = "V2: "+String(valorMultimetro2)+ "V_DC";
135     display.println(texto);
136
137 } else {
138     //Reiniciar el medidor AC
139     valorMultimetro2=0;
140     // Medir varias veces para encontrar el valor máximo
141     for (int i = 0; i<100; i++){
142         temp=analogRead(A1);
143         if (temp > valorMultimetro2){
144             valorMultimetro2 = temp;
145         }
146     }
147     //Calcular la tensión AC pico con base en la fórmula
148     valorMultimetro2=0.0538*valorMultimetro2-9.9656;
149     //Calcular valor rms de la tensión AC
150     valorMultimetro2=0.707*valorMultimetro2-7.065;
151
152     //Desplegar info AC
153     display.setCursor(0,1*espacioReglones);
154     texto = "V2: "+String(valorMultimetro2)+ "Vrms";
155     display.println(texto);
156 }
```

Figura 3: Primera sección del código del voltímetro.

Descripción del código:

- En la línea 128 se crea un condicional IF que decide si los datos medidos deben ser tratados como en tensión AC o DC.
- En la línea 130, con base en el valor cuantizado, se calcula la tensión que se está midiendo en la entrada del circuito, haciendo uso de la ecuación antes calculada.
- En las líneas 132-135 se despliega la información en pantalla sobre la tensión medida. Para realizar esto se utiliza una biblioteca que permite simplemente indicar donde se desea posicionar el cursor, y escribir texto en esa posición.
- En la línea 137 en adelante se trata el valor cuantizado para una tensión AC.

- En esta sección, primero se reinicia el valor cuantizado, luego se pasa por un bucle en donde se obtiene la mayor tensión medida, es decir, el valor pico de la tensión de entrada. Luego de esto, se utiliza la formula antes encontrada para calcular la tensión pico AC medida, con la cual luego se obtiene el valor Vrms y luego se agrega un pequeño offset que se obtuvo con base en la prueba y error, ya que los valores de tensión medidos no correspondían con la tensión. Estas incongruencias podrían deberse al hecho de no poder separar las tierras de la etapa de medición, por lo que se generan tensiones indeseadas.
- Finalmente, la tensión se imprime, junto a un mensaje indicando que se está midiendo tensión AC.

```

158 //Imprimir mensaje de peligro por exceso de tensión
159 if ((valorMultimetro2>20 || valorMultimetro2 < -20) && dc_2_en){
160     delay(20);
161     display.clearDisplay();
162     texto = "!Exc. tension!";
163     display.println(texto);
164     delay(4);
165 } else if((valorMultimetro2>14.14) && dc_2_en==0){
166     delay(20);
167     display.clearDisplay();
168     texto = "!Exc. tension!";
169     display.println(texto);
170     delay(4);
171 }

```

Figura 4: Segunda sección del código del voltímetro.

En esta sección del código, lo que se hace es imprimir mensajes de advertencia en caso de que la tensión sea menor a -20V o mayor a 20V para DC, o que supere los 14.14Vrms para tensión AC (que corresponde a 20V pico). En este punto es importante mencionar que el generador de ondas sinusoidales del simulador genera las tensiones sinusoidales a partir de 0, es decir, no las genera de forma simétrica con respecto a 0V, por lo que el comportamiento del voltímetro se calibró con base en este comportamiento. Aunque era posible modificar el generador para aplicar un off-set que centrara la señal, esto resultaba un poco impráctico ya que era necesario volver a modificarlo cada vez que se cambiaba la tensión.



## Muestra de funcionamiento

En la siguiente figura se muestra el funcionamiento de todos los voltímetros para diferentes situaciones según se solicitó en el enunciado.

