INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRÁCTICA 14: VÚMETRO

ALUMNOS: MALAGON BAEZA ALAN ADRIAN
MARTINEZ CHAVEZ JORGE ALEXIS

GRUPO: 6CM3

U.A: SISTEMAS EN CHIP

PROFESOR: FERNANDO AGUILAR SÁNCHEZ

FECHA DE ENTREGA: 28 DE MAYO DE 2023

OBJETIVO

Al término de la sesión, los integrantes del equipo contaran con la habilidad de hacer uso del convertidor analógico digital del microcontrolador, implementándolo como un Vúmetro de dos canales.

INTRODUCCIÓN TEÓRICA

VÚMETRO

El vúmetro es un dispositivo capaz de medir el nivel de volumen en unas unidades especiales de las hablaré más adelante. Estos dispositivos pueden ser tanto digitales como analógicos, siendo éstos últimos los más habituales en la actualidad.[1]

Un vúmetro convencional básicamente está compuesto por una bobina móvil o galvanómetro con una cierta amortiguación. Ésta estará alimentada por un rectificador de onda completa que se conecta a una línea de audio mediante una resistencia en serie. De esa forma, no necesitará ninguna fuente de energía adicional, solo la energía de la propia señal del sonido.[1]

De esa forma, podrá mostrar las variaciones de tensión de la señal de audio, consiguiendo que el usuario pueda ver el nivel de volumen en cada momento. Lo harán mediante una aguja en un dial si son analógicos, o mediante LEDs si son digitales.[1]

Figura 1. Vúmetro analógico

La unidad de volumen VU (Volume Unit) se define como «El indicador de volumen marca 0 VU cuando se conecta una salida con una resistencia interna de 600 ohmios, para una señal sinusoidal de 1000 Hz y una amplitud de +4 dBu.«.[1]

Un término adoptado para mediciones de volumen relacionados con la percepción humana de intensidad de señal de audio.[1]

Al ser un dispositivo para medir volumen, el vúmetro tiene multitud de aplicaciones en dispositivos de sonido. Por ejemplo, lo podrás encontrar en algunas mesas de mezclas, ecualizadores, reproductores de audio, equipos de música, programas de audio, y un largo etc. Así que seguramente en más de una ocasión has tenido uno delante y ni siquiera sabías que se llamaba vúmetro.[1]

Tú podrías usarlo para obtener de una forma más visual el nivel de sonido para un ecualizador casero, para conectarlo a la salida de audio de algún dispositivo que hayas montado, etc. Algunos incluso crean composiciones de LEDs complejas como simples elementos decorativos, para que se iluminen al «ritmo» de la música.[1]

Figura 2. Vúmetro en consola para DJ

MATERIALES Y EQUIPO EMPLEADO

- ✓ CodeVision AVR
- ✓ AVR Studio 4
- ✓ Microcontrolador ATmega 8535
- ✓ 16 LEDS
- ✓ 16 Resistores de 330 Ω a 1⁄4 W
- ✓ 2 resistores de 100 kΩ a 1⁄4 W
- ✓ 2 Micrófonos

DESARROLLO EXPERIMENTAL

1. Realice una conversión de 8 bits sobre los canales 0 y 1 del ADC, establezca su voltaje de referencia para mostrar el resultado con leds en el Puerto B y D.

