Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0624 – Laboratorio de Microcontroladores
II ciclo 2022

Laboratorio 3

GPIO, ADC y comunicaciones

Kenny Wu Wen C08592
Oscar Fallas B92861
Grupo 01

Profesor: MSc. Marco Villalta Fallas

6 de octubre de 2023

Índice

Índice de figuras

1. Resumen

En este laboratorio se van a utilizar el microcontrolador Arduino Uno para diseñar un voltímetro que lea tensiones desde -24 a 24. Se va a trabajar con los GPIOs, ADC y protocolos de comunicación USART y SPI para guardar la lectura de tensiones en una archivo csv.

El repositorio para el laboratorio es el siguiente: GIT

2. Nota teórica

2.1. Arduino UNO

En el campo de los microcontroladores el Arduino UNO es uno de los más utilizados, especialmente en los ámbitos educativos y de robótica principiante, debido a su sintaxis bastante intuitiva y simple de aprender. Las placas de Arduino vienen equipadas con un procesador ATmega328P, 14 pines de entrada digitales, 6 pines de lectura analógica, puertos USB para transmisión de datos, ICSP para la carga del firmware y un botón de reset. Entre las características más importantes a tener en cuenta antes de manipular un microcontrolador son los rangos de uso mínimos y máximos y el consumo de potencia, lo cuál se puede ver a continuación:

Symbol	Description	Min	Typ	Max	Unit
VINMax	Maximum input voltage from VIN pad	6	-	20	V
VUSBMax	Maximum input voltage from USB connector		-	5.5	V
PMax	Maximum Power Consumption	-	-	xx	mA

Figura 1: Uso de potencia[1]

Se puede notar que el arduino es posible de ser alimentado mediante el puerto o la entrada VIN. La manera en como se distruye la corriente entre toda la placa se nota en la siguiente figura:

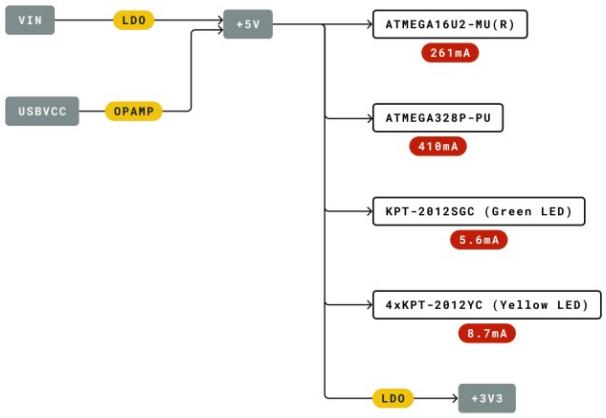

Figura 2: Distribución de corriente entre la placa

2.1.1. GPIOs

Como se mencionó anteriormente la placa de Arduino cuenta con una serie de puertos de entrada/salida de proposito general(GPIO), a continuación se presenta un diagrama con la distribución de pines a través de la placa:

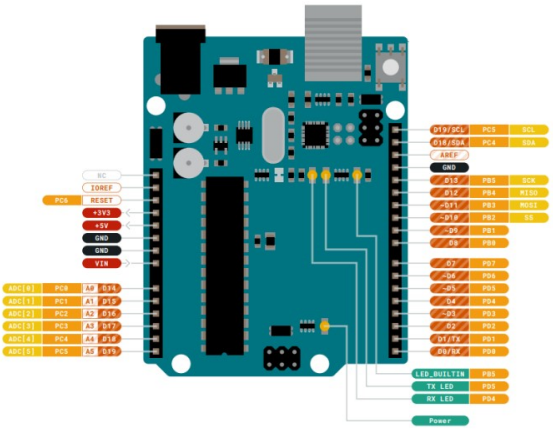

Figura 3: Diagrama Arduino UNO[1]

Ahora bien, estos pines se distribuyen por características:

- **Pines de energía**: Estos son los que proveen o generan tensión a la placa, entre ellos tenemos la entrada de tensión Vin, el generador de 5V, el generador de 3.3V, el pin de tierra GND.
- **Pines de lectura analógica**: Estos pines se distribuyen desde el A0 hasta el A3 los que funcionan como entrada o salida, además de A4 y A5 que funcionan únicamente como entrada.
- **Pines Digitales**: Estos pines son capaces de leer o generar señales digitales y van distribuidos desde D0 hasta el D0 hasta el D9, los pines 11, 12, 13 y 14 son para protocolo SPI.

2.1.2. Convertidor analógico a digital

Las señales eléctricas son continuas y variables en el tiempo, por lo que se necesita una forma de representarlas de manera que sean comprensible su lectura por un procesador. Un convertidor analógico digital(ADC) tiene la función de transformar una lectura de tensión en bits, con el fin que el Arduino utilizar dichas variables a conveniencia. El principio es simple, los puerto de lectura analógica del Arduino van desde los 0V hasta los 5V, estos puertos tienen una resolución de 10 bit, por lo tanto pueden representar valores hasta $2^{N-1} = 1024$. Por lo tanto, es simple realizar una proporción directa entre 0V a un dato analógico de 0, y 5V a una lectura analógica de 5V [2].

2.1.3. Interrupciones

El Arduino UNO tiene dos pines para interrupciones los cuales son el **3** y **2** y para instanciarlo se utiliza el metodo **attachInterrupt(interrupt, ISR, mode)**, con interrupt siendo 0/1, ISR la funcion a llamar cuando ocurre la interrupcion y mode puede ser:

- LOW, La interrupción se dispara cuando el pin es LOW.
- CHANGE, Se dispara cuando pase de HIGH a LOW o viceversa.
- RISING, Dispara en el flanco de subida (Cuando pasa de LOW a HIGH).
- FALLING, Dispara en el flanco de bajada (Cuando pasa de HIGH a LOW).

2.2. Protocolos de comunicaciones

2.2.1. UART

Las siglas UART significan Receptor-Transmisor Asincronico Universal o en ingles *Universal Asyncronic Receptor-Transmitter*. Es un hardware que se encarga para la comunicación serial entre dos dispositivos UART significa Receptor/Transmisor Asíncrono Universal. Es un dispositivo de hardware (o circuito) utilizado para la comunicación en serie entre dos dispositivos. En el Arduino tenemos un pin Tx y otro Rx que son para la comunicación UART [3].