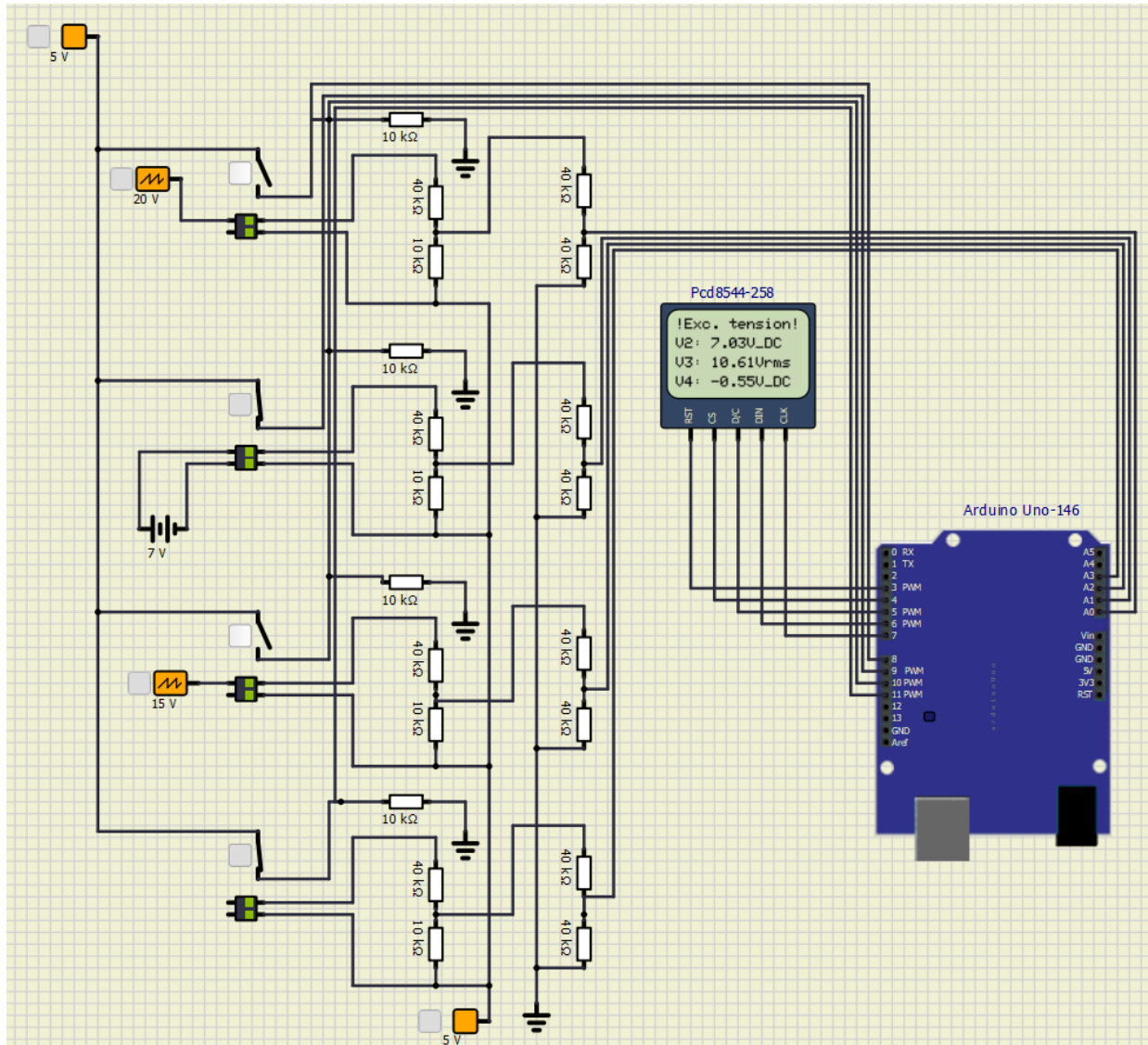


Figura 5: Muestra del funcionamiento del voltímetro

En la imagen anterior se puede observar lo siguiente:

- Para el primer voltímetro (el de la parte superior) se establece una señal sinusoidal con una frecuencia de 60 Hz y una tensión de 20V. Se puede observar que en la pantalla se muestra un mensaje de alerta indicando exceso de tensión. Mientras se mantenga este nivel de tensión la pantalla intercambiará entre el mensaje de advertencia y los datos de la medición.
