Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

IE-0624 Laboratorio de Microprocesadores

Laboratorio 3

José Antonio Ramos Pereira, B86485

Grupo 01

Prof. MSc. Marco Villalta Fallas.

https://github.com/JAR1224/Laboratorio_3_Jose_Ramos

Índice de contenidos

1. Introducción

1.1. Resumen

La elaboración del proyecto consistió en plantear una solución utilizando una tarjeta de desarrollo Arduino Uno y otros componentes electrónicos para simular un voltímetro de 4 canales. La aplicación debía realizar las siguientes funciones:

- Medir al mismo tiempo los voltajes de los 4 canales
- Condicionar los voltajes de entrada de [-24,24]V a un rango de voltaje que el ADC del Arduino pueda manejar ([0,5]V)
- Incluir un switch para configurar el modo de medición (AC o DC)
- En caso de medir un voltaje menor a -20V o superior a 20V debe encender un LED de alarma correspondiente al canal de entrada.
- Procesar la señal de entrada para escalar y mostrar en la pantalla LCD los valores reales.
- Los datos obtenidos se enviarán hacia la computadora a través del puerto serial con el bloque USART
- La transmisióon serial será controlada por un switch
- Se tendrá un programa en python que leerá el puerto serial y guardará el registro de datos como un archivo en formato CSV

Los retos principales para la elaboración de la solución fueron los siguientes:

1. Cálculo del valor RMS de la tensión en modo AC
2. Condicionar los niveles de voltaje entre -24 y 24 a un rango de 0 a 5.

Para poder superar los retos, se realizaron los siguientes pasos:

1. Muestrear la señal AC y por medio de dos variables, identificar cambios en el valor pico de la señal y calcular el valor RMS a partir de este dato.

2. Acoplar un divisor de tensión y un amplificador operacional en modo additivo.

1.2. Conclusiones

Se concluyó que por medio de las metodologías mencionadas anteriormente, se pudo implementar una solución al problema planteado que cumple con todos los requisitos de diseño requeridos por el enunciado.

2. Nota Teórica

2.1. Microcontrolador

Características generales

Las características generales del Arduino Uno son las siguientes [1]:

- **ATMega328P** Processor
  - **Memory**
    - AVR CPU at up to 16 MHz
    - 32KB Flash
    - 2KB SRAM
    - 1KB EEPROM
  - **Security**
    - Power On Reset (POR)
    - Brown Out Detection (BOD)
  - **Peripherals**
    - 2x 8-bit Timer/Counter with a dedicated period register and compare channels
    - 1x 16-bit Timer/Counter with a dedicated period register, input capture and compare channels
    - 1x USART with fractional baud rate generator and start-of-frame detection
    - 1x controller/peripheral Serial Peripheral Interface (SPI)
    - 1x Dual mode controller/peripheral I2C
    - 1x Analog Comparator (AC) with a scalable reference input
    - Watchdog Timer with separate on-chip oscillator
    - Six PWM channels
    - Interrupt and wake-up on pin change
  - **ATMega16U2 Processor**
    - 8-bit AVR® RISC-based microcontroller
  - **Memory**
    - 16 KB ISP Flash
    - 512B EEPROM
    - 512B SRAM
    - debugWIRE interface for on-chip debugging and programming
  - **Power**
    - 2.7-5.5 volts

Características Generales 2.1: Obtenido de [1]

Las características generales del Atmega328P son las siguientes [2]:

- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
  - 131 powerful instructions – most single clock cycle execution
  - 32 × 8 general purpose working registers
  - Fully static operation
  - Up to 16MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
  - 32K bytes of in-system self-programmable flash program memory
  - 1Kbytes EEPROM
  - 2Kbytes internal SRAM
  - Write/erase cycles: 10,000 flash/100,000 EEPROM
  - Optional boot code section with independent lock bits
    - In-system programming by on-chip boot program
    - True read-while-write operation
  - Programming lock for software security
- Peripheral features
  - Two 8-bit Timer/Counters with separate prescaler and compare mode
  - One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
  - Real time counter with separate oscillator
  - Six PWM channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature measurement
  - Programmable serial USART
  - Master/slave SPI serial interface
  - Byte-oriented 2-wire serial interface (Phillips I^2C compatible)
  - Programmable watchdog timer with separate on-chip oscillator
  - On-chip analog comparator
  - Interrupt and wake-up on pin change
- Special microcontroller features
  - Power-on reset and programmable brown-out detection
  - Internal calibrated oscillator
  - External and internal interrupt sources
  - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby

Características Generales 2.2: Obtenido de [2]