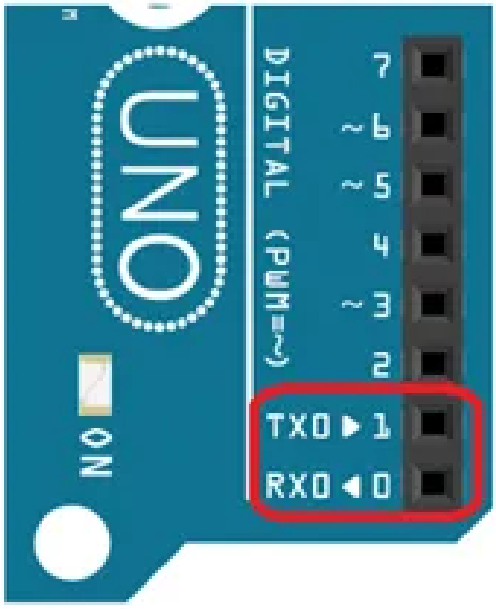

Figura 4: Comunicación UART

El como funciona el protocolo de comunicación UART, es simple, se envían datos en serie y luego se reconstruyen. Estos datos en realidad son paquetes se dividen en bytes y luego a bits y los bytes son etiquetas que significan cuando empiezan, en donde se ubican los datos y donde termina el paquete de enviarse.

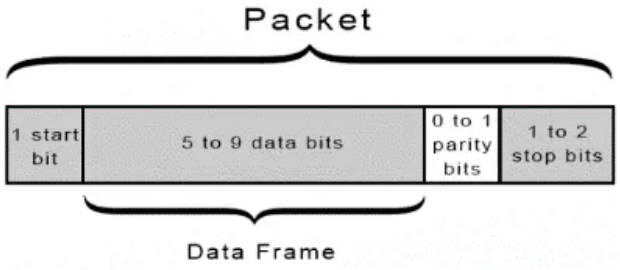

Figura 5: Paquete UART

El UART se requiere que los dos dispisitivos a comunicarse tengan los mismos parametros de comunicación UART, como la tasa de baudios, la longitud de datos, el bit de paridad, el numero de bits de parada y el control de flujo [3].

- Tasa de baudios: numero de bits por segundo que un dispositivos logra transmitir/recibir, generalmente los valores utilizados son 1200, 2400, 4800, 9600, 19200, 38400, 57600 y 115200 bps.

- Longitud de datos: cantidad de bits por bytes de datos.
- Bit de paridad: bit agregado a los datos transmitidos, que permiten al receptor identificar la cantidad de 1's en los datos transmitidos es par o impar.
- Control de flujo: se utiliza para controlar la cantidad de datos enviados y recibidos con el fin evitar el riesgo a perder datos.

2.3. PCD8544-1

El PCD8544-1 es una pantalla LCD conocido por utilizarse en el Nokia3310. Este es una pantalla gráfica monocromática de 84x48 pixeles con retroiluminación LED a baja luz [4].

Algunas de sus características según la hoja de fabricante [5]:

- Pantalla de 48x84 bits
- Un unico chip microcontrolador/driver LCD
- Interfaz serial de hasta 4.0Mbits/s máximo
- Tensión de operación desde los 2.7 a los 3.3 V
- Consumo bajo de potencia

En la figura 6, se observa el diagrama de bloques del dispositivo. El cual se puede ver en el I/O buffer las entradas para la comunicación serial y en la figura 7, las funciones de cada pin.

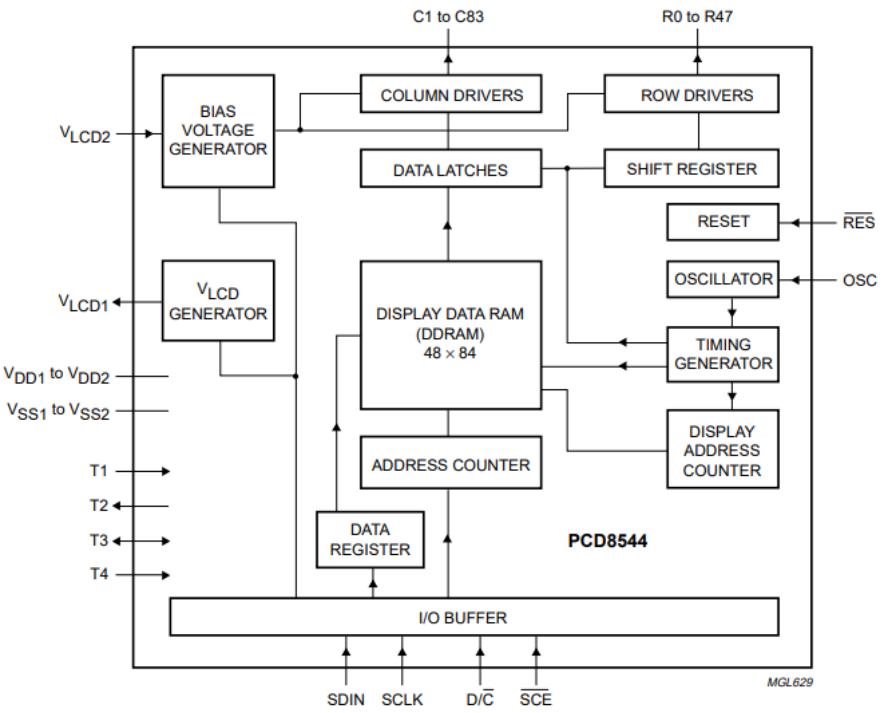

Figura 6: Diagrama de bloques del PCD8544-1

SYMBOL	DESCRIPTION
R0 to R47	LCD row driver outputs
C0 to C83	LCD column driver outputs
V_{SS1}, V_{SS2}	ground
V_{DD1}, V_{DD2}	supply voltage
V_{LCD1}, V_{LCD2}	LCD supply voltage
T1	test 1 input
T2	test 2 output
T3	test 3 input/output
T4	test 4 input
SDIN	serial data input
SCLK	serial clock input
$D/\overline{C}$	data/$\overline{\text{command}}$
$\overline{SCE}$	chip enable
OSC	oscillator
$\overline{RES}$	external reset input
dummy1, 2, 3, 4	not connected

Figura 7: Funciones de los pines del PCD8544-1

2.3.1. Sistema de coordenadas

La pantalla tiene un sistema de coordenadas que están descritos por coordenadas horizontales (H) y coordenadas verticales (Y). Cada coordenada es asignado a una posición de píxel de la pantalla [4].

Para simplificar el como escribir en la pantalla se puede utilizar librerías en especifico PCD8544.h o Adafruit_GFX.h con Adafruit_PCD8544.h [6]. Esto nos simplifica a la hora de escribir en la pantalla y no tener que modificar píxel por píxel.