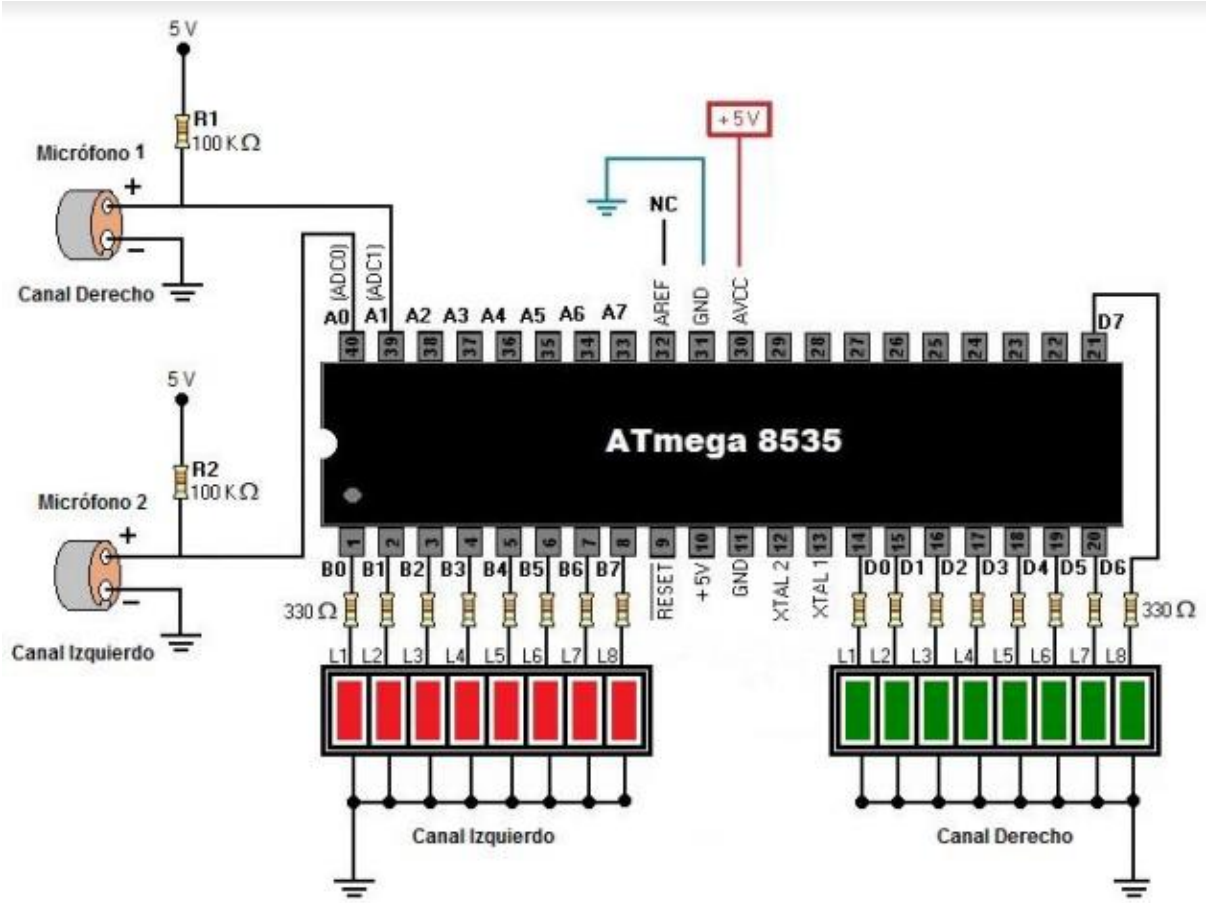


Figura 3. Circuito para el Vúmetro

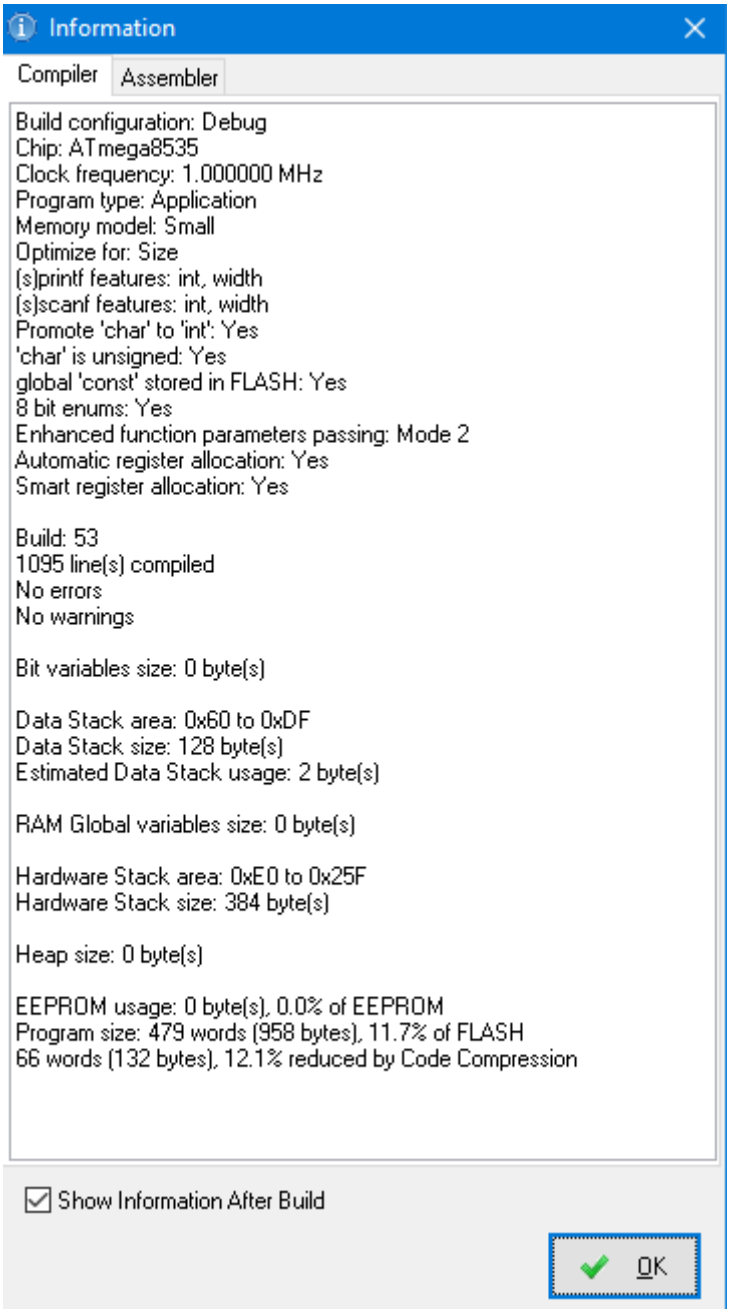


Figura 4. Compilación exitosa en CodeVision

CÓDIGO GENERADO POR CODEVISION

```
/*****************************************************
This program was created by the CodeWizardAVR V3.48b
Automatic Program Generator
© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    :
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

#include <mega8535.h>

#include <delay.h>

#define barra PINC.0
#define hold PINC.1
#define punto PINC.2

// Voltage Reference: AVCC pin
#define ADC_VREF_TYPE ((0<<REFS1) | (1<<REFS0) | (1<<ADLAR))

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=(1<<ADSC);
// Wait for the AD conversion to complete
while ((ADCSRA & (1<<ADIF))==0);
ADCSRA|=(1<<ADIF);
return ADCH;
}
```

```
// Declare your global variables here

const char array
[9]={0x00,0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF};
int i, j;

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In
Bit1=In Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) |
(0<<DDA2) | (0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) |
(0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out
Bit1=Out Bit0=Out
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) |
(1<<DDB2) | (1<<DDB1) | (1<<DDB0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) |
(0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In
Bit1=In Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) |
(0<<DDC2) | (0<<DDC1) | (0<<DDC0);
// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
PORTC=(1<<PORTC7) | (1<<PORTC6) | (1<<PORTC5) | (1<<PORTC4) |
(1<<PORTC3) | (1<<PORTC2) | (1<<PORTC1) | (1<<PORTC0);

// Port D initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out
Bit1=Out Bit0=Out
DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) |
(1<<DDD2) | (1<<DDD1) | (1<<DDD0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) |
(0<<PORTD3) | (0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization
```

```
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) |
(0<<CS02) | (0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) |
(0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) |
(0<<CS12) | (0<<CS11) | (0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) |
(0<<CS22) | (0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) |
(0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);
```

```
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// USART disabled
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN)
| (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) |
(0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);

// ADC initialization
// ADC Clock frequency: 500.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: ADC Stopped
