Universidad de Costa Rica

Laboratorio de Microcontroladores

Laboratorio #3:
Arduino: GPIO, ADC y comunicaciones

Prof. MSc. Marco Villalta Fallas

Marco Vásquez Ovares
B17032
Grupo: 01

II - 2024

Índice

1. Introducción **5**

2. Nota teórica **6**
2.1. Arduino UNO . . . 6
2.2. Diagrama de pines . . . 7
2.2.1. Pines de Alimentación: . . . 7
2.2.2. Pines Digitales (D0 - D13): . . . 7
2.2.3. Pines Analógicos (A0 - A5): . . . 8
2.2.4. Pines de Comunicación: . . . 8
2.3. Descripción de pines: . . . 8
2.4. Mapa de memoria de datos . . . 8
2.5. Especificaciones eléctricas . . . 8
2.6. Periféricos . . . 9
2.6.1. Interrupciones . . . 9
2.7. Botón/Switch . . . 10
2.8. Pantalla LCD PCD8544 . . . 10
2.9. Convertidor analógico a digital (CAD) . . . 11
2.10. Protocoloes de comunicación . . . 11
2.10.1. USART (Universal Synchronous/Asynchronous Receiver-Transmitter) . . . 11
2.10.2. SPI (Serial Peripheral Interface) . . . 11
2.11. Diseño del circuito . . . 12

3. Desarrollo **13**
3.1. Análisis de resultados . . . 13
3.1.1. Resistencias de protección para los LED's . . . 13
3.1.2. Efecto rebote . . . 13
3.1.3. Convertir [-24, 24]V a [0, 5]V . . . 13
3.2. Análisis del Firmware . . . 15
3.2.1. Variables de interés . . . 15
3.3. Función get_RMS(int PIN) . . . 15
3.4. Funciones de Modo . . . 16
3.5. Función warning() . . . 16
3.6. Bucle Principal loop() . . . 17
3.7. Análisis electrónico . . . 17
3.7.1. DC Mode . . . 18
3.7.2. AC Mode . . . 19
3.7.3. Transmisión Serial . . . 20
3.7.4. archivo .csv . . . 21

4. Conclusiones y Recomendaciones **22**
4.1. Conclusiones . . . 22
4.2. Recomendaciones . . . 22

Bibliografía **23**

5. Apéndices **23**

Índice de figuras

1. Arduino UNO 6
2. Diagrama de pines, Arduino UNO 7
3. Power Consumption, Arduino UNO 9
4. Power tree, Arduino UNO 9
5. Configuración del botón 10
6. Diseño final del circuito 12
7. Divisor de tensión 14
8. Conversión [-24,24]V a [0,5]V 15
9. Funcionamiento en DC Mode 18
10. Funcionamiento en AC Mode 19
11. Transmisión Serial en funcionamiento 20
12. Archivo csv 21

Índice de tablas

1. Introducción

Resumen

En este laboratorio, se diseña y desarrolla un voltímetro de 4 canales utilizando un microcontrolador Arduino UNO. El sistema es capaz de medir tensiones en un rango de -24V a 24V, tanto en corriente alterna (AC) como continua (DC), seleccionable mediante un interruptor. Los valores medidos se visualizan en una pantalla LCD PCD8544, y se activa una luz LED de precaución cuando las tensiones superan los ±20V.

El diseño incluye el uso de componentes pasivos y un divisor de tensión para asegurar que las lecturas sean compatibles y seguras para el Arduino. Adicionalmente, los datos obtenidos se transmiten a una computadora a través del puerto serial utilizando el protocolo USART. Una vez en la computadora, los datos se almacenan en un archivo .csv mediante un script en Python.

Este proyecto utiliza los GPIOs del Arduino, el conversor ADC para la lectura de las tensiones, y los protocolos de comunicación USART y SPI para el manejo y almacenamiento de los datos.

2. Nota teórica

2.1. Arduino UNO

El Arduino Uno es una de las placas más populares y versátiles de la plataforma Arduino. Está basada en el microcontrolador ATmega328P y es ideal para principiantes en el mundo de la programación y la electrónica, así como para proyectos más avanzados en robótica, automatización y sistemas embebidos.

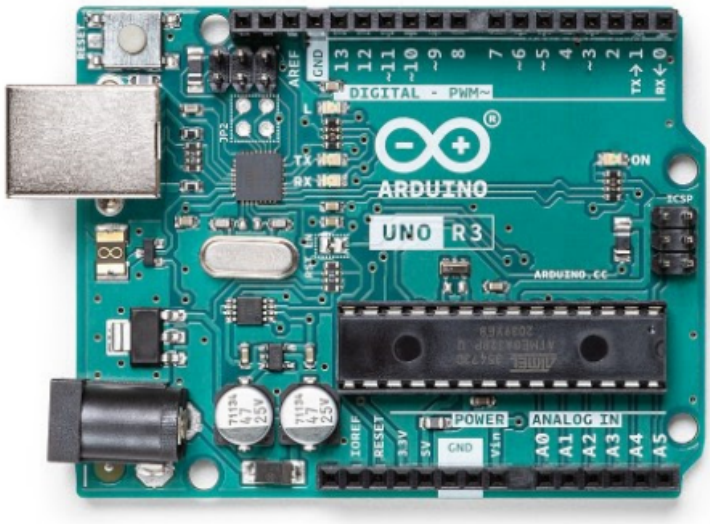


Figura 1: Arduino UNO

Características Generales:

- Microcontrolador: ATmega328P
- Voltaje de funcionamiento: 5V
- Voltaje de entrada (recomendado): 7-12V
- Voltaje de entrada (límites): 6-20V
- Pines digitales I/O: 14 (de los cuales 6 se pueden usar como salidas PWM)
- Pines analógicos: 6 (resolución de 10 bits)
- Corriente máxima por pin I/O: 20 mA
- Corriente máxima del pin 3.3V: 50 mA
- Memoria flash: 32 KB (0.5 KB ocupados por el bootloader)
- SRAM: 2 KB

- EEPROM: 1 KB
- Frecuencia de reloj: 16 MHz
- Interfaces de comunicación: UART, I2C, SPI

2.2. Diagrama de pines

El Arduino Uno cuenta con un total de 28 pines disponibles para su uso.