2.4. Voltímetro

Se va a crear un voltímetro con cuatro canales, que pueden recibir un rango de [-24,24] V DC/AC y se deben de visualizar estos valores en una pantalla LCD. Además, los datos leídos serán almacenados en un archivo csv.

2.4.1. Switches

Vamos a tener dos switches, uno para seleccionar modo AC/DC y otro para activar/desactivar la transmisión de datos serial. Ambos switches tendrán el diseño de la figura 8.

Se agrega el capacitor para el riesgo del efecto de rebote, el cual el tiempo de asentamiento va ser de $5T = 500\mu$s .

2.4.2. Conversión [-24,24]V a [0,5]V

Para pasar la tensión de entrada de una escala mayor a una menor, ya que el arduino lee de 0 a 5V, vamos a ocupar un circuito con división de tensión con dos fuentes como se muestra en la figura 9

Obtenemos la expresión para la salida V_{out}, utilizando el teorema de superposición, el cuál explica que cada fuente es independiente y la suma de lo que aporta cada una es el valor total de V_{out}.

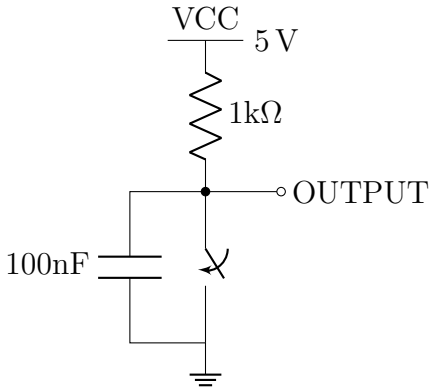

Figura 8: Configuración pull-up de un switch

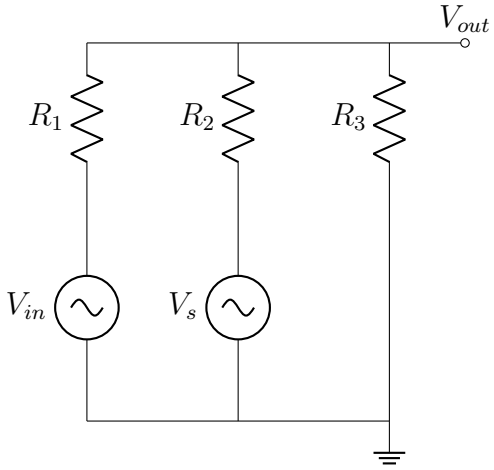

Figura 9: Divisor de tensión con dos fuentes

Primero analizamos una fuente apagada y obtenemos el circuito de la figura 10. El fácilmente mediante LCK, ley de ohm y división de tensión obtenemos que:

$$V_{out1} = \frac{R_1||R_3}{(R_1||R_3) + R_2} \cdot V_s \tag{1}$$

Lo mismo aplicamos con la otra fuente y el resultado es similar:

$$V_{out2} = \frac{R_2||R_3}{(R_2||R_3) + R_1} \cdot V_{in} \tag{2}$$

Por lo tanto, sumando (1) y (2):

$$V_{out} = V_{out1} + V_{out2}$$

$$\boxed{\frac{R_1||R_3}{(R_1||R_3) + R_2} \cdot V_s + \frac{R_2||R_3}{(R_2||R_3) + R_1} \cdot V_{in}} \tag{3}$$

Por lo que tenemos cuatro grados de libertad y sabemos que cuando la $V_{in} = 24 \implies V_{out} = 5$ y $V_{in} = -24 \implies V_{out} = 0$, por lo que tenemos dos ecuaciones y cuatro grados de libertad, entonces podemos escoger dos variables y las otras dos resolviendo las condiciones. En nuestro caso, vamos a utilizar $R_3 = 1000\Omega$ y $V_s = 5\text{V}$, con eso las ecuaciones a resolver son:

$$5 = \frac{R_1||1000}{(R_1||1000) + R_2} \cdot 5 + \frac{R_2||1000}{(R_2||1000) + R_1} \cdot 24 \tag{4}$$

$$0 = \frac{R_1||1000}{(R_1||1000) + R_2} \cdot 5 + \frac{R_2||1000}{(R_2||1000) + R_1} \cdot -24 \tag{5}$$

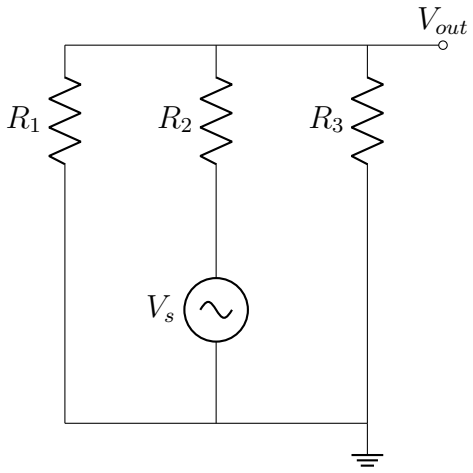

Figura 10: Fuente V_{in} apagada

Resolviendo para R_2 y $R1$ tenemos:

$$R_1 = 7600\Omega \tag{6}$$
$$R_2 = 1425\Omega \tag{7}$$

2.5. Firmware

Para la construcción del Firmware del programa se utilizó el ArduinoIDE el cuál está basado en lenguaje C. La lógica de flujo del código se ve representada en el siguiente diagrama:

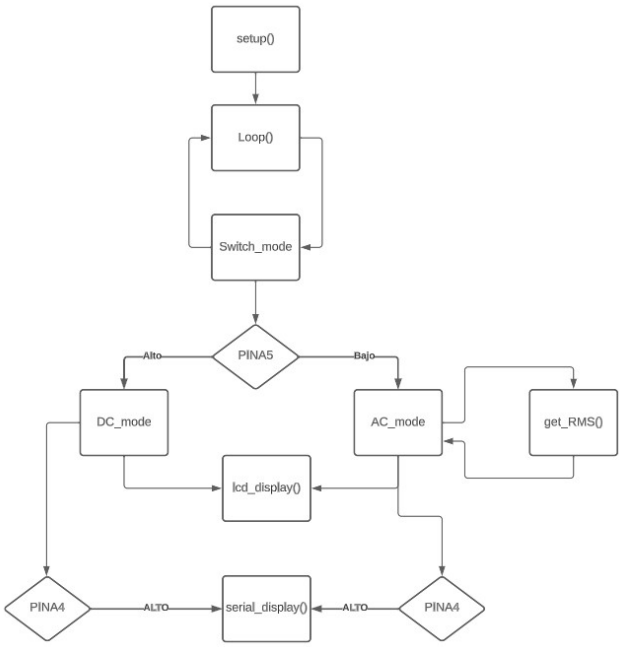

Figura 11: Diagrama de flujo para voltimeter.ino

2.5.1. Setup

Está función se encargara de inicializar los GPIOs, interrupciones, configuración del baudrate de la transimisión serial y el despliegue de un mensaje de bienvenida.