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE)
| (0<<ADPS2) | (0<<ADPS1) | (1<<ADPS0);
SFIOR=(1<<ADHSM) | (0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) |
(0<<CPHA) | (0<<SPR1) | (0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

while (1)
      {
        if(!barra){ //Modo barra
            //Lectura datos
            i = read_adc(0);
            j = read_adc(1);
```

```
            PORTB=array[i/32];
            PORTD=array[j/32];
        }
        else if(!hold){ //Modo hold
            //Lectura datos
            i = read_adc(0);
            j = read_adc(1);

            PORTB=array[i/32];
            PORTD=array[j/32];
            delay_ms(150);  //Mantiene el dato
        }
        else if(!punto){    //Modo punto
            //Lectura datos
            i = read_adc(0);
            j = read_adc(1);

            //Pin A a Port B
            if(i>0 && i<32){
                PORTB.0 = 1;
                PORTB.1 = 0;
                PORTB.2 = 0;
                PORTB.3 = 0;
                PORTB.4 = 0;
                PORTB.5 = 0;
                PORTB.6 = 0;
                PORTB.7 = 0;
            }
            else if(i>32 && i<64){
                PORTB.0 = 0;
                PORTB.1 = 1;
                PORTB.2 = 0;
                PORTB.3 = 0;
                PORTB.4 = 0;
                PORTB.5 = 0;
                PORTB.6 = 0;
                PORTB.7 = 0;
            }
            else if(i>64 && i<96){
                PORTB.0 = 0;
                PORTB.1 = 0;
                PORTB.2 = 1;
                PORTB.3 = 0;
                PORTB.4 = 0;
                PORTB.5 = 0;
                PORTB.6 = 0;
                PORTB.7 = 0;
            }
            else if(i>96 && i<126){
```

```
                PORTB.0 = 0;
                PORTB.1 = 0;
                PORTB.2 = 0;
                PORTB.3 = 1;
                PORTB.4 = 0;
                PORTB.5 = 0;
                PORTB.6 = 0;
                PORTB.7 = 0;
            }
            else if(i>126 && i<160){
                PORTB.0 = 0;
                PORTB.1 = 0;
                PORTB.2 = 0;
                PORTB.3 = 0;
                PORTB.4 = 1;
                PORTB.5 = 0;
                PORTB.6 = 0;
                PORTB.7 = 0;
            }
            else if(i>160 && i<192){
                PORTB.0 = 0;
                PORTB.1 = 0;
                PORTB.2 = 0;
                PORTB.3 = 0;
                PORTB.4 = 0;
                PORTB.5 = 1;
                PORTB.6 = 0;
                PORTB.7 = 0;
            }
            else if(i>192 && i<224){
                PORTB.0 = 0;
                PORTB.1 = 0;
                PORTB.2 = 0;
                PORTB.3 = 0;
                PORTB.4 = 0;
                PORTB.5 = 0;
                PORTB.6 = 1;
                PORTB.7 = 0;
            }