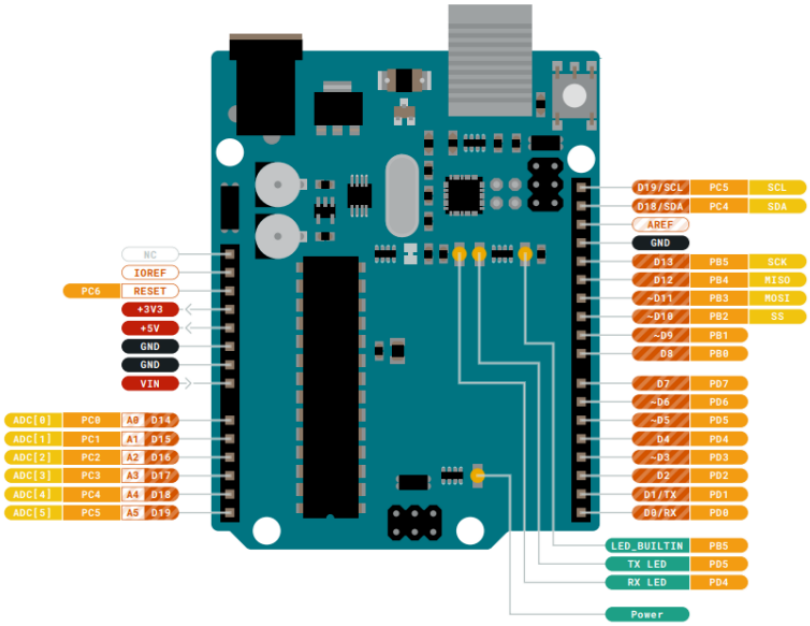


Figura 2: Diagrama de pines, Arduino UNO

2.2.1. Pines de Alimentación:

- **VCC**: Proporciona un voltaje de 5V para alimentar sensores y otros dispositivos.
- **GND**: Conexión a tierra (disponible en varios pines).
- **3.3V**: Salida de 3.3V para dispositivos de baja potencia.
- **Vin**: Voltaje de entrada externa (puede ir desde 6V hasta 12V).

2.2.2. Pines Digitales (D0 - D13):

- **D0 (RX) y D1 (TX)**: Pines de comunicación UART.
- **D2 - D13**: Pines de entrada/salida digital. Se pueden configurar como entradas o salidas.
- **D3, D5, D6, D9, D10, D11**: Pines que soportan salidas PWM.

2.2.3. Pines Analógicos (A0 - A5):

- Son pines de entrada analógica que permiten leer señales con una resolución de 10 bits (valores de 0 a 1023).

2.2.4. Pines de Comunicación:

- **SPI**: Pines 10 (SS), 11 (MOSI), 12 (MISO) y 13 (SCK) se usan para comunicación SPI.
- **I2C**: Pines A4 (SDA) y A5 (SCL) se usan para comunicación I2C.

2.3. Descripción de pines:

- **Pines Digitales**: Los pines D0 a D13 son pines de propósito general que pueden configurarse como entradas o salidas. Algunos soportan funciones PWM (modulación por ancho de pulso).
- **Pines Analógicos**: Los pines A0 a A5 pueden leer señales analógicas con una resolución de 10 bits. También pueden usarse como pines digitales adicionales.
- **Pines de Comunicación**: Además de los pines para la interfaz USB, los pines 0 y 1 (RX/TX) se usan para comunicación UART, y los pines A4/A5 para comunicación I2C.
- **Pines de Alimentación**: VCC y GND son esenciales para la alimentación de circuitos externos. El pin Vin permite alimentar el Arduino desde una fuente externa de voltaje.

2.4. Mapa de memoria de datos

La memoria del ATmega328P se divide en tres secciones:

- **Memoria Flash (32 KB)**: Aquí se almacena el código del programa. De esos 32 KB, 0.5 KB son utilizados por el bootloader.
- **SRAM (2 KB)**: Es la memoria de acceso aleatorio, usada por las variables del programa durante su ejecución.
- **EEPROM (1 KB)**: Es una memoria no volátil utilizada para almacenar datos que deben persistir después de apagar el Arduino. Puede ser leída y escrita a voluntad.

2.5. Especificaciones eléctricas

- **Voltaje de operación**: 5V.
- **Voltaje de entrada**: 7-12V (recomendado), hasta 20V como límite.
- **Corriente máxima por pin digital**: 40 mA (recomendado no exceder 20 mA por pin).
- **Corriente total de salida para todos los pines I/O**: Máximo 200 mA.
- **Corriente del pin 3.3V**: 50 mA máximo.
- **Protección de sobrecorriente**: El Arduino incluye un fusible de restablecimiento automático en la entrada USB para evitar daños por sobrecarga.