- Establece el baudrate en un valor de transmisión de 9600.
- Asigna los pines 8, 9, 10, 11 como salidas digitales para los LED de advertencia de alta tensión.
- Establece la resolución de la pantalla PCD8544
- Envía un mensaje de inicio por medio de la pantalla.
- Configura una interrupción que limpia la pantalla.

```
void setup() {
    Serial.begin(9600);
    pinMode(8, OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(10, OUTPUT);
    pinMode(11, OUTPUT);
    lcd.begin(84, 48);

    lcd.clear();
    lcd.setCursor(1,0);
    lcd.print("Welcome to");
    lcd.setCursor(1,1);
```

```
    lcd.print("4-Channel");
    lcd.setCursor(1,2);
    lcd.print("Voltimeter!");

    attachInterrupt(digitalPinToInterrupt(2), clear_display, CHANGE);

    delay(2000);

}
```

2.5.2. Lazo Principal

El lazo principal es un bloque que únicamente hace llamada de manera infinita a la función Switch_mode.

```
void loop() {
    Switch_mode();
}
```

La función Switch_mode() lee de manera analógica el pin A5, el cuál sirve de interruptor para determinar el modo de operacion; AC_mode() cuando la lectura es menor a la 2.5V o a 511.5, y DC_mode() cuando la lectura es mayor a dichos parámetros.

```
void Switch_mode(){
  float button = analogRead(A5);

  if(button < converter){
    AC_mode();
  } else DC_mode();
}
```

2.5.3. Modos de Operación

Estas funciones son la esencia principal del voltímetro. Primero tenemos el modo de lectura en corriente directa, DC_mode(); dicho modo le de manera analógica los pines A0, A1, A2 y A3; los cuáles corresponde a los cuatro canales de lectura del voltímetro. Por consiguiente, envían dichas mediciones a la función de advertencia de alto voltaje que se explicará más adelante, y realiza la conversión por proporción directa de analógico a digital. La forma de realizar dicha conversión esta dada por la siguiente fórmula:

$$V = (valor_analogico - 511.5) * (24/511.5)$$

Dicha fórmula es una variación de la regla de 3 o proporción directa adaptada una transformación de escala de [0,5] a [-24,24] V. Una vez calculado envía los datos a las funciones encargadas de su despliegue.

```
void DC_mode() {
    // Read the analog values from pins A0 to A3
    vCH1 = (analogRead(A0) - converter) * (referenceVoltage / converter);
    vCH2 = (analogRead(A1) - converter) * (referenceVoltage / converter);
    vCH3 = (analogRead(A2) - converter) * (referenceVoltage / converter);
    vCH4 = (analogRead(A3) - converter) * (referenceVoltage / converter);

    high_v_warming(
```

```
        vCH1,
        vCH2,
        vCH3,
        vCH4);

    lcd_display("MODO-DC", vCH1, vCH2, vCH3, vCH4);

    serial_display("MODO-DC",vCH1,vCH2,vCH3,vCH4);
}
```

Ahora, el AC_mode() es muy similar al modo anterior con la diferencia que utiliza una función extra llamada get_RMS(), la cuál realiza un muestreo simple de la señal de tensión en corriente continua, encontrando su valor máximo para finalmente obtener su valor RMS mediante la siguiente fórmula:

$$V_{RMS} = V_{max}\frac{1}{\sqrt{2}}$$

```
float get_RMS(int PIN){
  const int samples = 1000;
  float tmax = 0.00;
  float one_sqrt = 0.7071;

  for(int i = 0; i < samples; i++){
    float t = analogRead(PIN);
    if(t > tmax) tmax = t;
    }
    float t_rms = ((tmax - converter)*(referenceVoltage / converter));

    return t_rms * one_sqrt;
  }
  void AC_mode(){
    // Calcular la tensi n RMS para cada canal
    vCH1 = get_RMS(A0);
    vCH2 = get_RMS(A1);
    vCH3 = get_RMS(A2);
    vCH4 = get_RMS(A3);

    high_v_warming(
      vCH1 * 1.41,
      vCH2 * 1.41,
      vCH3 * 1.41,
      vCH4 * 1.41);

    // Mostrar los valores en la pantalla
    lcd_display("MODO-AC", vCH1, vCH2, vCH3, vCH4);

    serial_display("MODO-AC",vCH1,vCH2,vCH3,vCH4);
  }
```

2.5.4. Funciones de despliegue

Estas funciones son relativamente simples, primero lcd_display() es una función que mediante la librería PCD8544.h permite la impresión de datos de manera sencilla, como se explicó en la

nota teórica de la funcionalidad de la pantalla PCD8544. Se debe notar que para cada impresión primero se debe ajustar el cuadrante y luego enviar el comando de print().

```
void lcd_display(const char *MODO,
                 float vCH1,
                 float vCH2,
                 float vCH3,
                 float vCH4) {
    lcd.setCursor(0, 0);
    lcd.print(MODO);

    lcd.setCursor(0, 1);
    lcd.print("CH1:");
    lcd.print(vCH1);
    lcd.print("V");

    lcd.setCursor(0, 2);
    lcd.print("CH2:");
    lcd.print(vCH2);
    lcd.print("V");

    lcd.setCursor(0, 3);
    lcd.print("CH3:");
    lcd.print(vCH3);
    lcd.print("V");

    lcd.setCursor(0, 4);
    lcd.print("CH4:");
    lcd.print(vCH4);
    lcd.print("V");

}
```

La función serial_display() es casi indentica a la anterior con la diferencia de referenciar a la clase lcd lo hace a la clase Serial que permite el despliegue por monitor serial, que posteriormente será capturada dicha salida mediante un script de python adjuntado en 5.

```
void serial_display(const char *MODO,
                    float vCH1,
                    float vCH2,
                    float vCH3,
                    float vCH4){
   float serial_en = analogRead(A4);

   if(serial_en > converter){
    Serial.println(MODO);
    Serial.println(vCH1);
    Serial.println(vCH2);
    Serial.println(vCH3);
    Serial.println(vCH4);}
  }
```

Finalmente, una función simple que se activa con la interrupción configurada en el setup, con el fin de limpiar la pantalla lcd cada que haya un cambio de modo de operación.

```
void clear_display() {
  lcd.clear();
}
```

