INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores

3CM16

Practica 14

Vúmetro

Equipo 7

Integrantes:

- Bocanegra Heziquio Yestlanezi
- Dominguez Durán Alan Axel
- Hernandez Mendez Oliver Manuel
- Martinez Cruz José Antonio

Profesor: Aguilar Sanchez Fernando

Índice

Índice de imágenes

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

Convertidor Analógico-Digital

Un Vúmetro, abreviatura de "medidor de volumen", es un dispositivo utilizado para visualizar el nivel de una señal de audio en tiempo real. El Vúmetro es muy utilizado en aplicaciones de audio, como estudios de grabación, sistemas de sonido y equipos de música[1]. Con el avance de la tecnología, ahora es posible implementar un Vúmetro utilizando microcontroladores, lo que permite una mayor flexibilidad y capacidad de personalización.

En la implementación de un Vúmetro con microcontroladores, se utiliza el microcontrolador como el cerebro del sistema, encargado de adquirir la señal de audio, procesarla y controlar los indicadores visuales correspondientes, como los LED[2]. El microcontrolador realiza la conversión analógico-digital (ADC) para obtener una representación digital del nivel de la señal de audio y utiliza algoritmos y técnicas de procesamiento para interpretar y visualizar esta información en forma de barras o gráficos.

Material 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.

Circuito armado

Con los conocimientos obtenidos en las diferentes materias de electrónica, procedemos a realizar el montado del circuito en la protoboard como se muestra en la imagen 1.