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 3: Power Consumption, Arduino UNO

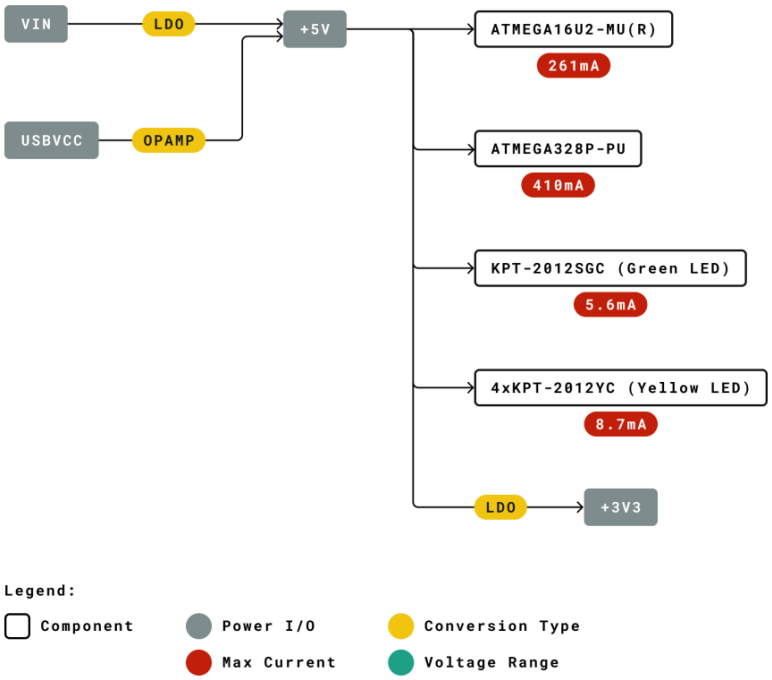


Figura 4: Power tree, Arduino UNO

2.6. Periféricos

2.6.1. Interrupciones

El Arduino UNO cuenta con dos pines dedicados para interrupciones: el pin 2 y el pin 3. Para configurar una interrupción, se utiliza el método `attachInterrupt(interrupt, ISR, mode)`, donde:

- `interrupt` puede ser 0 o 1, lo que corresponde a los pines 2 y 3 respectivamente.
- `ISR` es la función que se ejecutará cuando se detecte la interrupción.
- `mode` define cuándo se activará la interrupción y puede tener los siguientes valores:
  - `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).

2.7. Botón/Switch

Para llevar a cabo la implementación del botón, se utiliza una red de pull-up. Una red de pull-up es un conjunto de resistencias conectadas entre una línea de señal y el voltaje de alimentación positivo (VCC). Su función principal es garantizar que los pines de entrada de un microcontrolador, se mantengan en un estado lógico alto (1) cuando no están siendo activados por ningún otro dispositivo. Esto evita que el pin quede en un estado flotante o indeterminado, lo que podría causar un comportamiento impredecible. Las resistencias pull-up son comunes en pines de entrada como los de comunicación I2C o botones.

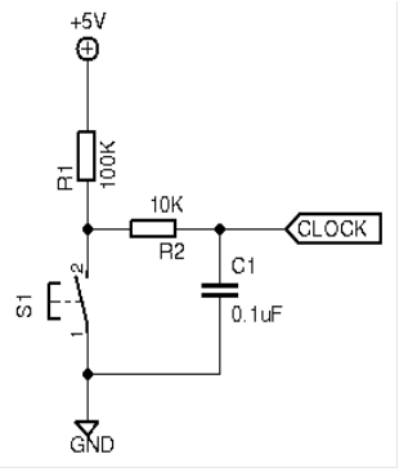


Figura 5: Configuración del botón

2.8. Pantalla LCD PCD8544

La pantalla LCD PCD8544 es una pantalla gráfica de bajo consumo utilizada comúnmente en dispositivos portátiles como el Nokia 5110. Tiene una resolución de 84x48 píxeles, lo que permite mostrar texto y gráficos simples. Está basada en el controlador PCD8544, que facilita su manejo a través de protocolos de comunicación como SPI. Esta pantalla es ideal para proyectos con microcontroladores debido a su bajo costo y eficiencia energética, funcionando con un voltaje de entre 2.7V y 3.3V. Es ampliamente utilizada en aplicaciones que requieren una interfaz gráfica compacta y fácil de controlar.

Carácteristicas generales:

- Resolución: 84x48 píxeles.
- Controlador: PCD8544, que gestiona la comunicación con el microcontrolador.
- Comunicación: Protocolo SPI de 4 o 5 pines (SCLK, DIN, D/C, CS, RST).
- Voltaje de operación: Entre 2.7V y 3.3V.
- Consumo de corriente: Muy bajo, ideal para aplicaciones portátiles (alrededor de 200 µA).
- Tipo de pantalla: Monocromática (solo blanco y negro).
- Dimensiones: Aproximadamente 45 mm x 45 mm.

- Retroiluminación: A menudo tiene retroiluminación LED, que se puede controlar por separado.
- Facilidad de uso: Soporta librerías ampliamente disponibles en plataformas como Arduino, lo que facilita su implementación en proyectos.

2.9. Convertidor analógico a digital (CAD)

Un convertidor analógico-digital (ADC) es un dispositivo o circuito que convierte señales analógicas continuas, como voltajes, en valores digitales que pueden ser procesados por un microcontrolador o una computadora. El ADC toma muestras de la señal analógica en intervalos regulares y las convierte en valores binarios, que representan niveles discretos de la señal original. La resolución del ADC, generalmente medida en bits, determina la precisión de la conversión, mientras que la tasa de muestreo indica la velocidad a la que se capturan las muestras. Es esencial en sistemas que necesitan interpretar señales del mundo real, como sensores.

2.10. Protocoloes de comunicación

2.10.1. USART (Universal Synchronous/Asynchronous Receiver-Transmitter)

El USART es un protocolo de comunicación que puede operar tanto de forma sincrónica como asincrónica. En su modo más común, el UART (asincrónico), se utiliza para intercambiar datos seriales entre dos dispositivos sin la necesidad de un reloj compartido.

Pines:

- TX (Transmisión): Línea de datos para enviar información.
- RX (Recepción): Línea de datos para recibir información.

Características principales:

- Comunicación half-dúplex (envío y recepción no simultáneos en UART).
- Transmite los datos en forma de bits individuales (start bit, data bits, optional parity bit, stop bit).
- Incluye control de errores mediante bits de paridad y control de flujo con señales RTS/CTS.
- Velocidad moderada de transmisión comparada con SPI, pero con mayor robustez.