            else if(i>224){
                PORTB.0 = 0;
                PORTB.1 = 0;
                PORTB.2 = 0;
                PORTB.3 = 0;
                PORTB.4 = 0;
                PORTB.5 = 0;
                PORTB.6 = 0;
                PORTB.7 = 1;
            }
```

```
            else{
                PORTB = 0x00;
            }

            //Pin A a Port D
            if(j>0 && j<32){
                PORTD.0 = 1;
                PORTD.1 = 0;
                PORTD.2 = 0;
                PORTD.3 = 0;
                PORTD.4 = 0;
                PORTD.5 = 0;
                PORTD.6 = 0;
                PORTD.7 = 0;
            }
            else if(j>32 && j<64){
                PORTD.0 = 0;
                PORTD.1 = 1;
                PORTD.2 = 0;
                PORTD.3 = 0;
                PORTD.4 = 0;
                PORTD.5 = 0;
                PORTD.6 = 0;
                PORTD.7 = 0;
            }
            else if(j>64 && j<96){
                PORTD.0 = 0;
                PORTD.1 = 0;
                PORTD.2 = 1;
                PORTD.3 = 0;
                PORTD.4 = 0;
                PORTD.5 = 0;
                PORTD.6 = 0;
                PORTD.7 = 0;
            }
            else if(j>96 && j<126){
                PORTD.0 = 0;
                PORTD.1 = 0;
                PORTD.2 = 0;
                PORTD.3 = 1;
                PORTD.4 = 0;
                PORTD.5 = 0;
                PORTD.6 = 0;
                PORTD.7 = 0;
            }
            else if(j>126 && j<160){
                PORTD.0 = 0;
                PORTD.1 = 0;
                PORTD.2 = 0;
```

```
                PORTD.3 = 0;
                PORTD.4 = 1;
                PORTD.5 = 0;
                PORTD.6 = 0;
                PORTD.7 = 0;
            }
            else if(j>160 && j<192){
                PORTD.0 = 0;
                PORTD.1 = 0;
                PORTD.2 = 0;
                PORTD.3 = 0;
                PORTD.4 = 0;
                PORTD.5 = 1;
                PORTD.6 = 0;
                PORTD.7 = 0;
            }
            else if(j>192 && j<224){
                PORTD.0 = 0;
                PORTD.1 = 0;
                PORTD.2 = 0;
                PORTD.3 = 0;
                PORTD.4 = 0;
                PORTD.5 = 0;
                PORTD.6 = 1;
                PORTD.7 = 0;
            }
            else if(j>224){
                PORTD.0 = 0;
                PORTD.1 = 0;
                PORTD.2 = 0;
                PORTD.3 = 0;
                PORTD.4 = 0;
                PORTD.5 = 0;
                PORTD.6 = 0;
                PORTD.7 = 1;
            }
            else{
                PORTD = 0x00;
            }
        }
        else{   //No se seleccionó ningun modo
            PORTB = 0x00;
            PORTD = 0x00;
        }
    }
}
```