Diagrama de bloques Atmega328P

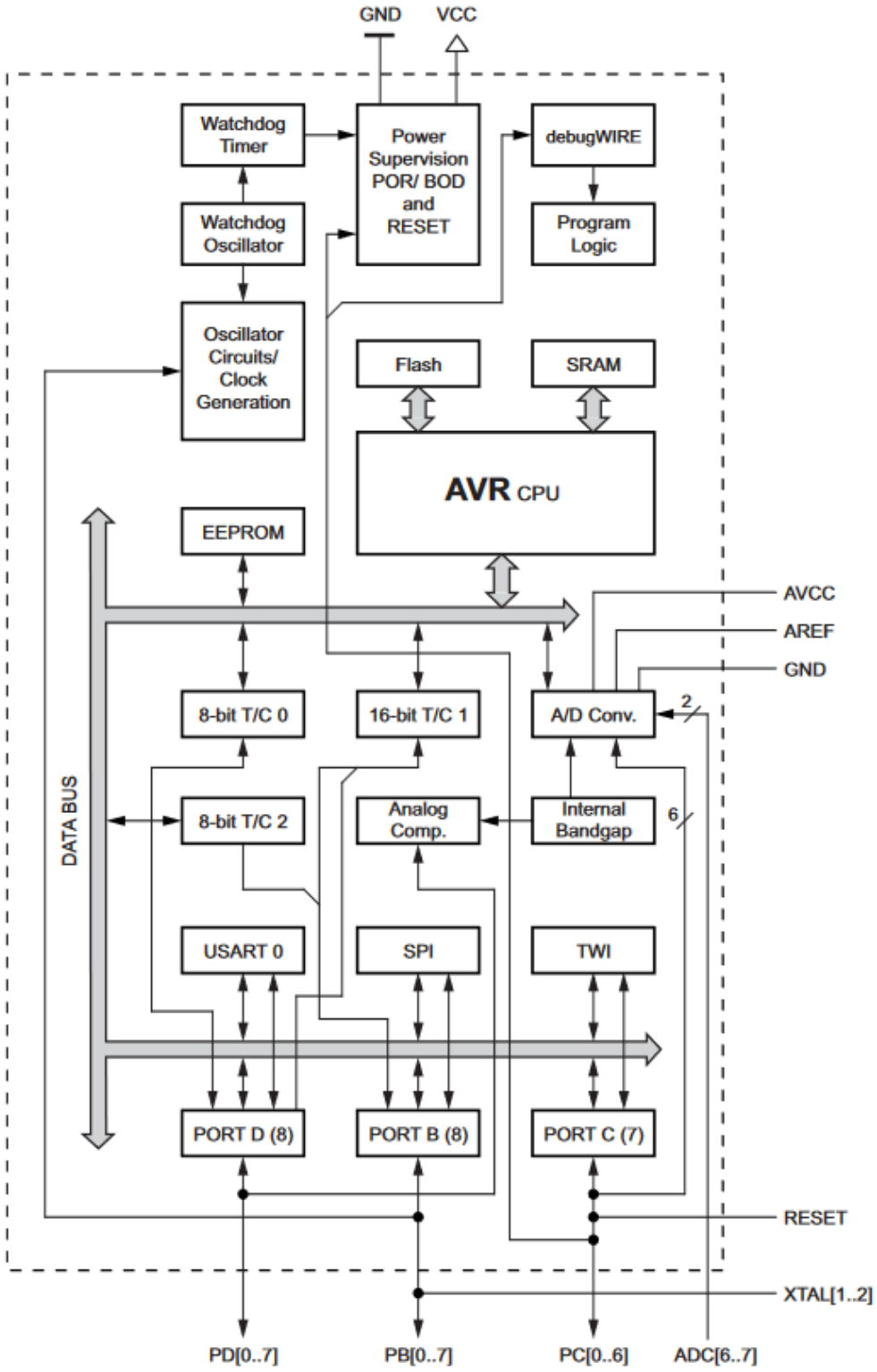


Diagrama de bloques 2.3: Obtenido de [2]

Diagrama de pines Arduino Uno

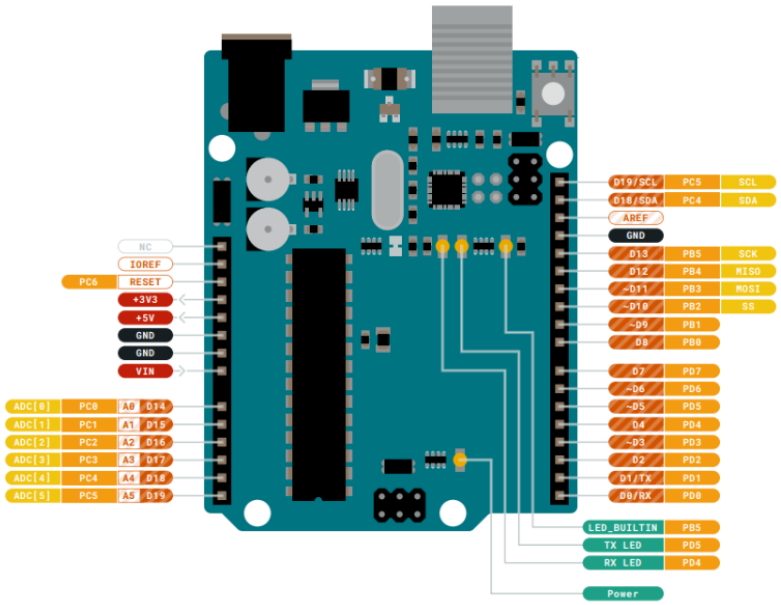


Diagrama de Pines Arduino Uno 2.4: Obtenido de [1]

Características eléctricas

Parameters	Min.	Typ.	Max.	Unit
Operating temperature	–55		+125	°C
Storage temperature	–65		+150	°C
Voltage on any pin except $\overline{\text{RESET}}$ with respect to ground	–0.5		V_{CC} + 0.5	V
Voltage on $\overline{\text{RESET}}$ with respect to ground	–0.5		+13.0	V
Maximum operating voltage		6.0		V
DC current per I/O pin		40.0		mA
DC current V_{CC} and GND pins		200.0		mA
Injection current at V_{CC} = 0V		±5.0[1]		mA
Injection current at V_{CC} = 5V		±1.0		mA

Note: 1. Maximum current per port = ±30mA

Características eléctricas 2.5: Obtenido de [2]

Registros

Los registros que se utilizaron para la configuración de las interrupciones de conversión analógica digital fueron los siguientes:

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registro 2.6: Obtenido de [2]

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8[1]

Bits MUX0...MUX3 2.7: Obtenido de [2]

Los bits que se configuraron para ADMUX fueron MUX0...MUX3. Estos 4 bits se utilizan para elegir cual canal se utiliza para la siguiente conversión.

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registro ADCSRA 2.8: Obtenido de [2]

Se configuraron los bits ADEN (para habilitar conversiones), ADSC (para iniciar una conversión) y ADIE (para habilitar la interrupción).

2.2. Componentes Electrónicos Complementarios

Pantalla PCD8544-64

1 FEATURES

- Single chip LCD controller/driver
- 48 row, 84 column outputs
- Display data RAM 48 × 84 bits
- On-chip:
  - Generation of LCD supply voltage (external supply also possible)
  - Generation of intermediate LCD bias voltages
  - Oscillator requires no external components (external clock also possible).
- External $\overline{RES}$ (reset) input pin
- Serial interface maximum 4.0 Mbits/s
- CMOS compatible inputs
- Mux rate: 48
- Logic supply voltage range V_{DD} to V_{SS}: 2.7 to 3.3 V
- Display supply voltage range V_{LCD} to V_{SS}
  - 6.0 to 8.5 V with LCD voltage internally generated (voltage generator enabled)
  - 6.0 to 9.0 V with LCD voltage externally supplied (voltage generator switched-off).
- Low power consumption, suitable for battery operated systems
- Temperature compensation of V_{LCD}
- Temperature range: –25 to +70 °C.

Características Generales 2.9: Obtenido de [3]

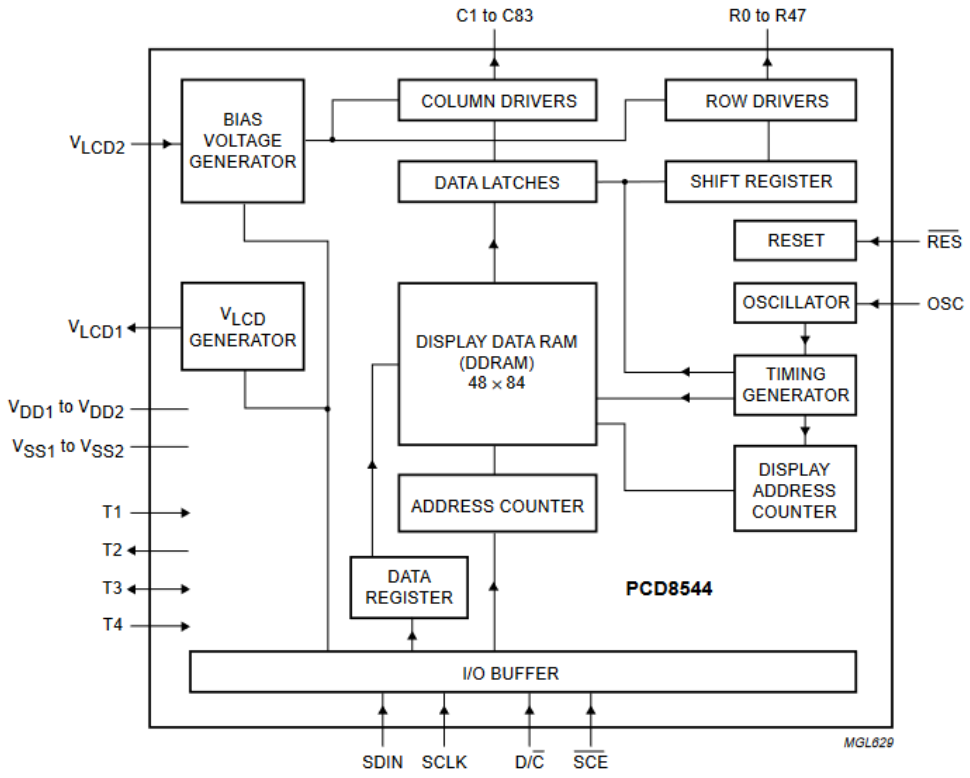


Diagrama de bloques 2.10: Obtenido de [3]