2.10.2. SPI (Serial Peripheral Interface)

El SPI es un protocolo de comunicación serial sincrónico que permite la transferencia de datos de alta velocidad entre un microcontrolador y periféricos como sensores, pantallas y memorias. Utiliza un esquema maestro-esclavo, donde el maestro controla la comunicación. Se basa en cuatro líneas principales:

- MISO (Master In Slave Out): Línea de datos desde el esclavo al maestro.
- MOSI (Master Out Slave In): Línea de datos desde el maestro al esclavo.

- SCK (Serial Clock): Señal de reloj generada por el maestro para sincronizar la transmisión.
- SS (Slave Select): Señal que habilita la comunicación con un esclavo específico.

Características principales:

- Comunicación full-dúplex (envío y recepción simultánea).
- Alta velocidad de transmisión.
- Puede conectar múltiples dispositivos esclavos.
- No incluye verificación de errores ni control de flujo, por lo que es más rápido pero menos robusto que otros protocolos.

2.11. Diseño del circuito

A continuación se muestra el diseño final del circuito:

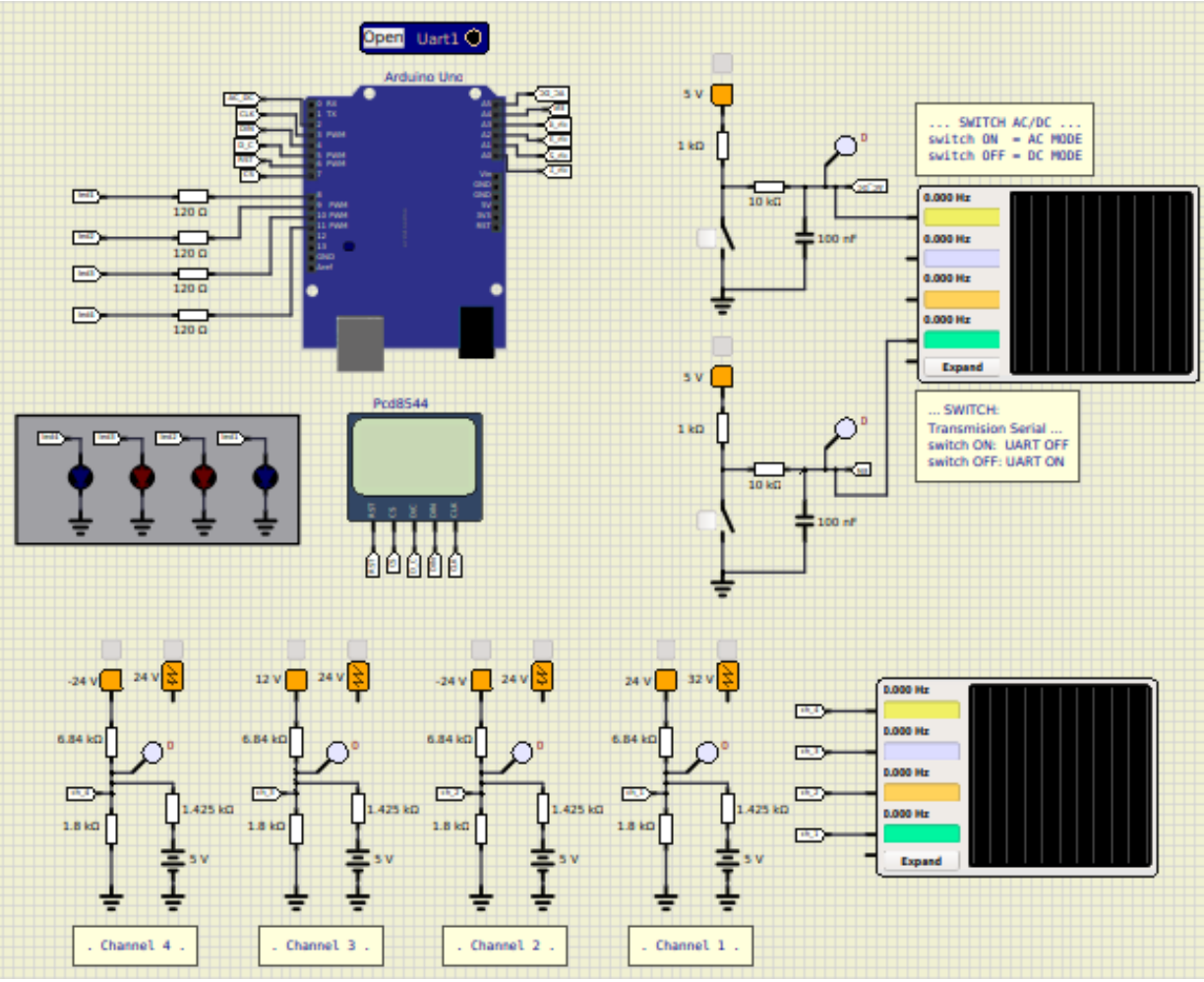


Figura 6: Diseño final del circuito

3. Desarrollo

3.1. Análisis de resultados

3.1.1. Resistencias de protección para los LED's

En cuanto a las resistencias de protección; para llevar a cabo la escogencia, se hace una LTK, con un umbral de 2.2 V, dado que el rango de valores de tensión varía de acuerdo al color, se opta por un valor típico de 2.2 V por LED, y con una corriente de 20 mA (valor típico), Ahora considerando una tensión ideal de 5 V. de pin (sin pérdidas) se tiene:

$$-5 + V_{R_{led}} + 2{,}2 = 0$$

Al sustituir $V_{R_{led}}$ con ley de Ohm, y despejar para encontrar el valor de la resistencia R_{led} se tiene lo siguiente:

$$R_{led} = \frac{5 - 2{,}2}{20x10^{-3}}$$

$$R_{led} = 140\Omega$$

Pero como 140 Ω no es un valor comercional, se opta por colocar resistencias de 120 Ω, por lo tanto:

$$R_{led} = 120\Omega$$

3.1.2. Efecto rebote

Para suprimir el efecto rebote que existe al pulsar el botón, se opta por escoger un capacitor comercial típico, de forma que sea capaz de filtrar posibles afectaciones.