2.5.5. Función de advertencia

Esta función utiliza como salida los pines configurados al inicio, para advertir si una alta tensión esta circulando. Lo anterior lo hace simplemente determinando si el dato ya convertido a un rango de [-24,24] se encuentra menor a -20V o mayor a 20V. En caso de esto pone el alto las salidas de pin.

```
void high_v_warming(float vCH1,
                    float vCH2,
                    float vCH3,
                    float vCH4){
  if(vCH1 > 20.00 || vCH1 < -20.00) digitalWrite(8, HIGH);
  else digitalWrite(8, LOW);

  if(vCH2 > 20.00 || vCH2 < -20.00) digitalWrite(9, HIGH);
  else digitalWrite(9, LOW);

  if(vCH3 > 20.00 || vCH3 < -20.00) digitalWrite(10, HIGH);
  else digitalWrite(10, LOW);

  if(vCH4 > 20.00 || vCH4 < -20.00) digitalWrite(11, HIGH);
  else digitalWrite(11, LOW);
  }
```

2.6. LEDs

Los LEDs (*light-emitting diode*) son diodos emisores de luz. Contienen dentro un semiconductor el cual al recibir una diferencia de potencial continua, produce luz, fenómeno conocido como electro luminiscencia [7].

Cuando se conecta un diodo LED, la tensión que aguanta es normalmente de 2 V, en caso de ser mayor se debe colocar una resistencia en serie para no quemar el LED.

En este laboratorio se van a usar LEDs amarillos de 3mm con un costo de 100 colones por unidad. Funcionan a 2V y 20mA [8].

Como los pines del PIC entregan una tensión de 5 V aproximadamente, por lo tanto la resistencia de protección va ser:

$$R_{led} = \frac{5-2}{20 \times 10^{-3}}$$
$$\boxed{R_{led} = 150\Omega} \tag{8}$$

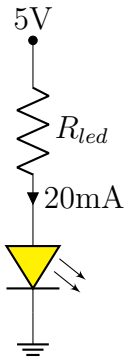

Figura 12: LED conectado a un pin

Para dos LEDs en serie con los mismas parámetros:

$$R_{led} = \frac{5 - 2 \cdot 2}{20 \times 10^{-3}}$$
$$\boxed{R_{led} = 50\Omega} \tag{9}$$

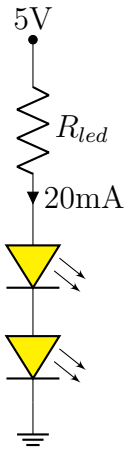

Figura 13: LED conectado a un pin

3. Desarrollo/Análisis de resultados

3.1. Análisis de funcionalidad electrónica

En las figuras 14 y 15, se puede apreciar que la formula (3) concuerda con lo práctico y se tiene cuando la entrada es 24V la salida es 5V, cuando la entrada es -24V la salida es 0V y cuando la entrada es 0V la salida es 2.5V.