11 DC CHARACTERISTICS

V_{DD} = 2.7 to 3.3 V; V_{SS} = 0 V; V_{LCD} = 6.0 to 9.0 V; T_{amb} = –25 to +70 °C; unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
V_{DD1}	supply voltage 1	LCD voltage externally supplied (voltage generator disabled)	2.7	–	3.3	V
V_{DD2}	supply voltage 2	LCD voltage internally generated (voltage generator enabled)	2.7	–	3.3	V
V_{LCD1}	LCD supply voltage	LCD voltage externally supplied (voltage generator disabled)	6.0	–	9.0	V
V_{LCD2}	LCD supply voltage	LCD voltage internally generated (voltage generator enabled); note 1	6.0	–	8.5	V
I_{DD1}	supply current 1 (normal mode) for internal V_{LCD}	V_{DD} = 2.85 V; V_{LCD} = 7.0 V; f_{SCLK} = 0; T_{amb} = 25 °C; display load = 10 µA; note 2	–	240	300	µA
I_{DD2}	supply current 2 (normal mode) for internal V_{LCD}	V_{DD} = 2.70 V; V_{LCD} = 7.0 V; f_{SCLK} = 0; T_{amb} = 25 °C; display load = 10 µA; note 2	–	–	320	µA
I_{DD3}	supply current 3 (Power-down mode)	with internal or external LCD supply voltage; note 3	–	1.5	–	µA
I_{DD4}	supply current external V_{LCD}	V_{DD} = 2.85 V; V_{LCD} = 9.0 V; f_{SCLK} = 0; notes 2 and 4	–	25	–	µA
I_{LCD}	supply current external V_{LCD}	V_{DD} = 2.7 V; V_{LCD} = 7.0 V; f_{SCLK} = 0; T = 25 °C; display load = 10 µA; notes 2 and 4	–	42	–	µA
Logic						
V_{IL}	LOW level input voltage		V_{SS}	–	0.3V_{DD}	V
V_{IH}	HIGH level input voltage		0.7V_{DD}	–	V_{DD}	V
I_L	leakage current	V_I = V_{DD} or V_{SS}	–1	–	+1	µA
Column and row outputs						
$R_{o(C)}$	column output resistance C0 to C83		–	12	20	kΩ
$R_{o(R)}$	row output resistance R0 to R47		–	12	20	kΩ
$V_{bias(tol)}$	bias voltage tolerance on C0 to C83 and R0 to R47		–100	0	+100	mV

Características eléctricas 2.11: Obtenido de [3]

Resistencias

- 8x 1 MΩ
- 8x 6 kΩ
- 4x 8,6 kΩ
- 8x 100 Ω

Amplificador Operacional

- 4x Op Amp LMC7101

LEDS

- 4x diodos LED

Componentes

- 1x pantalla PCD8544

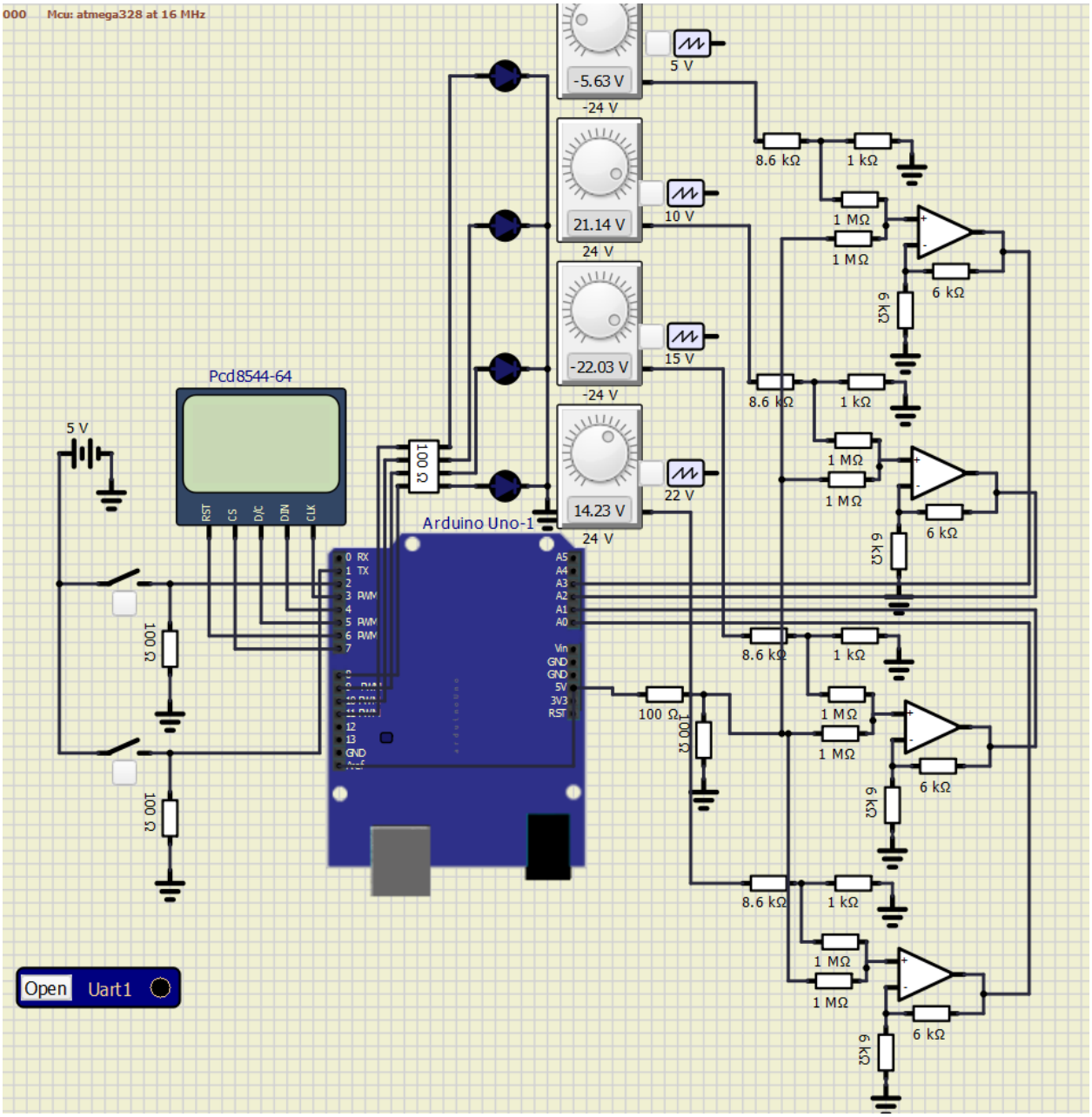
2.3. Precios

- 2x Op Amp LMC7101: $2.64
- 28x Resistencias: $1.00
- Arduino Uno: $26.71
- 4x diodos LED: $0.96
- 2x switches : $1.66
- 1x pantalla PCD8544: $8.95
- Total: **$41.92**