CIRCUITO ELECTRICO EN PROTEUS

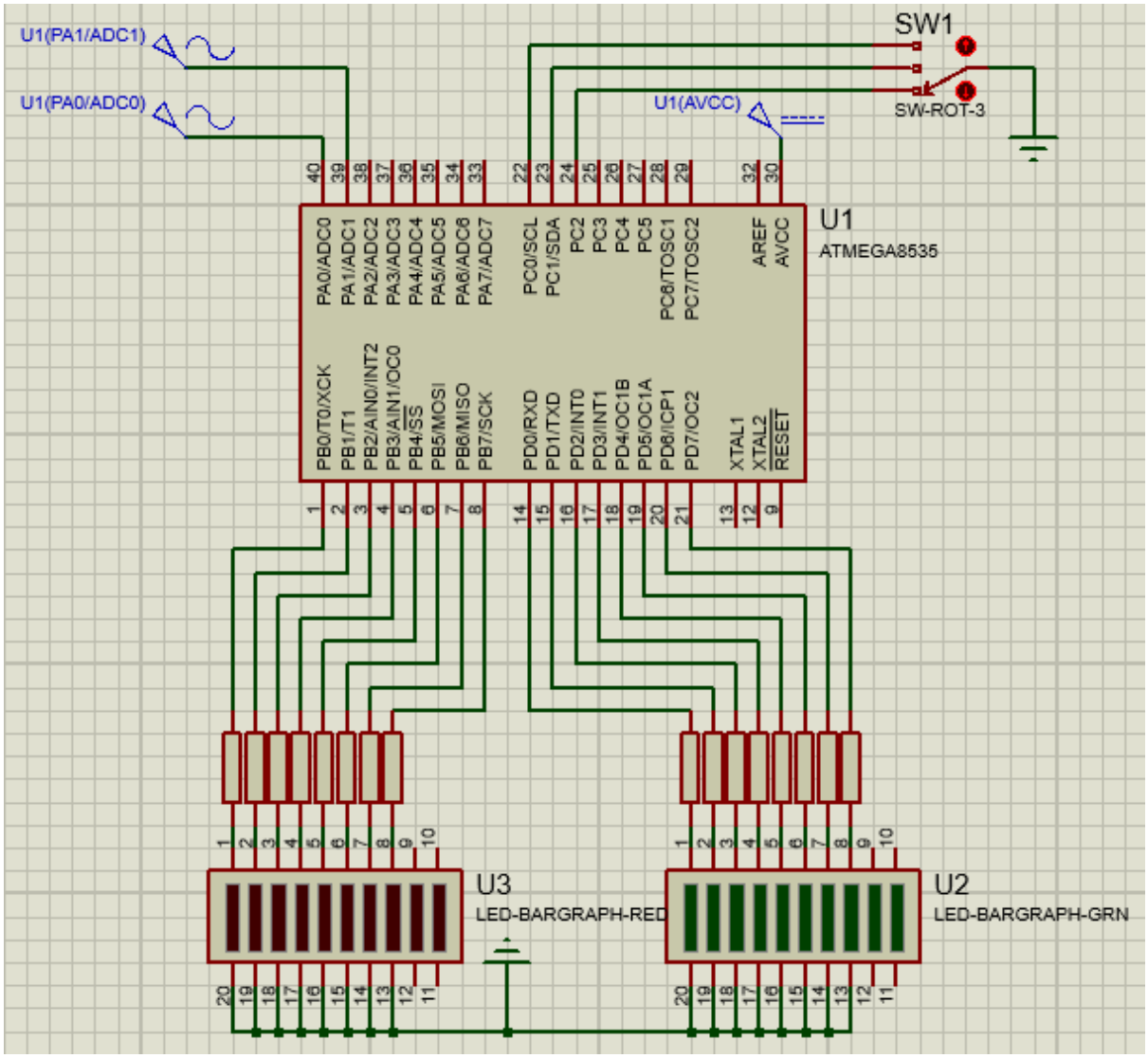