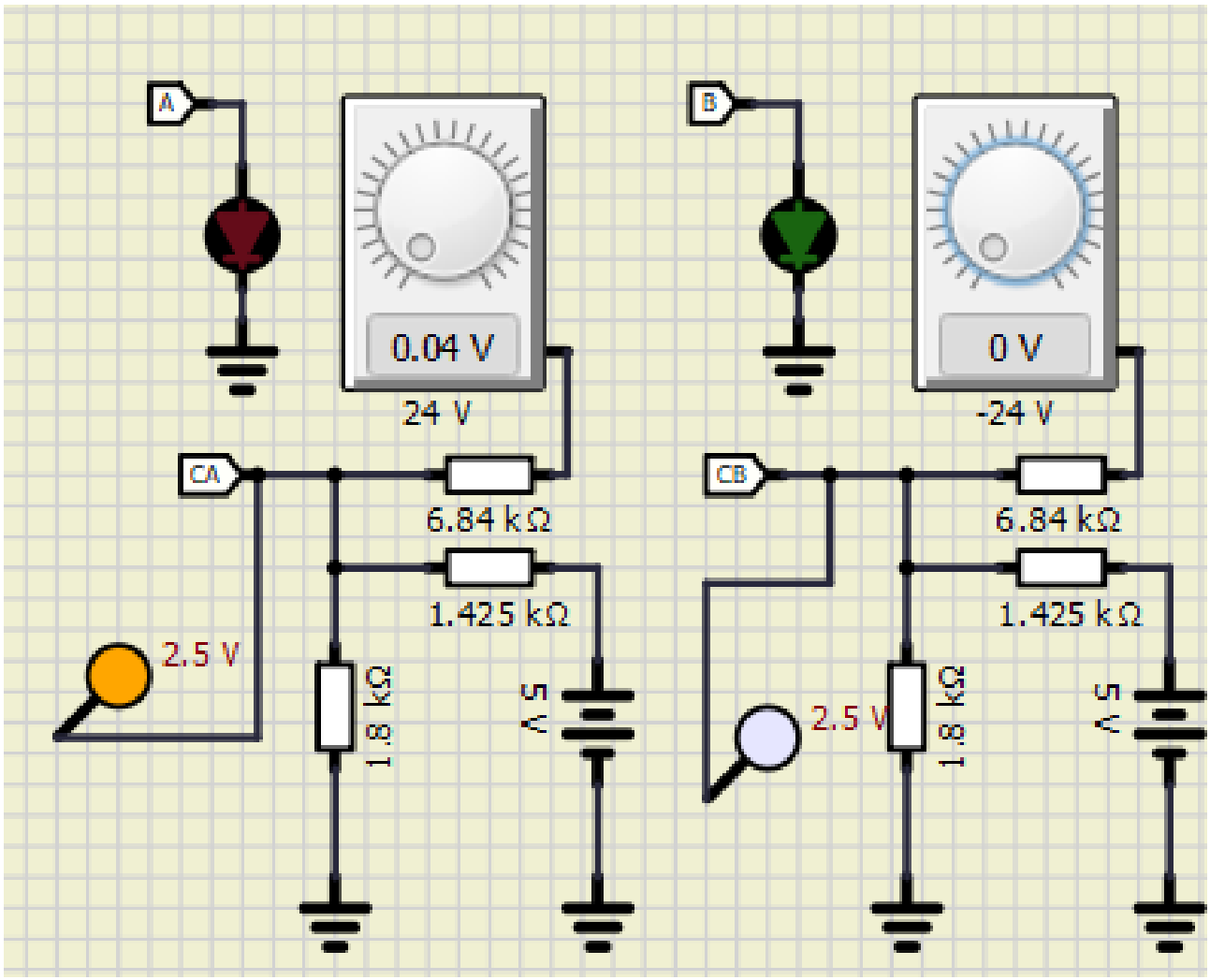

Figura 14: Rango cuando entrada es 0V

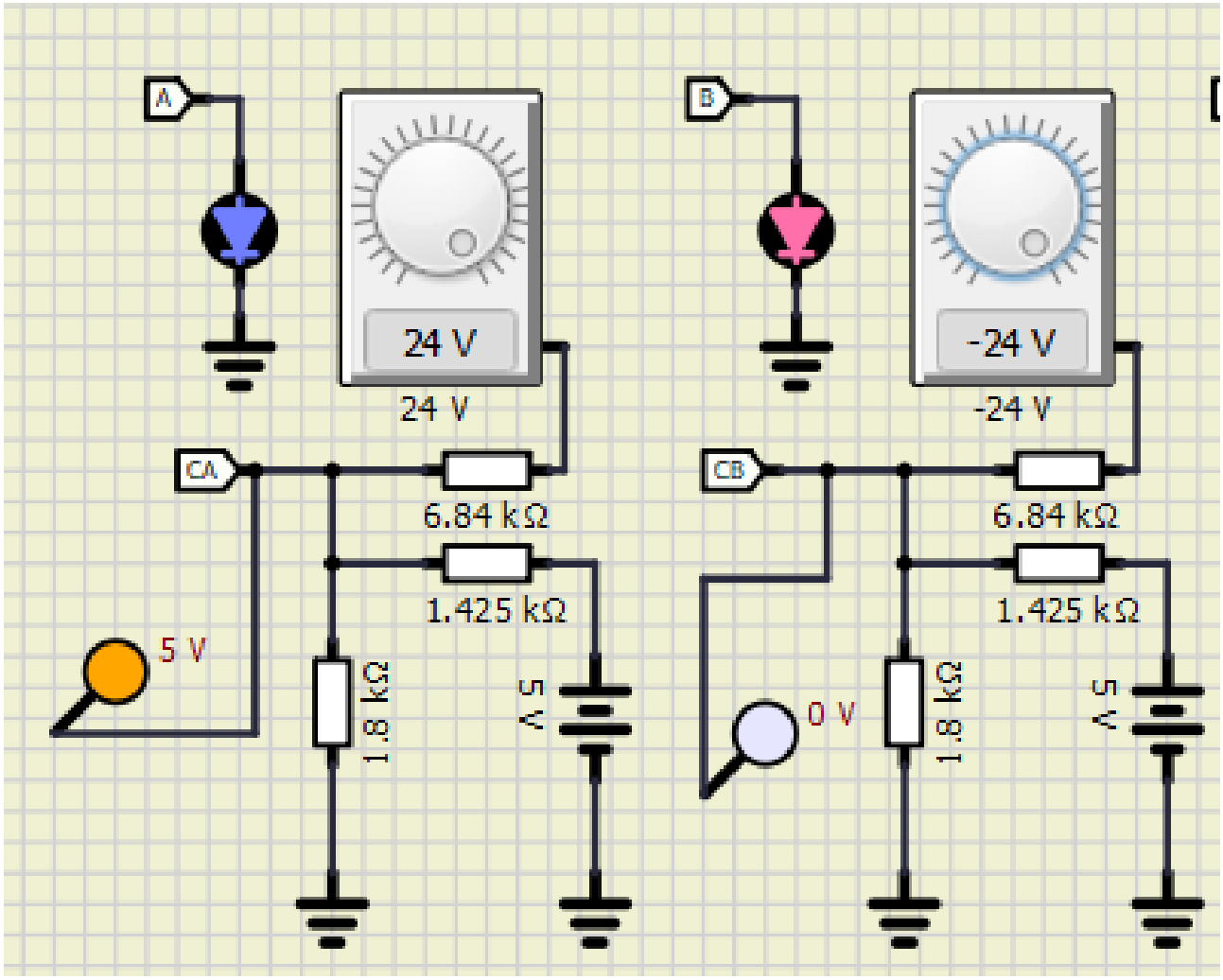

Figura 15: Rango cuando entrada es 24V y -24

3.2. Análisis de la funcionalidad del programa

Se puede observar en la figura 16, donde los cuatro canales funcionan de manera correcta en el modo DC, tomar en cuenta que la tensión del canal D al ser variable no se toma bien, también, se aprecia que cuando las tensiones son menores a -20 o mayores a 20 el LED correspondiente de cada canal se enciende. Agregar que se puede observar la comunicación UART en la consola cmd que los datos recibidos son los mismos.

Por otro lado, si activamos el modo AC, como se muestra en la figura 17, se aprecia que el valor del canal 4 ya es constante al ser el valor rms de la señal senoidal.

Además, se puede observar la salida en formato csv de los valores de los canales en la figura 18. Suma a esto, se aprecia el cambio de modo DC a AC. Tomar en cuenta que el archivo csv se actualizara solo cuando el programa de python pare de funcionar al este estar escribiendo continuamente.