Obtenido de: [4]

2.4. Diseño del circuito

Esquemático



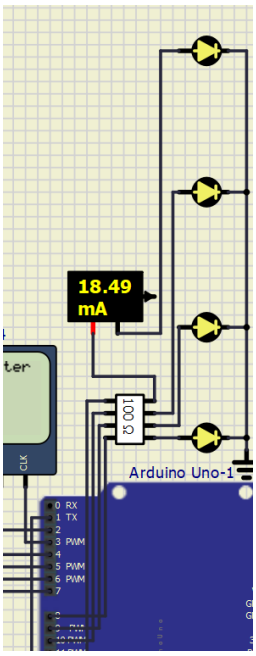
El circuito consta de 3 partes principales.

1) Switches

Hay 2 switches que se conectan a 2 pines IO del Arduino. Cada switch se encarga de: Habilitar comunicación serial y cambiar de modo DC-AC.

2) Diodos LED

Esta parte es una simple conexión de diodos LEDS con resistencias a puertos IO del Arduino para representar las alarmas de tensión para cada uno de los 4 canales de medición.



Circuito eq. LEDS 2.12: Circuito LEDS

Sabiendo que en la hoja del fabricante se incluye que la impedancia de salida de los pines es de 170 Ω. Y sabiendo que:

$$\begin{aligned} \frac{5V}{R_l + R_o} &< 20mA \\ 250\Omega &< R_l + R_o \end{aligned} \tag{1}$$

Entonces, eligiendo R_l=100Ω, se obtiene una impedancia total de 270Ω > 250Ω.

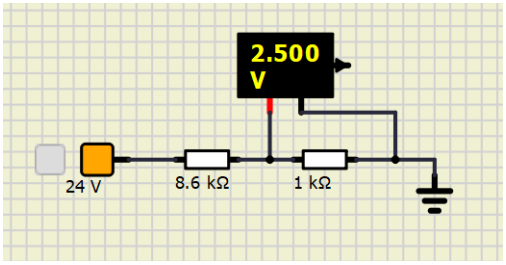
3) Divisores de tensión con etapa summatorio de amplificadores operacionales

Esta parte del circuito se encarga de convertir el rango de entrada de señales de [-24V, 24V] a [0V , 5V].

Para lograr esto primero se divide la tensión por un factor de 9,6, de esta forma el intervalo [-24V, 24V] se convierte a [-2,5V, 2,5V]. Para lograr esto se debe cumplir lo siguiente:

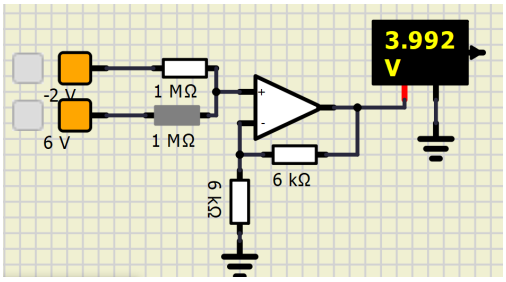
$$V_{out} = V_{in} \frac{R_2}{R_2 + R_1} \tag{2}$$

Para obtener el factor deseado de división, se elige $R_2 = 1k\Omega$ y $R_2 = 8,6k\Omega$:



Por último, se le suma a la tensión un valor de 2,5, transformando así el intervalo [-2,5V, 2,5V] a [0V, 5V].

Para esto se utilizan amplificadores operacionales en configuración summatoria:



2.5. Temas de laboratorio

USART

La UART (Universal Asynchronous Receiver/Transmitter) es un componente fundamental en la comunicación de datos digitales entre dispositivos electrónicos. Es una interfaz serial

asíncrona que permite la transmisión y recepción de datos binarios mediante un par de hilos, uno para enviar (TX) y otro para recibir (RX).

La transmisión UART se realiza mediante un proceso de envío de bits secuenciales, donde cada bit es precedido por un bit de inicio y seguido por uno o más bits de parada. La velocidad de transmisión se determina por una tasa de baudios, que representa la cantidad de veces por segundo que se transmiten los bits.

[5]

SPI

SPI (Serial Peripheral Interface) es un protocolo de comunicación sincrónico utilizado para la transferencia de datos entre dispositivos electrónicos. Se basa en una arquitectura de maestro y esclavo, donde un dispositivo maestro controla la comunicación con uno o varios dispositivos esclavos.

La comunicación SPI se realiza mediante la transmisión de bits secuenciales a través de cuatro líneas: la línea de reloj (SCK), la línea de selección de esclavo (SS), la línea de salida del maestro (MOSI) y la línea de entrada del maestro (MISO). El maestro genera una señal de reloj que se utiliza para sincronizar la transmisión de los bits de datos entre el maestro y el esclavo.

La velocidad de transmisión en SPI se determina por la frecuencia del reloj, que se puede configurar para operar a diferentes velocidades. Además, SPI permite la configuración de varios parámetros, como el modo de transmisión, la configuración de bit de datos y la activación de la interrupción. [5]

ADC

Las conversiones analógicas a digitales (ADC) en Arduino se realizan utilizando un conversor analógico a digital de 10 bits. El ADC mide el voltaje de entrada en la entrada analógica y lo convierte en un valor digital de 0 a 1023, donde 0 representa un voltaje de 0 voltios y 1023 representa un voltaje de referencia de 5 voltios. La precisión de la conversión

ADC se puede mejorar utilizando un voltaje de referencia externo o ajustando la resolución del conversor a través de la biblioteca de Arduino.

La biblioteca de Arduino proporciona funciones para realizar conversiones ADC de manera sencilla. Los usuarios pueden usar la función .ªnalogRead()"para leer el valor de voltaje en una entrada analógica determinada y convertirlo en un valor digital. Además, los usuarios pueden configurar la resolución del conversor y el voltaje de referencia utilizando las funciones .ªnalogReference()ƨ .ªnalogReadResolution()". [5]

3. Desarrollo y análisis

3.1. Programa

Descripción del programa

El programa consta de 2 partes principales: El main, que se encarga de constantemente actualizar los datos a imprimir en la pantalla, y la interrupción ADC, que se encarga de realizar la conversión para una medición AC o DC en los 4 canales disponibles, enviar el dato serial por UART e iniciar la siguiente conversión.

main() lee del pin 2 que está conectado al switch que determina si la medición se debe de hacer para AC o DC. Según la lectura de este pin, se va a indicar en la pantalla si se está midiendo valores en DC o en AC (rms).