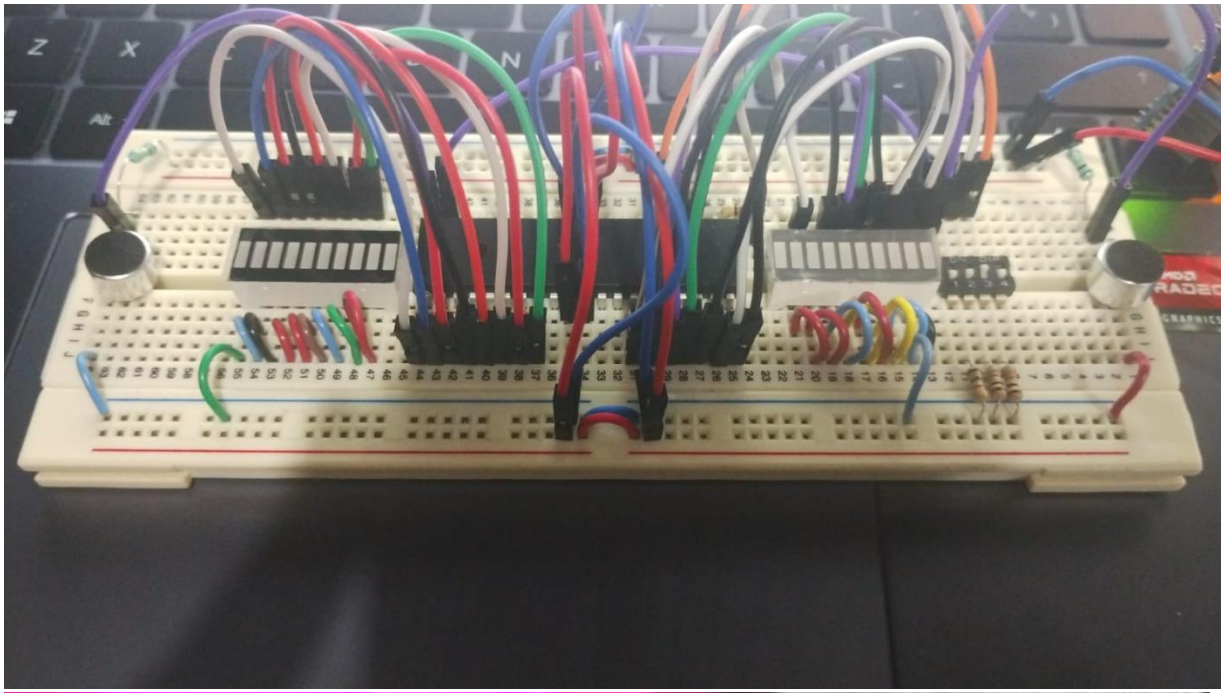