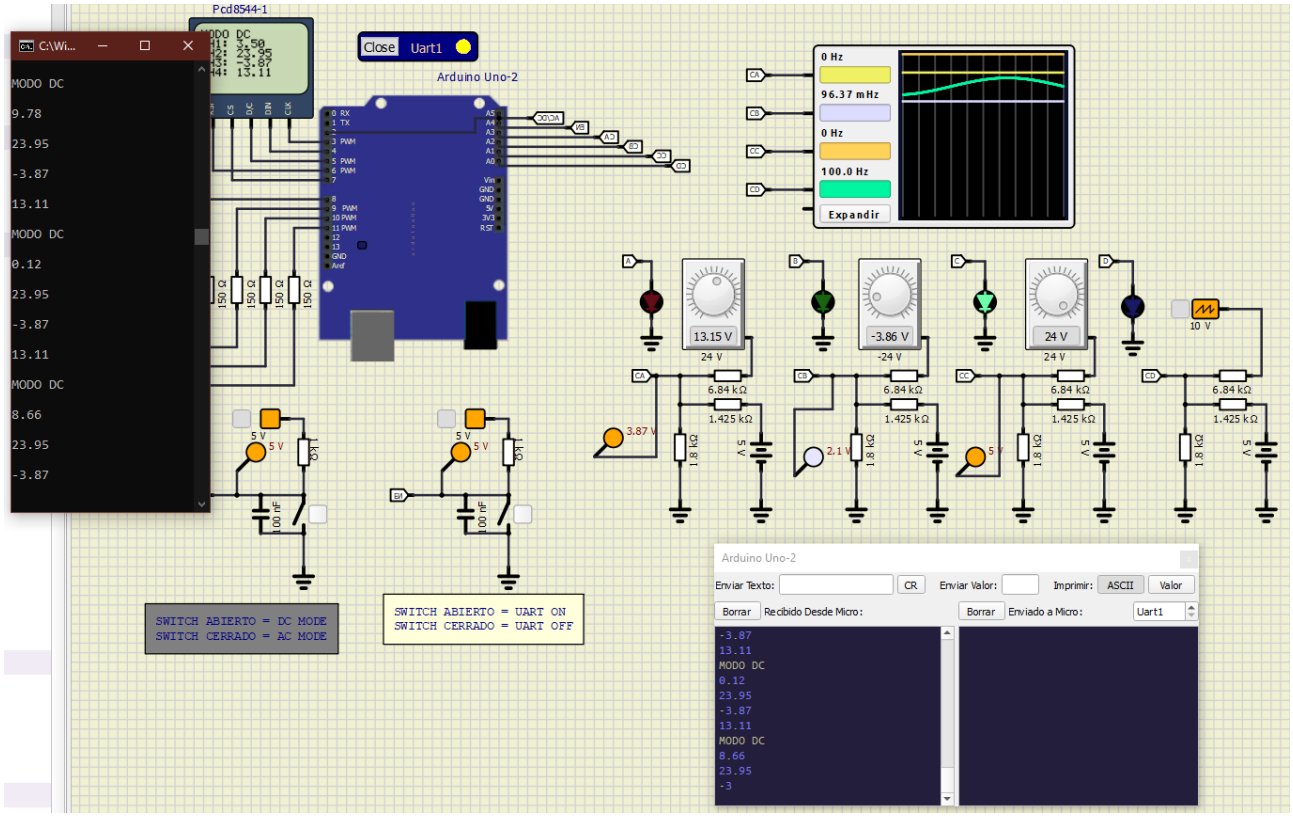

Figura 16: Funciomiento en modo DC y UART activado

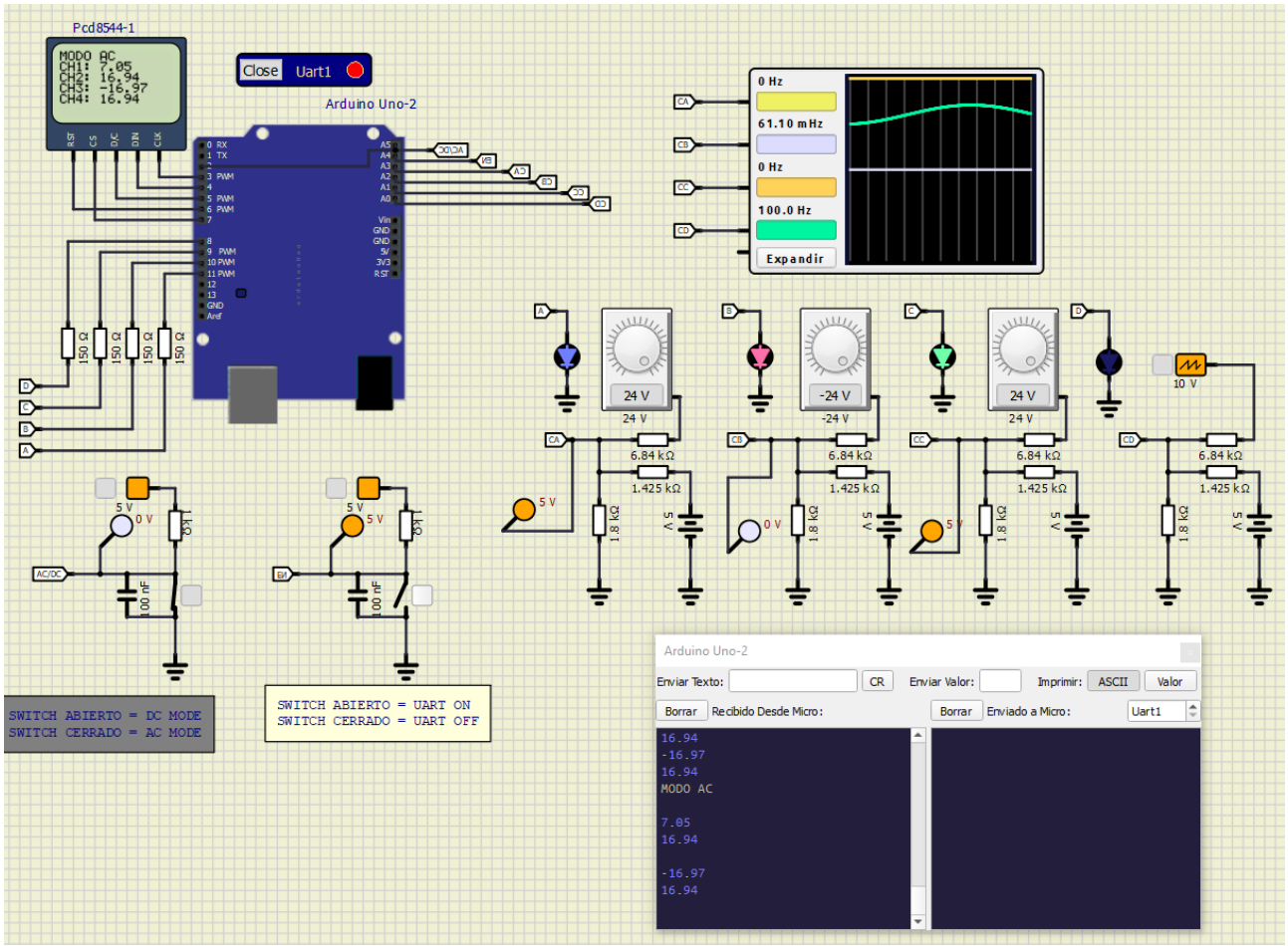

Figura 17: Funciomiento en modo AC y UART activado

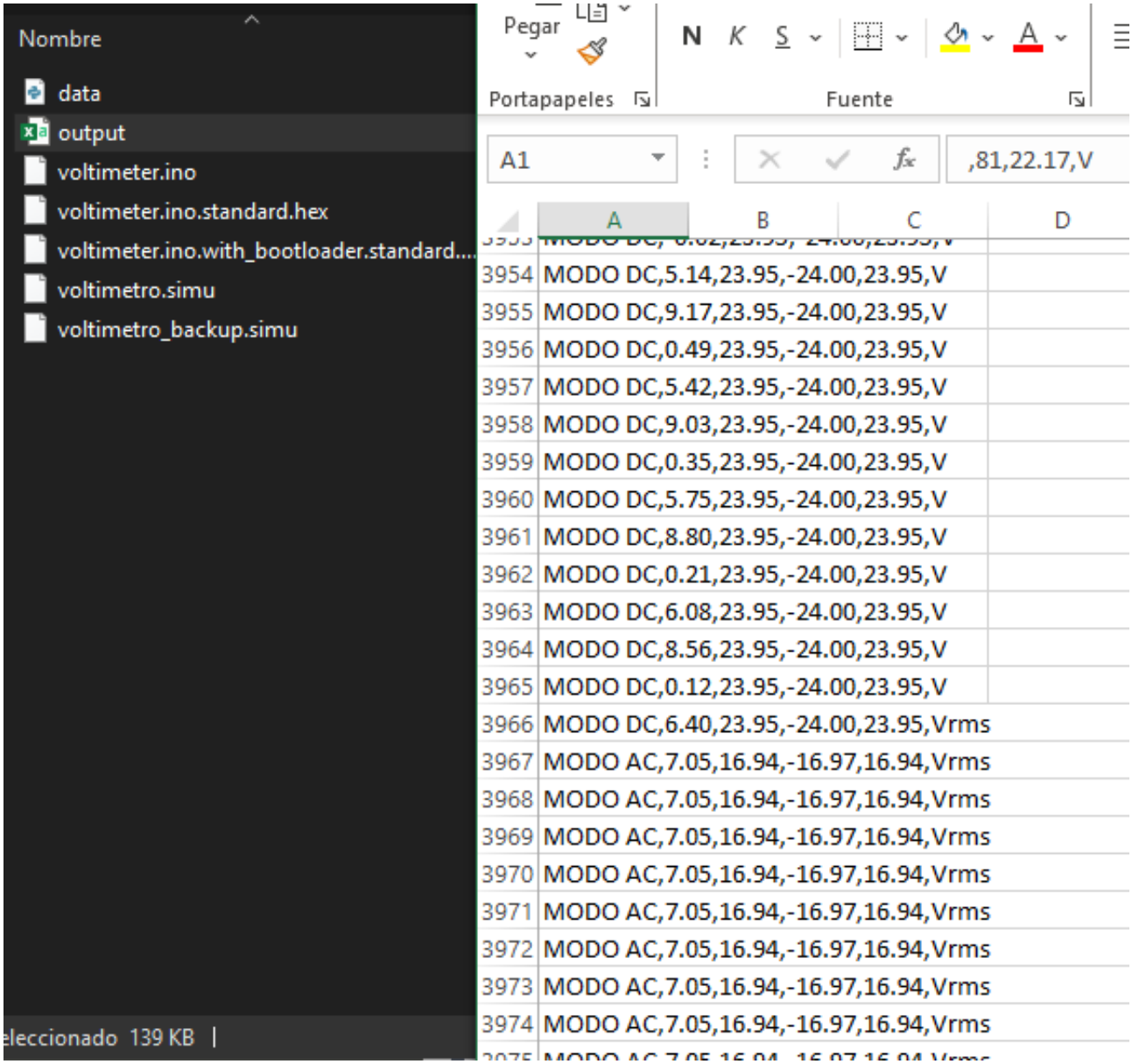

Figura 18: Archivo csv

4. Conclusiones y recomendaciones

- Se logro crear un circuito para convertir las tensiones adecuadas para el arduino
- Se pudo crear un voltímetro de cuatro canales en modo AC o DC
- Se comprendió la comunicación UART y utilizarlo
- Se entendio más sobre el funcionamiento de las pantallas LDC
- Se profundizo el entendimiento sobre el arduino
- Se recomienda siempre tomar en cuenta los detalles como efecto rebote, corriente de entradas a los dispositivos y valores de tensiones que puedan dañar equipo
- A la hora de escrbir firmware, tener orden y se recomienda escribir métodos para la mayoría de acciones, ya que permite entender mejor el código

Referencias

[1] U. Arduino, "Arduino uno datasheet," *Versión A000066*, 2020.

[2] L. Llamas, "Entradas analógicas en arduino," 9 2014. [Online]. Available: https://www.luisllamas.es/entradas-analogicas-en-arduino/

[3] Arduino. (2023) Cómo configurar la comunicación uart en arduino. [Online]. Available: https://arduino.cl/como-configurar-la-comunicacion-uart-en-arduino/

[4] D. E. Hoy, "Lcds gráficos pcd8544," 10 2023. [Online]. Available: https://www.diarioelectronicohoy.com/blog/lcds-graficos-pcd8544

[5] S. Electronics, "48 x 84 pixels matrix lcd controller/driver," 10 2023. [Online]. Available: https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf

[6] Adafruit, "Arduino driver for pc8544, most commonly found in small nokia 5110's," 10 2023. [Online]. Available: https://github.com/adafruit/Adafruit-PCD8544-Nokia-5110-LCD-library