La interrupción ADC se encarga de leer la conversión realizada previamente, indicar si este valor está dentro del rango seguro para esta aplicación, mandar el dato por el puerto serial y mandar a realizar la siguiente conversión en el siguiente canal escribiendo un 1 al bit ADSC.

La interrupción está configurada de manera que cada conversión individual se realiza cuando se escribe un 1 al bit ADSC del registro de control ADCSRA. Adicionalmente, se habilita la interrupción de fin de conversión.

Medición DC

Realizar la medición real DC es bastante sencillo, simplemente se transforma el valor obtenido en los registros del convertidor AD usando la siguiente fórmula:

$$ValorReal = (adcValue * 0{,}0469208) - 24 \tag{3}$$

Esto transforma el número guardado en adcValue, que va de 0 a 1023 a un número entre -24V y 24V de forma lineal. En principio, esta operación es el inverso a la transformación que está haciendo la etapa de divisor de tensión y summatoria de tensión en las entradas analógicas del Arduino.

Medición AC

La medición del valor AC en rms es más dificil de obtener. La solución propuesta en este laboratorio fue utilizar una variable para capturar el valor máximo en una ventana de tiempo.

Es decir, existe una variable que se utiliza para guardar el valor de tensión medido. Si la siguiente medición es mayor, la variable se sobrescribe, si no se mantiene igual. De esta forma, en un tiempo dado y con un muestreo lo suficientemente alto, se puede asegurar que se tiene guardado el valor pico de la señal AC.

Sinembargo, que pasa si cambia el valor rms de la señal AC? En este caso la solución anterior sería incapaz de reconocer el cambio. Para arreglar esto, se incorpora una ventana de tiempo que se encarga de reiniciar la variable cada cierto intervalo.

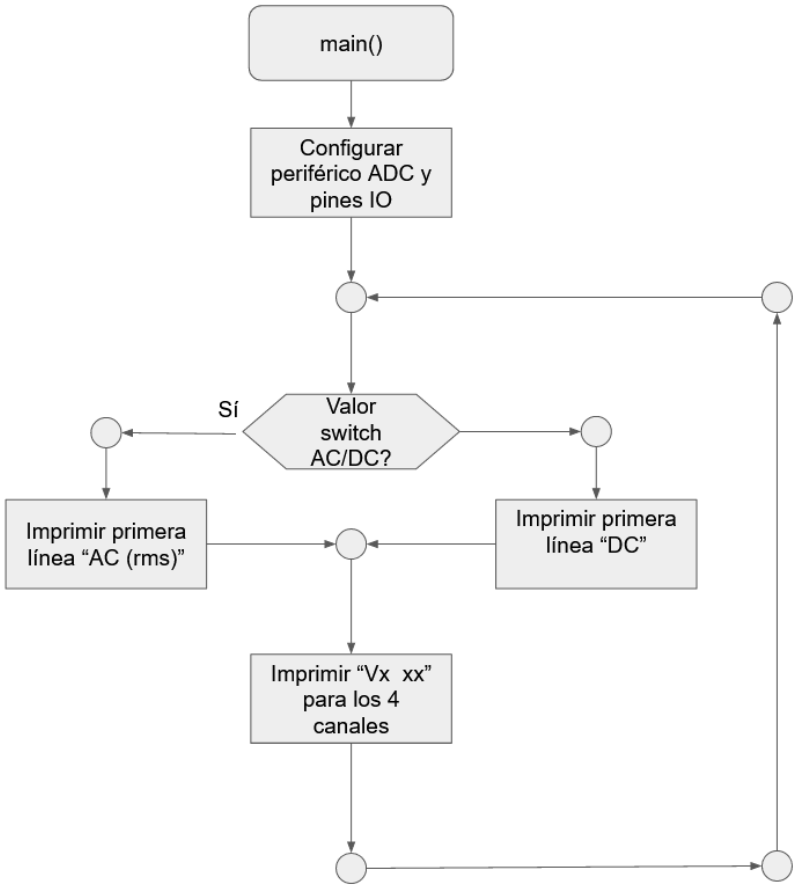
Para cada valor pico medido, el valor rms se obtiene dividiendo este valor por la raíz cuadrada de 2, que es aproximadamente igual que multiplicar el valor por 0,7071.

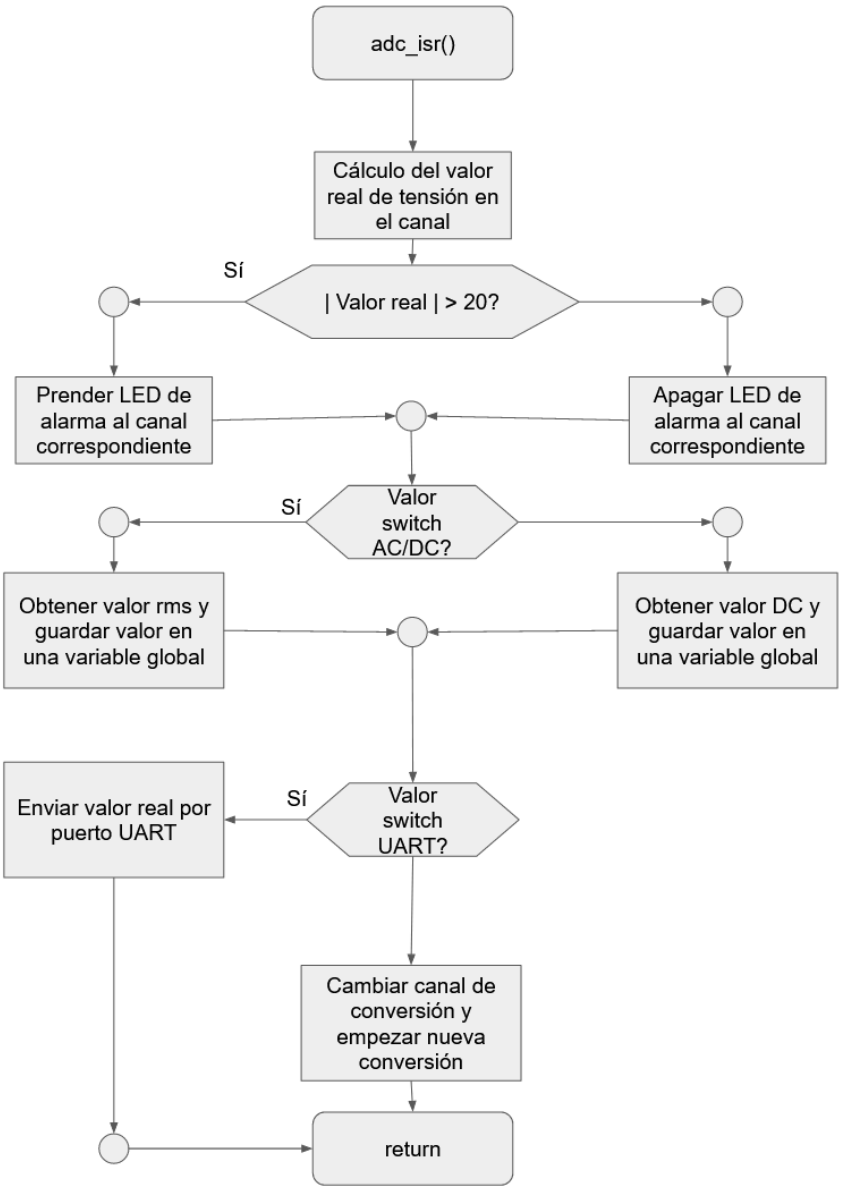
Esta ventana de tiempo es lo que limita la frecuencia mínima en la que se puede medir. Y la velocidad de muestreo es la que define la frecuencia máxima en la que se puede medir. En esta aplicación, el convertidor analógico digital está configurado para realizar una conversión a una frecuencia de 16 MHz / 128 = 125kHz. Sin embargo, como hay 4 canales, entonces cada canal tiene una velocidad de muestreo de 125 kHz / 4 = 31 kHz. Adicionalmente, según el teorema de muestreo de Nyquist, se ocupa una frecuencia de muestro el doble de rápida que la frecuencia de la señal para no perder información. Entonces la frecuencia máxima de