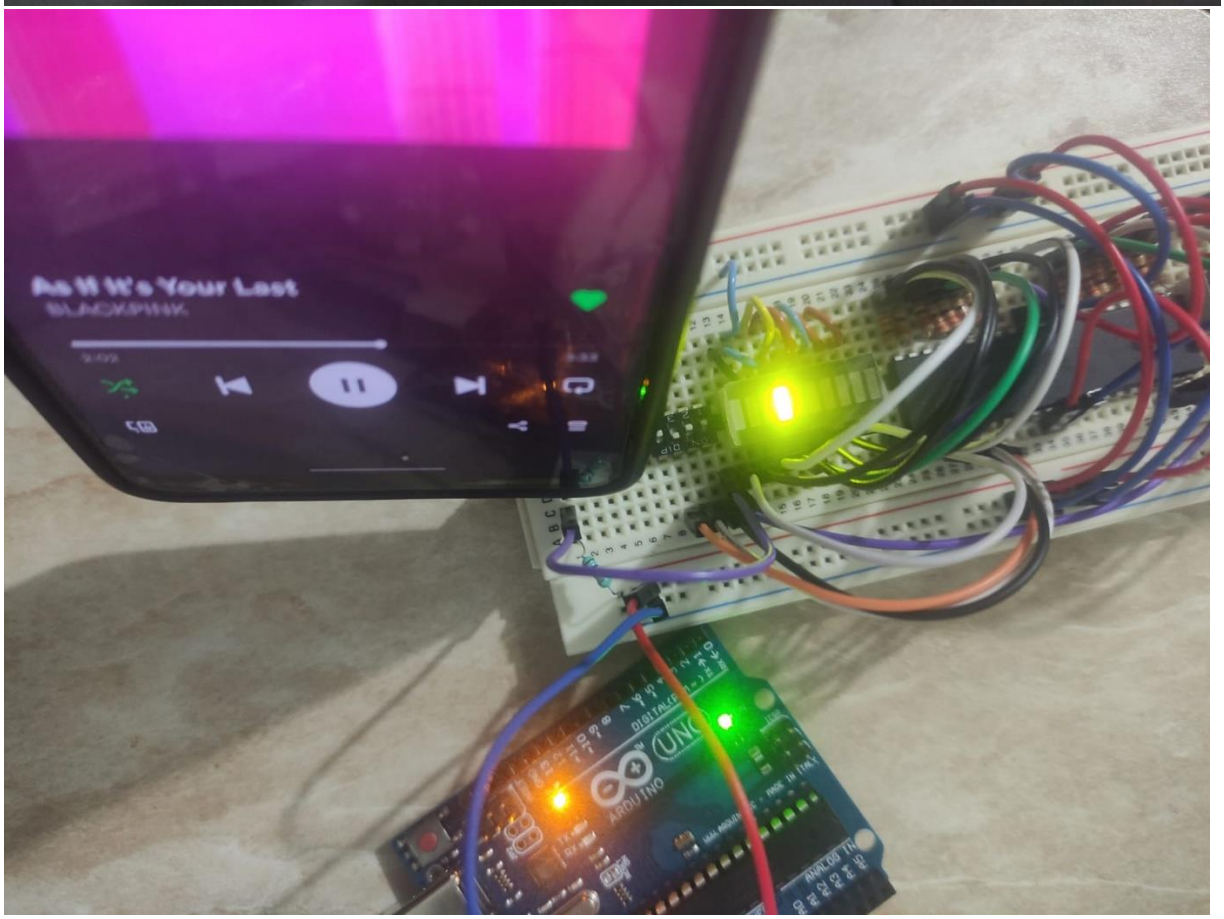
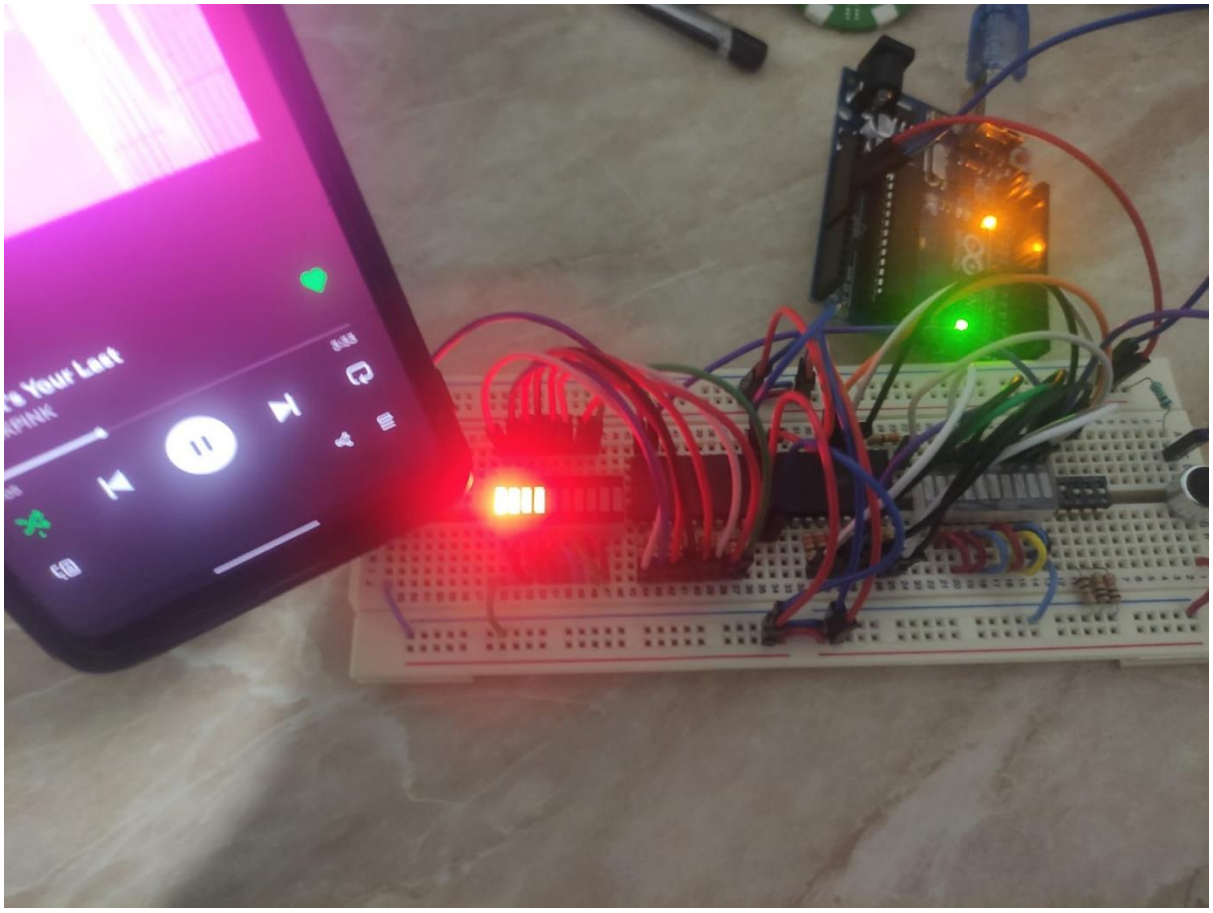