- En el voltímetro 2 se colocó una batería con una tensión de 7V. Se puede ver en la pantalla que se muestra la tensión medida con un pequeño margen de error, además

de indicar que se está midiendo de DC. Es importante observar que para este caso el switch se cerró, para indicarle al Arduino que se va a medir tensión DC.

- En el voltímetro 3, se aplicó una tensión AC, pero por debajo del umbral de peligro, por lo que se puede ver que en la pantalla se muestra el valor  $V_{rms}$  de la tensión correspondiente.
- Finalmente, el voltímetro 4 se dejó sin tensión, y se puede observar que en la pantalla se muestra una tensión diferente a 0, correspondiente a un pequeño margen de error de la medición, pero que se puede aproximar como 0V. Es importante mencionar que esto solo sucede cuando se deja la tensión “flotando” ya que, si se agrega una tensión muy pequeña, como 0.10V, el voltímetro igualmente es capaz de medirla, tal como se muestra en la siguiente figura.

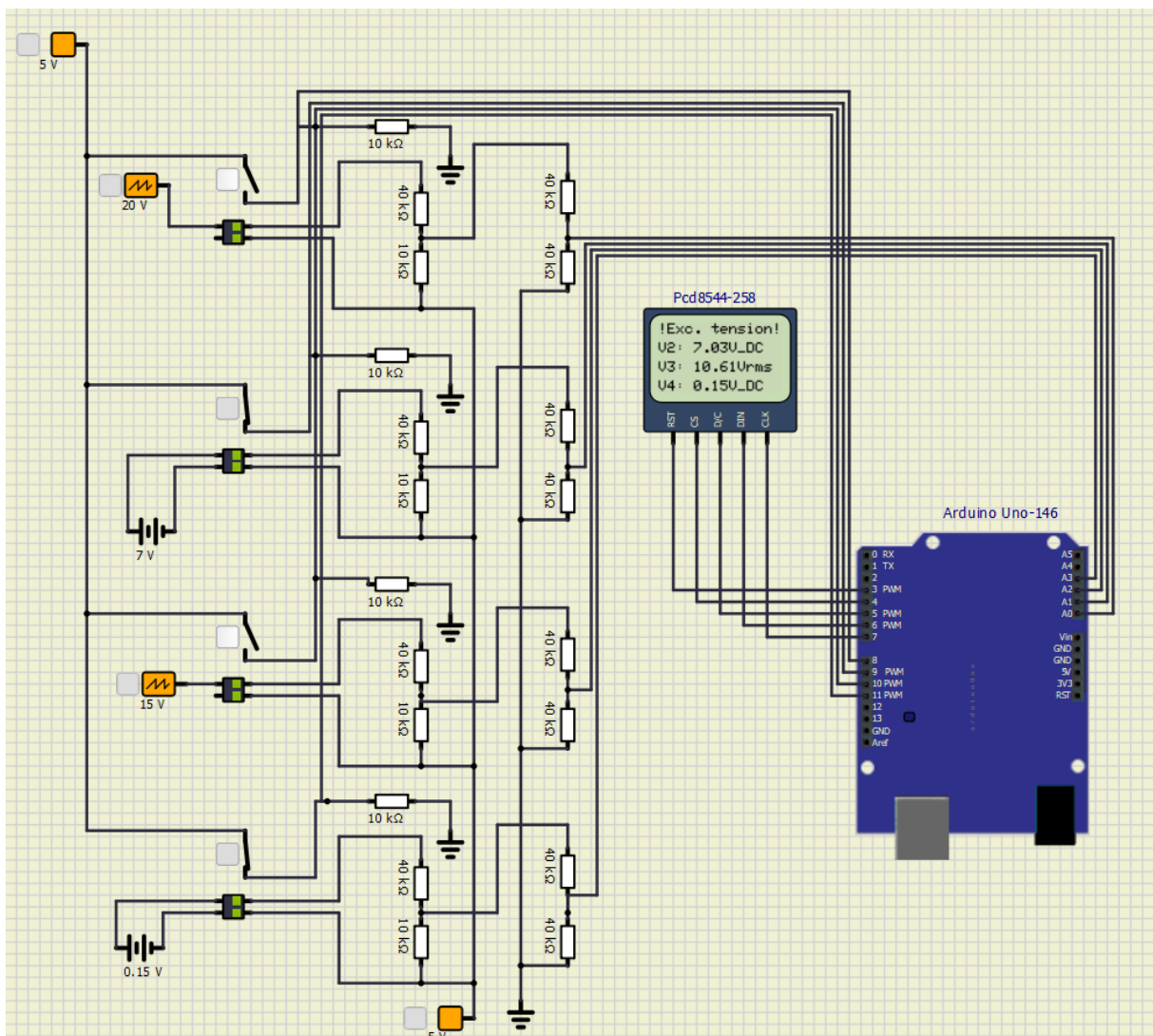


Figura 6: Muestra de que tensiones pequeñas como 0.15V son medidas de forma correcta por el Arduino.