[7] "¿qué es un led? tipos de led, aplicaciones y usos." Visual Led, 2023. [Online]. Available: https://visualled.com/glosario/que-es-un-led/

[8] "Led 3mm amarillo," DBU Electronics, 2023. [Online]. Available: https://www.dbuelectronics.cr/leds/624-led-3mm-amarillo.html

5. Apéndice

5.1. Precios de componentes

Cuadro 1: Precios de cada componente(Tienda)

Componente	Precio (Colones)	Cantidad
ARDUINO UNO	15500	1
PCD8544	4800	1
100 ohm	300	4
300 ohm	300	4
1k ohm	300	2
1.5k ohm	300	4
Botón	200-500	2
Cerámico 0.1uF	80	2
LEDs 3mm	100	4

5.2. Analizador Serial de python

```
import serial
from re import search as re_search

ser = serial.Serial(
    port = 'COM3',\
    baudrate = 9600,\
    parity = serial.PARITY_NONE,\
    stopbits=serial.STOPBITS_ONE,\
    bytesize=serial.EIGHTBITS,\
    timeout=0\
)

f = open('output.csv', 'w+')

print(f'connected to: {ser.portstr}')

line = []

reg = r'MODO\s+(?P<mode>\w+)'

f_line = True

while True:
    for c in ser.read():
        c = chr(c)
        #print(c)
        line.append(c)
        if c=='\n':
            print(''.join(line))
            str = ''.join(line)

            re_result = re_search(reg, str)
            if(re_result is not None):
                if(re_result.group('mode') == 'AC'):
                    if f_line:
                        f.write(str.strip())
                    else: f.write(','+ 'Vrms' + "\n" + str.strip())
                else:
                    if f_line:
                        f.write(str.strip())
                    else: f.write(',' + 'V' + "\n" + str.strip())
            else:
                f.write("," + str.strip())
                f_line = False
            line = []
            break

ser.close()
```