Por lo tanto, de acuerdo al capacito C = 0.1uF (100nF) y una resistencia comercial típica R = 10KΩ, se obtiene la constante de tiempo τ.

Por lo tanto de acuerdo a la red RC que se forma, se tiene que:

$$\tau = R \cdot C$$

$$\tau = 10000 \cdot (100x10^{-9})$$

$$\tau = 1ms$$

Este tiempo es suficiente para cumplir con un adecuado funcionamiento.

3.1.3. Convertir [-24, 24]V a [0, 5]V

Uno de los requisitos es que el voltímetro debe ser capaz de medir tensiones en un rango de -24V a 24V. Sin embargo, esto presenta problemas que necesitan ser abordados. Los pines analógicos del Arduino solo pueden leer voltajes en el rango de 0V a 5V, lo que implica la necesidad de convertir la tensión para que pueda ser correctamente leída, permitiendo la conversión de un voltaje que varía de -24V a 24V a uno que oscila entre 0v y 5V., como se observa en la siguiente figura:

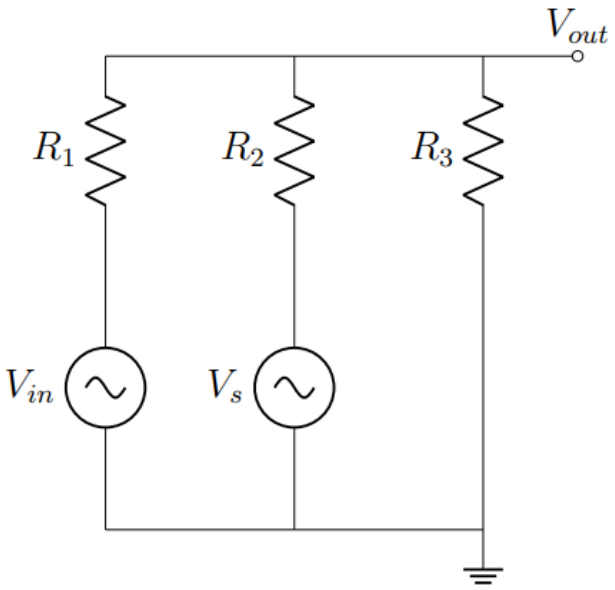


Figura 7: Divisor de tensión

Se obtiene la expresión para la salida V_{out}, utilizando el teorema de superposición. En este enfoque, se desactivan las fuentes independientes una a una, para evaluar la contribución de cada una sobre la salida (V_{out}).

Primero, se analiza una fuente apagada y obtenemos el siguiente equivalente, el cúal por medio de una LCK, ley de Ohm y del divisor de tensión se obtiene:

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

De la misma forma, se aplicar el mismo procedimiento para con la otra fuente, de donde se obtiene:

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

Por lo tanto, sumando la contribución de ambas fuentes se obtiene:

$$V_{out} = V_{out1} + V_{out2} \tag{3}$$

y sustituyendo ambas ecuaciones se obtiene que:

$$V_{out} = \frac{R_1||R_3}{(R_1||R_3) + R_2} \cdot V_s + \frac{R_2||R_3}{(R_2||R_3) + R_1} \cdot V_{in} \tag{4}$$

Por lo tanto se tiene las siguientes condiciones: cuando V_{in} = 24V $\Rightarrow$ V_{out} = 5V y por su parte cuando V_{in} = -24V $\Rightarrow$ V_{out} = 0V, por lo que se tienen dos ecuaciones y cuatro incógnitas, de donde se escogen dos variables y las otras dos incógnitas se resulven por medio del sistema de ecuaciones, por lo tanto, para efectos prácticas, se tiene que $R_3 = 1.8\text{k}\Omega$ (es un valor comercial útil en este caso) y $V_s = 5$, con estas condiciones se tiene que el sistema de ecuaciones es:

$$5 = \frac{R_1||1{,}8\text{K}\Omega}{(R_1||1{,}8\text{K}\Omega) + R_2} \cdot 5 + \frac{R_2||1{,}8\text{K}\Omega}{(R_2||1{,}8\text{K}\Omega) + R_1} \cdot 24 \tag{5}$$

$$0 = \frac{R_1||1{,}8\text{K}\Omega}{(R_1||1{,}8\text{K}\Omega) + R_2} \cdot 5 + \frac{R_2||1{,}8\text{K}\Omega}{(R_2||1{,}8\text{K}\Omega) + R_1} \cdot -24 \tag{6}$$

por lo tanto, resolviendo con $R_3 = 1.8\text{K}\Omega$ y $V_{in} = 5\text{V}$, se obtiene:

$$R_1 = 6840\Omega = 6{,}84\text{K}\Omega$$

$$R_2 = 1425\Omega = 1{,}425\text{K}\Omega$$

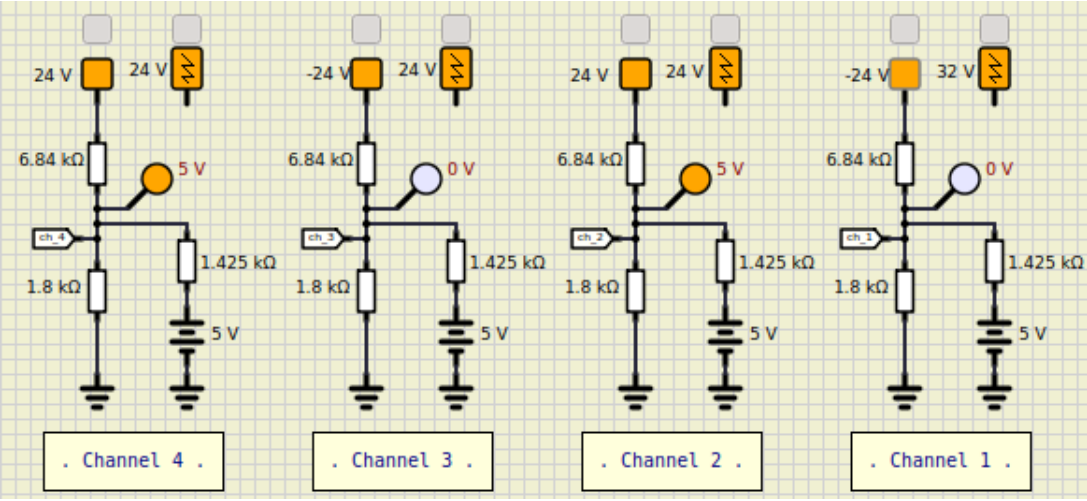


Figura 8: Conversión [-24,24]V a [0,5]V

3.2. Análisis del Firmware

3.2.1. Variables de interés

```
const int inputPins[4] = {A0, A1, A2, A3};
const int outputPins[4] = {8, 9, 10, 11};
```

- inputPins: Arreglo que define los pines analógicos utilizados para leer voltajes.
- outputPins: Arreglo que define los pines de salida para activar alertas.

3.3. Función get_RMS(int PIN)

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

```
    for(int i = 0; i < samples; i++) {
        float t = analogRead(PIN);
        if(t > tmax) tmax = t;
    }

    float t_rms = ((tmax - factor) * (ref_Voltaje / factor));
    return (t_rms * 0.7071);
}
```

Calcula el valor RMS: Realiza 1000 lecturas del pin analógico dado, guarda la lectura máxima y calcula el voltaje RMS.

3.4. Funciones de Modo

```
    void AC_mode() {
    // calcular la tensión RMS para cada canal
    for (int i = 0; i < 4; i++) {
        voltages[i] = get_RMS(inputPins[i]);
    }

    warning();
    lcd_Display("AC MODE:", voltages);
    serial_Display("AC MODE", voltages);
}
```

AC_mode: Lee voltajes AC y actualiza las visualizaciones.

```
    void DC_mode() {
    // calcular los valores analógicos y calcular el voltaje para cada canal
    for (int i = 0; i < 4; i++) {
        voltages[i] = (analogRead(inputPins[i]) - factor) * (ref_Voltaje / factor);
    }

    warning();
    lcd_Display("DC MODE", voltages);
    serial_Display("DC MODE", voltages);
}
```

DC_mode: Lee voltajes DC y actualiza las visualizaciones.

3.5. Función warning()

```
    void warning() {
```

```
    // encender ls warnings de alto voltaje
    for (int i = 0; i < 4; i++) {
        if(voltages[i] > 20.00 || voltages[i] < -20.00) {
            digitalWrite(outputPins[i], HIGH);
        } else {
            digitalWrite(outputPins[i], LOW);
        }
    }
}
```

Activa las advertencias: Enciende los pines de advertencia si el voltaje excede ciertos límites

3.6. Bucle Principal loop()

```
    void loop() {
    float switch_ACDC = analogRead(A5);
    if(switch_ACDC < factor) {
        AC_mode();
    } else {
        DC_mode();
    }
}
```

Selecciona el modo de operación: Lee el valor del pin A5 para decidir si debe estar en modo AC o DC, y llama a la función correspondiente.

3.7. Análisis electrónico

En base a las siguiente figuras, se puede demostrar el correcto funcionamiento del **Voltmeter** de 4 canales, y de como la electrónica aplicada presenta un correcto funcionamiento, así como la transmisión serial de datos con respecto a otro dispositivo.

Cabe recalcar que para observar un correcto funcionamiento, se tiene que tener ejecutado el programa, ya que con solo figuras en difícil demostrar toda la funcionalidad del voltímetro, por lo que solo se presentan capturas que demuestran el funcionamiento de los pasos más importantes.