una señal AC que puede medir la aplicación es de 31 kHz / 2 = **15,5 kHz**.

Entonces la aplicación puede medir señales AC con frecuencias entre 20 Hz y 15,5 kHz.

Diagrama de bloques

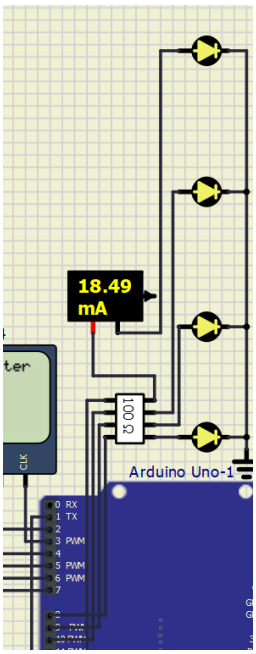




3.2. Componentes

LEDS de alarma

Según los valores escogidos de resistencias de acuerdo con lo mencionado en la nota teórica del diseño del circuito, la corriente pasando por los LEDS debería ser marginalmente menor a 20 mA.

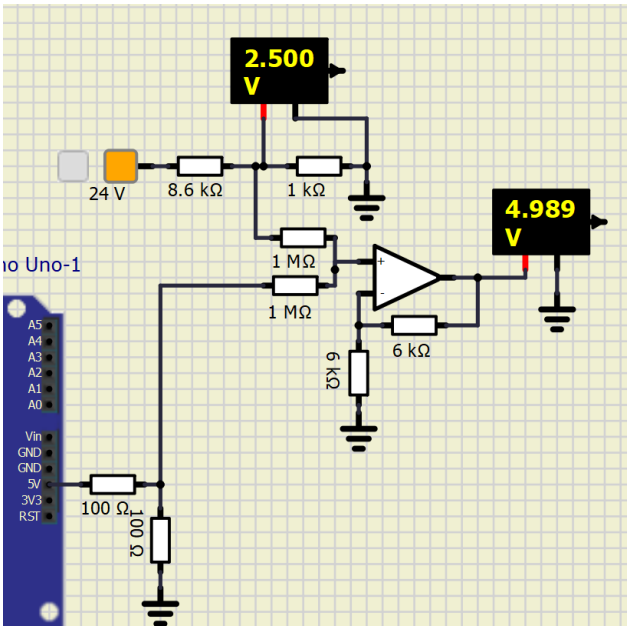


Corriente LEDS 3.1: Obtenido del simulador

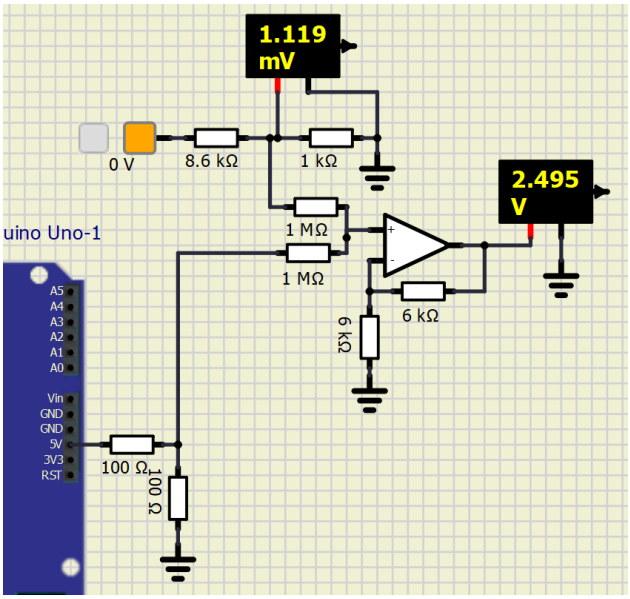
Como se observa de la imágen 3.1, la corriente es de 18,49 mA, un valor esperado.

Divisor y sumador de tensiones

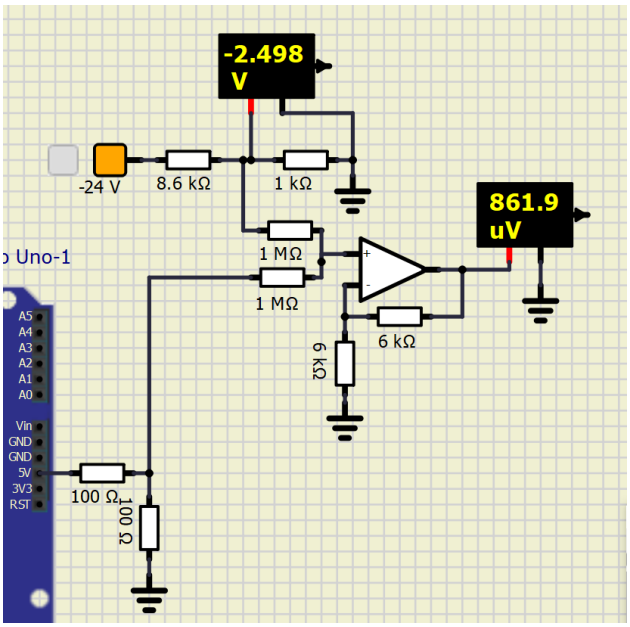
Según los valores escogidos de resistencias de acuerdo con lo mencionado en la nota teórica del diseño del circuito, la tensión a la salida de la etapa de división de tensión acoplada con la summatoria de tensiones debería estar entre 0V y 5V:



Límite superior 24V 3.2: Obtenido del simulador



Límite medio 0V 3.3: Obtenido del simulador

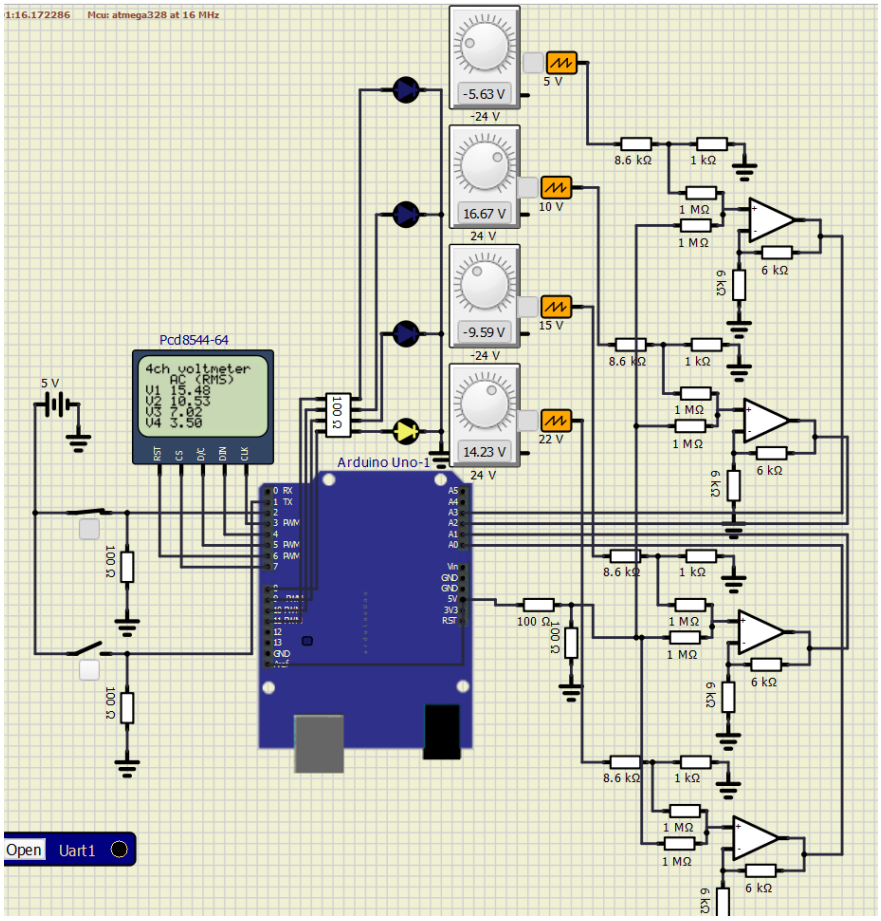


Límite inferior -24V 3.4: Obtenido del simulador

Como se observa de las imágenes, se obtiene el comportamiento esperado de lograr convertir las señales a un rango de 0V-5V con un grado leve de error. Específicamente, se observa en las imágenes que una tensión de 24V se convierte en ∼5V, 0V se convierte en ∼2.5 y -24V se convierte en ∼0V.

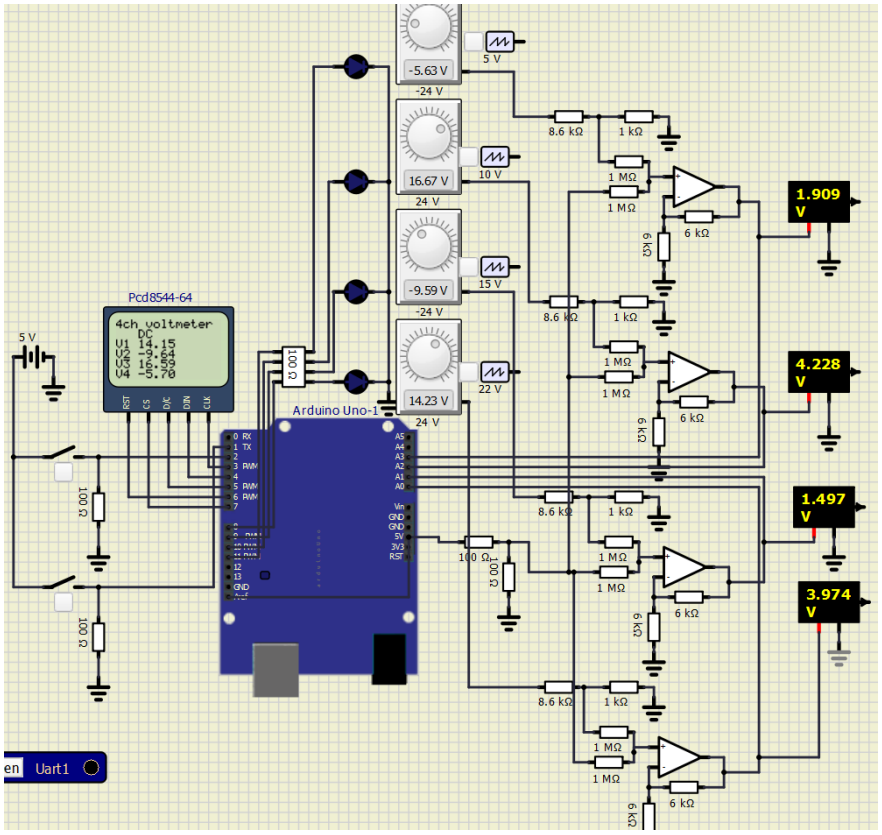
3.3. Funcionamiento

Mediciones en AC



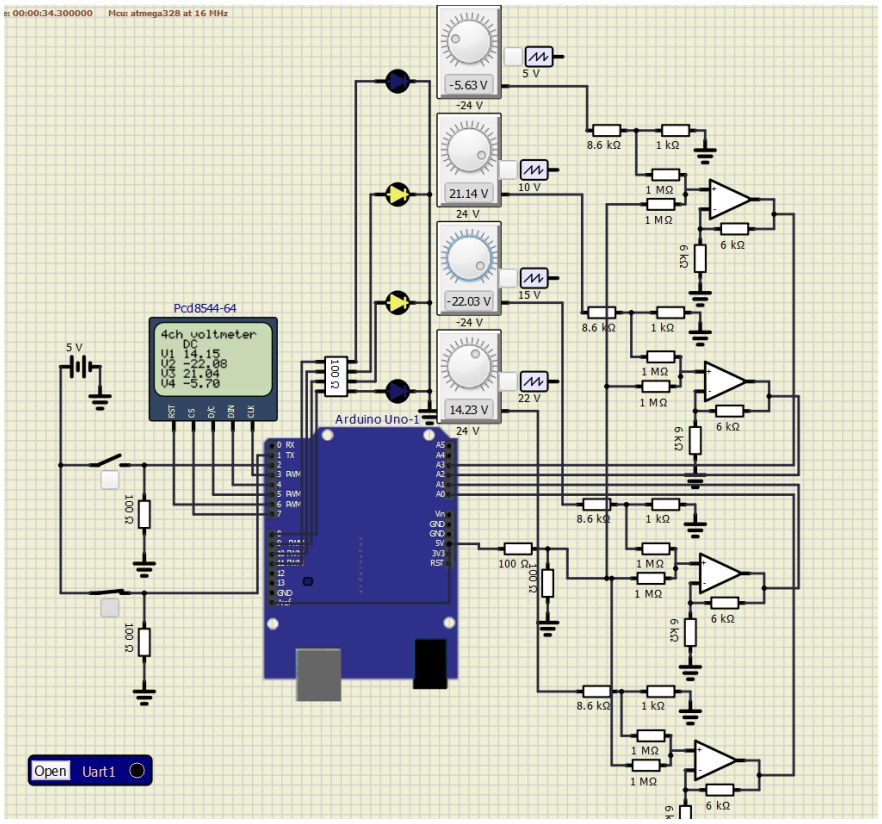
Funcionamiento en AC 3.5: Obtenido del simulador