Igualmente, el comportamiento anterior de medición se puede dar para tensiones negativas, en cuyo caso el Arduino muestra en pantalla el valor de la tensión, pero con el signo negativo. También se puede notar, que, debido a los altos valores de las resistencias, el consumo de corriente por parte del sistema del voltímetro es muy bajo, del orden de  $\mu A$ , lo cual es un requisito de un buen voltímetro.

## Envío de datos a archivo CSV

Luego de tener todo el circuito del voltímetro completado, se procede a realizar la etapa de envío de datos desde el simulador hasta un archivo CSV, a continuación, se detalla cada parte del proceso.

Lo primero que se realiza es agregar un interruptor que permite habilitar o deshabilitar el envío de datos por el puerto serial del Arduino. En la siguiente figura se muestra el switch implementado.

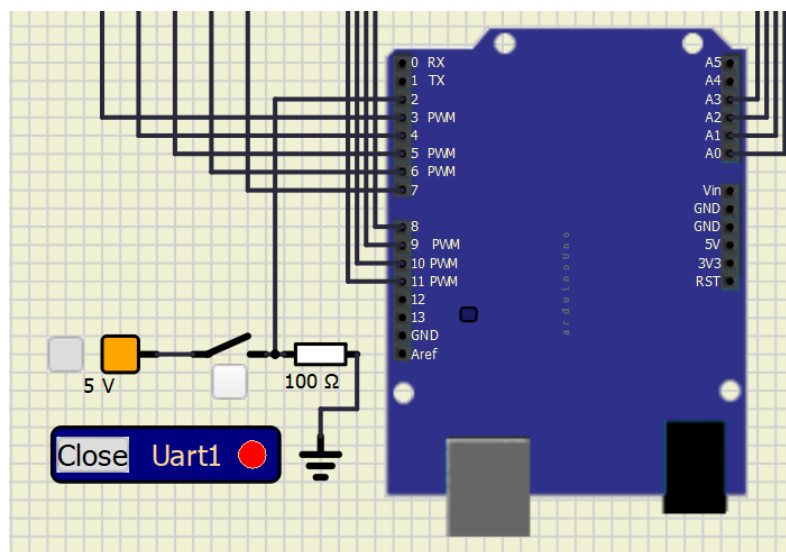


Figura 7: Switch agregado para controlar comunicaciones por UART.

Con este switch, se agrega un pequeño código en el programa del Arduino, que permite habilitar la comunicación con base en el estado del switch.

```
295 uart_en=digitalRead(2);
296 if (uart_en==1){
297     //Enviar info a pc
298     texto_uart=String(valorMultimetro1)+" "+String(valorMultimetro2)+" "+String(valorMultimetro3)+" "+String(valorMultimetro4);
299     Serial.println(texto_uart);
300 }
```

Figura 8: Código del Arduino para controlar envío de datos.