3.7.1. DC Mode

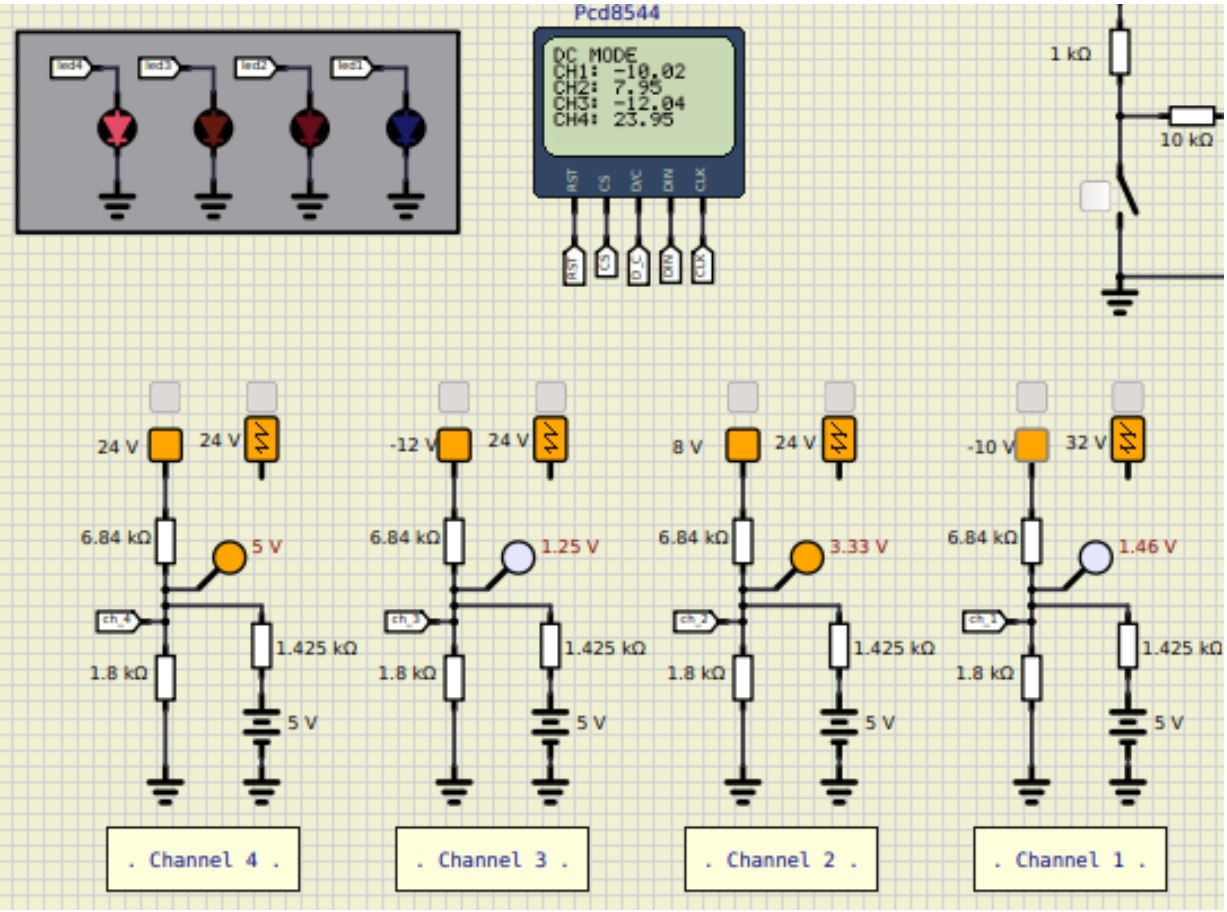


Figura 9: Funcionamiento en DC Mode

3.7.2. AC Mode

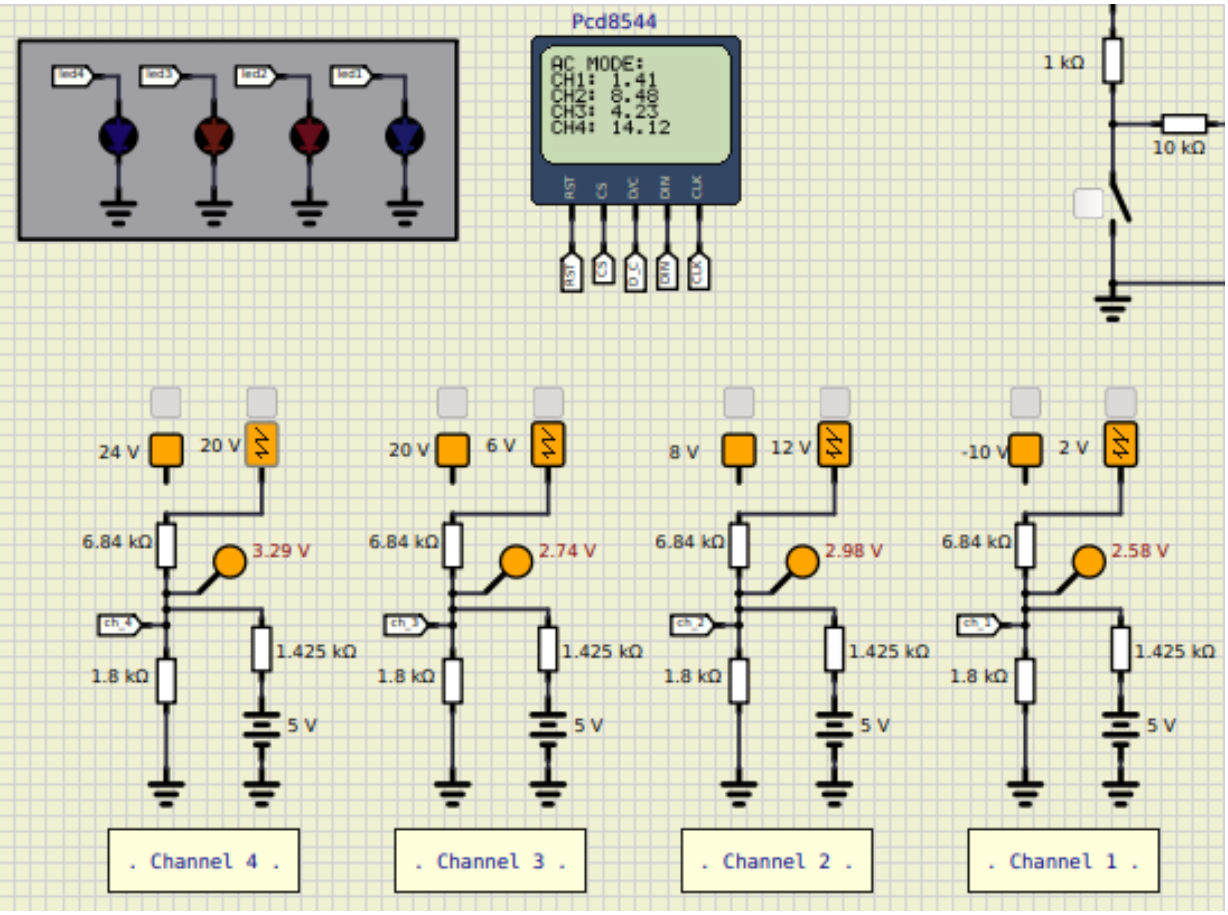


Figura 10: Funcionamiento en AC Mode

3.7.3. Transmisión Serial

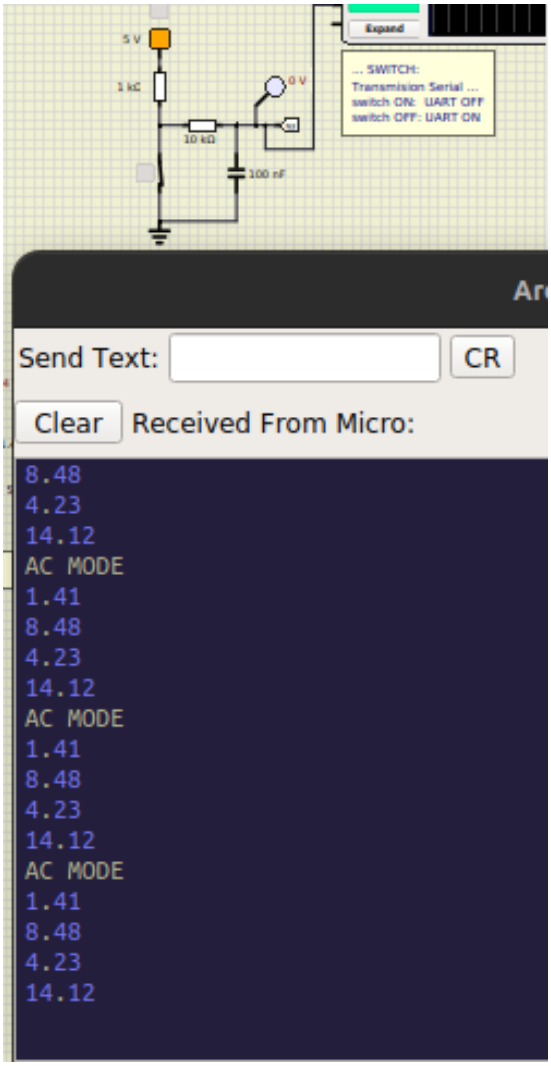
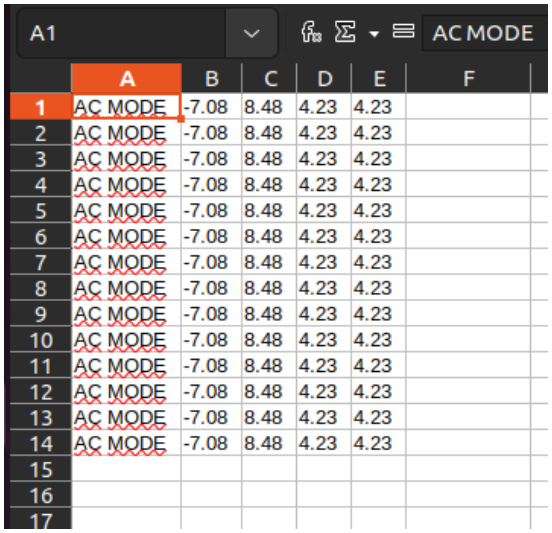


Figura 11: Transmisión Serial en funcionamiento

3.7.4. archivo .csv



	A	B	C	D	E	F
1	AC MODE	-7.08	8.48	4.23	4.23	
2	AC MODE	-7.08	8.48	4.23	4.23	
3	AC MODE	-7.08	8.48	4.23	4.23	
4	AC MODE	-7.08	8.48	4.23	4.23	
5	AC MODE	-7.08	8.48	4.23	4.23	
6	AC MODE	-7.08	8.48	4.23	4.23	
7	AC MODE	-7.08	8.48	4.23	4.23	
8	AC MODE	-7.08	8.48	4.23	4.23	
9	AC MODE	-7.08	8.48	4.23	4.23	
10	AC MODE	-7.08	8.48	4.23	4.23	
11	AC MODE	-7.08	8.48	4.23	4.23	
12	AC MODE	-7.08	8.48	4.23	4.23	
13	AC MODE	-7.08	8.48	4.23	4.23	
14	AC MODE	-7.08	8.48	4.23	4.23	
15						
16						
17						

Figura 12: Archivo csv

4. Conclusiones y Recomendaciones

4.1. Conclusiones

- Se obtiene mas conocimiento acerca del funcionamiento del Arduino.
- Se implementa exitosamente un medidor de voltajes tanto en AC como en DC.
- Se aprende mas sobre los protocolos de comunicación con respecto al Arduino.

4.2. Recomendaciones