Mediciones en DC



Funcionamiento en DC 3.6: Obtenido del simulador

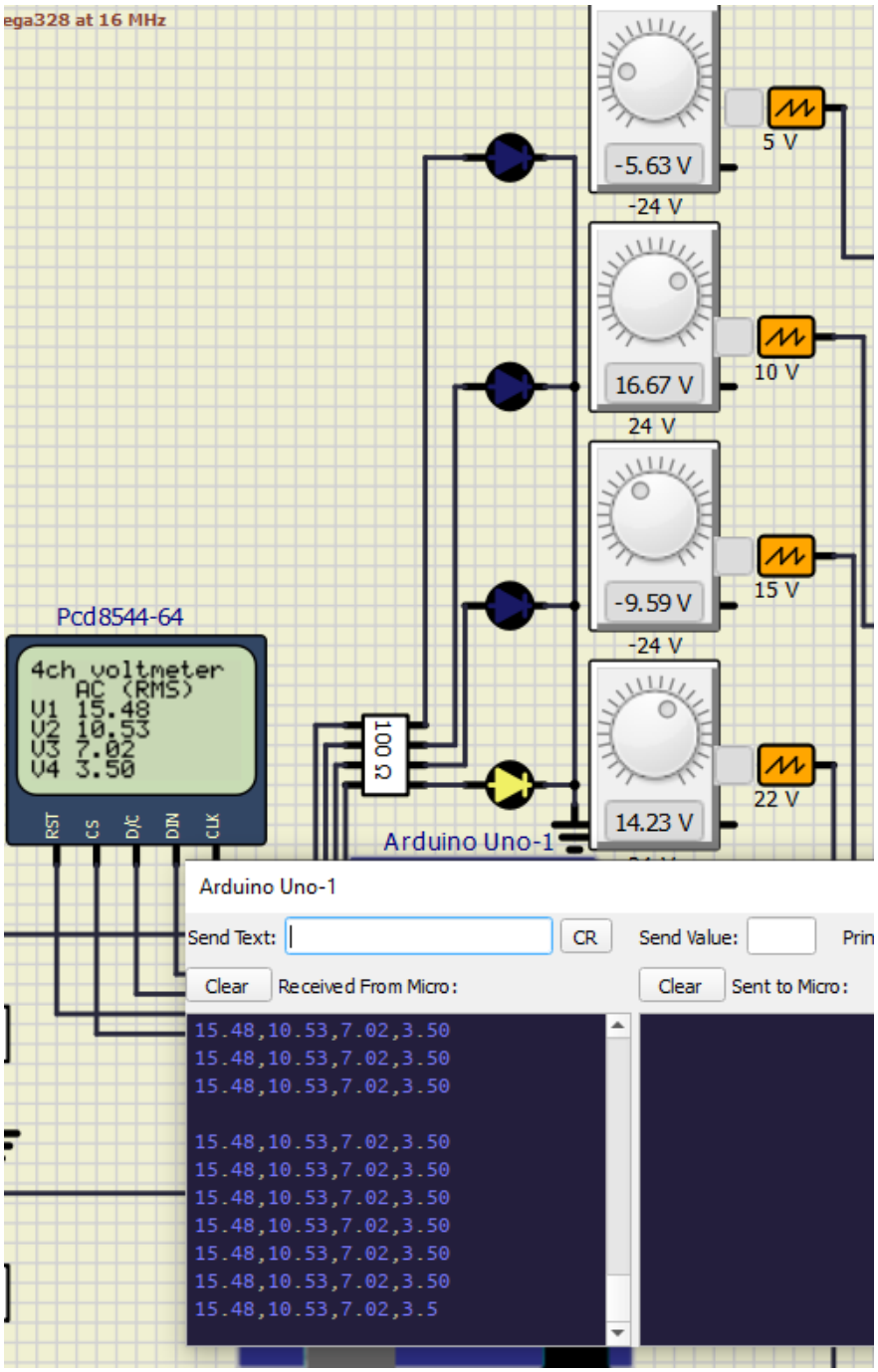
Alarma de tensiones mayores a 20V



Sobrecarga de tensión 3.7: Obtenido del simulador

Comunicación UART con la PC

El archivo de simulación y el script de python asume que existe una conexión virtual entre dos puertos seriales COM1 y COM2. El micro está conectado a un puerto de comunicación serial COM1 y el script de python al puerto COM2.



UART 3.8: Obtenido del simulador

Archivo CSV creado por python en la PC

	A	B	C	D
1	14.15	-22.08	21.04	-5.7
2	14.15	-22.08	21.04	-5.7
3	14.15	-22.08	21.04	-5.7
4	14.15	-22.08	21.04	-5.7
5	14.15	-22.08	21.04	-5.7
6	14.15	-22.08	21.04	-5.7
7	14.15	-22.08	21.04	-5.7
8	14.15	-22.08	21.04	-5.7
9	14.15	-22.08	21.04	-5.7
10	14.15	-22.08	21.04	-5.7
11	14.15	-22.08	21.04	-5.7
12	14.15	-22.08	21.04	-5.7
13	14.15	-22.08	21.04	-5.7
14	14.15	-22.08	21.04	-5.7
15	14.15	-22.08	21.04	-5.7
16	14.15	-22.08	21.04	-5.7

CSV 3.9: Obtenido de la PC

Como se observa de las imágenes, todas las partes (alarma de sobrecarga, comunicación SPI, comunicación UART, transformación de rango de tensiones y medición de valores reales de tensión) se comportan según lo esperado del diseño.

4. Conclusiones

Se concluyé que la solución propuesta para este tercer laboratorio cumple con los requerimientos del enunciado y se comporta según el diseño teórico realizado.

4.1. Recomendaciones

Un cambio que se puede realizar a la solución propuesta es cambiar la forma en la que se calcula los valores rms. En este caso, por medio de muestreo se obtiene el valor pico y se multiplica por 0,7071. Una forma alternativa es obtener el valor promedio de la señal y multiplicarlo por 1,1.

Bibliografía

[1] "Arduino uno data sheet." https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf, 2007. (Accessed on 21/04/2023).

[2] "Atmega328p data sheet." https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf, 2007. (Accessed on 21/04/2023).

[3] "Pcd8544 data sheet." https://www.alldatasheet.com/view.jsp?Searchword=Pcd8544%20datasheet&gclid=Cj0KCQjw3a2iBhCFARIsAD4jQB1Q01D8coKa4vAcT7sENEfJCbOcj4UV53bf04lnleD9aGQYH1CqdxEaAmPlEwcB, 2007. (Accessed on 21/04/2023).

[4] "Componentes electrónicos." https://www.digikey.com/en/products/, 2005. (Accessed on 21/04/2023).

[5] "Microcontroladores." https://en.wikipedia.org/wiki/Microcontroller, 2006. (Accessed on 21/04/2023).

5. GIT

https://github.com/JAR1224/Laboratorio_3_Jose_Ramos