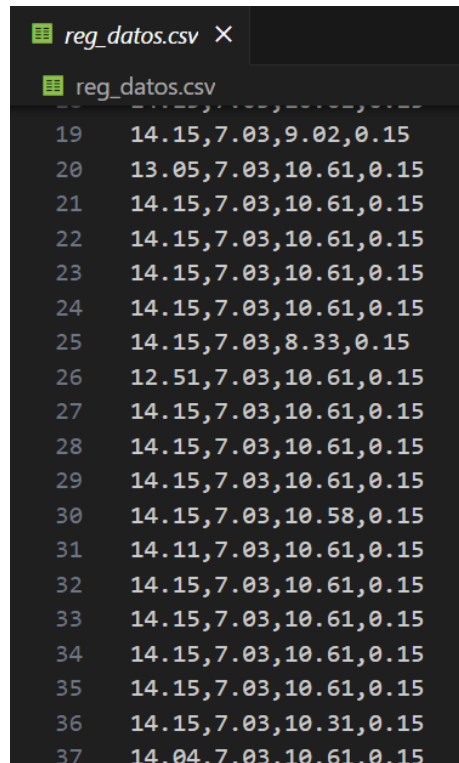
En la figura anterior se muestra el código utilizado para el envío de los datos. Básicamente lo que se realiza es crear una cadena de texto que contiene el valor de los 4 voltímetros, y este texto se envía luego por el puerto serial. Todo esto se realiza únicamente si el switch está cerrado.

Luego de esto, se crea el puerto serial virtual, tal como lo indica el profesor. Y del otro lado del puerto se crea un pequeño script en Python que se encarga de leerlo y cargar los datos a un documento CSV, tal como se muestra a continuación.

```
1  import serial
2  import csv
3
4  puerto_serial = serial.Serial('COM3', 9600, timeout=1)
5
6  while True:
7      """ if puerto_serial.in_waiting > 0: """
8      voltajes = puerto_serial.readline().decode('utf-8').rstrip()
9      print(voltajes + "\n")
10     voltajes = voltajes.split()
11
12     with open('reg_datos.csv', mode='a', newline='') as reg_datos:
13
14         writer = csv.writer(reg_datos)
15         writer.writerow(voltajes)
```

Figura 9: Script de Python para el procesamiento de los datos.

En la figura anterior, se puede observar que el script es bastante sencillo. Su función es leer el puerto serial y decodificar los datos como una cadena de texto, luego esta línea de texto se separa por el símbolo espacio, para genera una lista que contiene en sus cuatro posiciones los valores de las tensiones medidas. Finalmente, el script abre el archivo “reg\_datos.csv”, en donde agrega la lista que se acaba de crear. Con esto es posible guardar constante los valores en el documento, en donde cada columna corresponde al valor de la tensión medida.



```
reg_datos.csv X
reg_datos.csv
19 14.15,7.03,9.02,0.15
20 13.05,7.03,10.61,0.15
21 14.15,7.03,10.61,0.15
22 14.15,7.03,10.61,0.15
23 14.15,7.03,10.61,0.15
24 14.15,7.03,10.61,0.15
25 14.15,7.03,8.33,0.15
26 12.51,7.03,10.61,0.15
27 14.15,7.03,10.61,0.15
28 14.15,7.03,10.61,0.15
29 14.15,7.03,10.61,0.15
30 14.15,7.03,10.58,0.15
31 14.11,7.03,10.61,0.15
32 14.15,7.03,10.61,0.15
33 14.15,7.03,10.61,0.15
34 14.15,7.03,10.61,0.15
35 14.15,7.03,10.61,0.15
36 14.15,7.03,10.31,0.15
37 14.04,7.03,10.61,0.15
```

Figura 10: Archivo CSV con los valores de la tensión medida.

Finalmente, en la figura anterior se muestra el resultado obtenido. Se puede observar que el script guarda correctamente los valores medidos, lo que permite generar un registro de los valores medidos, para luego poder procesarlos como se desee desde el archivo creado.

## Conclusiones y recomendaciones

- Se logró crear el voltímetro de forma exitosa, solucionando problemas como los de las tierras compartidas.
- El uso de un script de Python permite llevar un registro de los datos registrados en el Arduino. Este mecanismo se pudo implementar de forma exitosa, y demostrar de forma sencilla el potencial de esta herramienta.
- Se pudo controlar de forma correcta la pantalla utilizada, esto mediante el uso de una librería que permitió un gran nivel de abstracción, permitiendo una mejor manipulación de los datos mostrados.