Figura 5. Circuito simulado en Proteus

CIRCUITO EN EL PROTO ARMADO




As If It's Your Last
BLACKPINK


Your Last

OBSERVACIONES Y CONCLUSIONES INDIVIDUALES

Malagón Baeza Alan Adrian

El vúmetro es un dispositivo que nos sirve para medir el nivel de volumen del sonido que estamos escuchando. Con el vúmetro podemos ver si un sonido es muy fuerte o si es muy débil. A los que les gusta el mundo de los DJ's tienen presente que el vúmetro es un dispositivo muy importante para medir el volumen de la pista que se está reproduciendo, para saber si el volumen es muy saturado o si se requiere aumentar el volumen para mantener un nivel igual en toda la reproducción de la música. Si bien, el vúmetro no es un dispositivo tan importante para todos los usuarios, si lo es para todas las personas que están interesadas en la música.

Martínez Chávez Jorge Alexis

Para implementar la práctica no fue tan difícil, puesto que sólo es leer el voltaje de entrada del ADC y convertirlo para representar esa información en una combinación binaria que se puede observar con las matrices de LED's. Para esta práctica se implementaron 3 formas de observar la información en el vúmetro: Barra, hold y punto. Para implementar el vúmetro de barra y punto no presenté muchos problemas, el que me llevó más tiempo de desarrollo fue el vúmetro de hold, puesto que es una especie de combinación del de barra y punto.

BIBLIOGRAFÍA

[1] Isaac, "Vúmetro: qué es y cómo se puede utilizar este dispositivo," *Hardware libre*, Nov. 25, 2020. https://www.hwlibre.com/vumetro/?utm_source=dlvr.it&utm_medium=facebook (accessed May 08, 2022).