- Como recomendación personal, siempre es bueno tener previamente un diseño inicial, con lo cual lograr ubicarse un poco en contexto, claramente este diseño inicial es casi seguro que cambie, pero al menos sirve de punto inicial.
- Tener siempre a mano la hoja de fabricante, ya que básicamente es nuestra guía de trabajo.
- Siempre verificar las conexiones, ya que al trabajar con muchos dispositivos, puede ocurrir que algún componente no está bien conectado y puede producir errores de HW y no de firware.
- Trabajar con orden y por partes, es bueno dividir el trabajo en secciones e ir poco a poco avanzando, y cada vez que se pueda, realizar pruebas.

Bibliografía

1. Material de clase

2. Arduino. (n.d.). Arduino Uno. Recuperado de https://www.arduino.cc/en/Guide/ArduinoUno

3. O'Connor, K. (2016). Using the Nokia 5110 LCD with Arduino. In Arduino Cookbook (p. 212). O'Reilly Media.

4. Rivas, A. (2019). Fundamentals of Serial Communication: USART and SPI. Recuperado de https://www.electronics-tutorials.ws/io/io_7.html

5. Horowitz, P., & Hill, W. (2015). The Art of Electronics (3rd ed.). Cambridge University Press.

6. Alexander, C. K., & Sadiku, M. N. O. (2018). Fundamentals of Electric Circuits (6th ed.). McGraw-Hill.

5. Apéndices