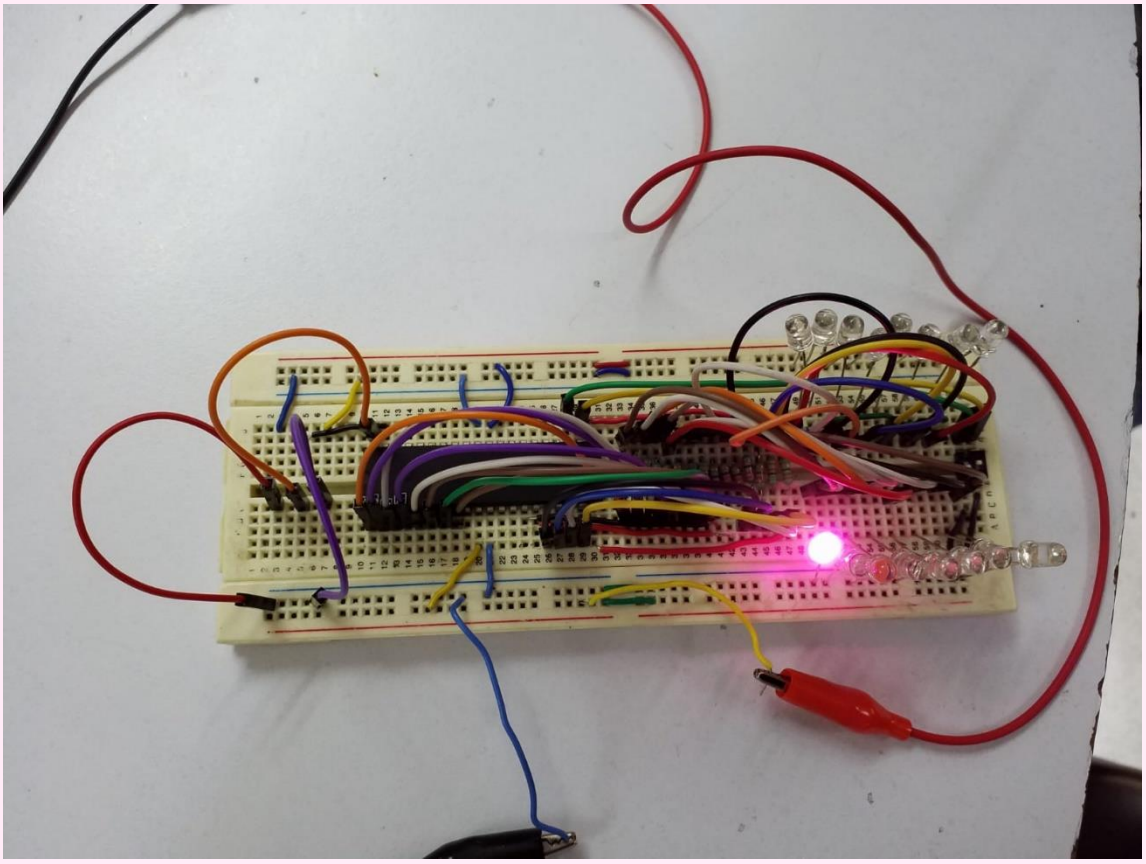

Imagen 1 vúmetro armado en protoboard

Estructura del programa

Código CodeVisionAVR

```
/*****************************************************
This program was created by the CodeWizardAVR V3.48b
Automatic Program Generator
ï¿½ Copyright 1998-2022 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 07/11/2022
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
*****************************************************/

// I/O Registers definitions
#include <mega8535.h>

#include <delay.h>

// 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
unsigned char adc0;
unsigned char adc1;
unsigned char salidasBarra[] = {0,1,3,7,15,31,63,127,255};
unsigned char salidasPunto[] = {0,1,2,4,8,16,32,64,128};
int i;
int hold0=0, hold1=0;
int retraso0=0, retraso1=0;
unsigned char getMSB(unsigned char numero);

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=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<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)
      {

            adc0 = read_adc(0);
            adc1 = read_adc(1);

            if(PINC.2==0 && PINC.1==0 && PINC.0==0){
                  PORTB = 0x00;
                  PORTD = 0x00;
            }else if(PINC.2==0 && PINC.1==0 && PINC.0==1){

                  PORTB = salidasBarra[getMSB(adc0)];
                  PORTD = salidasBarra[getMSB(adc1)];
            }else if(PINC.2==0 && PINC.1==1 && PINC.0==1){

                  PORTB = salidasPunto[getMSB(adc0)];
                  PORTD = salidasPunto[getMSB(adc1)];
            }else if(PINC.2==1 && PINC.1==1 && PINC.0==1){

                  if(getMSB(adc0) >= hold0){
                        hold0 = getMSB(adc0);
                        retraso0 = 0;
                  }
                  if(retraso0==400){
                        hold0--;
                        retraso0=0;
                  }

                  if(getMSB(adc1) >= hold1){
                        hold1 = getMSB(adc1);
                        retraso1 = 0;
                  }
                  if(retraso1==400){
                        hold1--;
                        retraso1=0;
                  }
```

```
                PORTB = salidasBarra[getMSB(adc0)] | salidasPunto[hold0];
                PORTD = salidasBarra[getMSB(adc1)] | salidasPunto[hold1];
                retraso0++;
                retraso1++;
            }

      }
}

unsigned char getMSB(unsigned char numero){
      unsigned char msb = 0;
      if(numero==0)
            return 0;
      for(i=0; i<=8; i++){
            numero = numero>>1;
            msb++;
            if(numero ==0){
                  break;
            }
      }
      return msb;
}
```

Simulación – Proteus

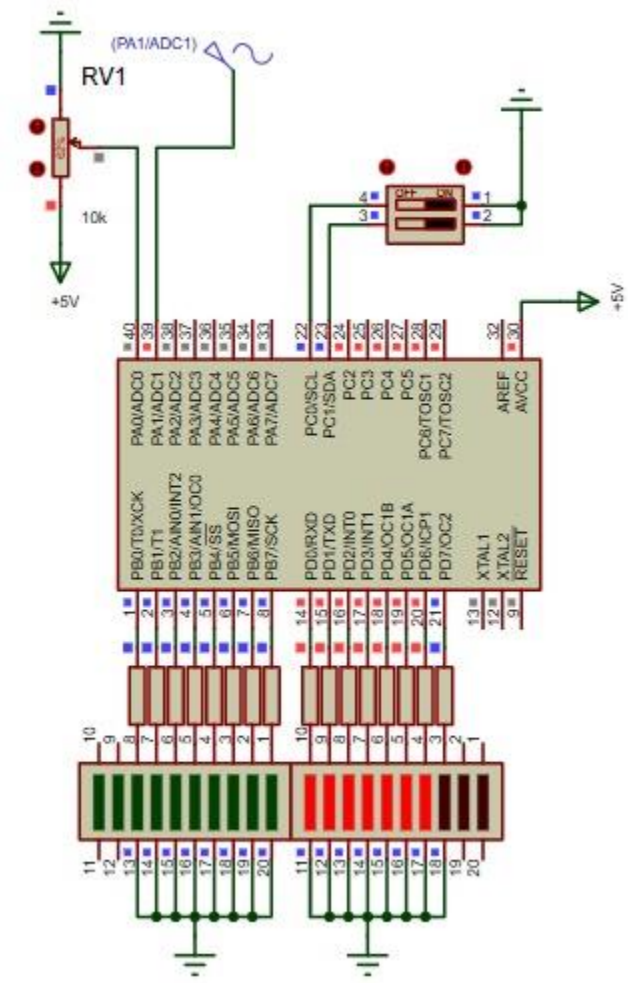

Imagen 2 Circuito simulado en proteus

Conclusiones

Bocanegra Heziquio Yestlanezi

Con las habilidades adquiridas anteriormente, somos capaces de desarrollado la habilidad de utilizar el convertidor analógico-digital del microcontrolador ATmega8535 para implementar un Vúmetro de dos canales. Mediante el uso de CodeVision AVR, AVR Studio 4, 16 LEDs, 16 resistores de 330 Ω, 2 resistores de 100 KΩ y 2 micrófonos, por lo cual con el conocimiento adquirido pudimos diseñar y construir un Vúmetro.

Utilizando CodeVision AVR y AVR Studio 4, programamos el microcontrolador de manera eficiente para convertir las señales analógicas provenientes de los micrófonos en datos digitales.

Gracias a los conocimientos adquiridos en el desarrollo de la práctica, podremos utilizar estas habilidades en futuros proyectos relacionados con el procesamiento de señales y la manipulación de datos analógicos y digitales.

Domínguez Durán Alan Axel

Utilizando nuevamente el módulo convertidor analógico digital, logramos captar una señal de audio con un micrófono en una de las entradas del microcontrolador y proyectamos ese audio con una serie de leds para visualizar la intensidad de esa señal. Esta es una de las muchas funciones que se le pueden dar al ADC, sin duda una práctica educativa e interesante.

Hernández Méndez Oliver Manuel

Esta práctica me brindó la oportunidad de aplicar mis conocimientos de microcontroladores para implementar un Vúmetro, una aplicación práctica y emocionante. Aprendí sobre la selección y configuración de componentes, la programación de microcontroladores y la realización de pruebas para garantizar un funcionamiento adecuado. Esta experiencia me permitió fortalecer mis habilidades en el ámbito de los microcontroladores y experimentar con la electrónica en un contexto práctico y divertido.

Martínez Cruz José Antonio

El objetivo de la práctica fue alcanzado a pesar de que existieron inconvenientes con los micrófonos. Asi mismo, logramos identificar el alcance que puede ofrecer el uso de microprocesadores como este, tales como la aplicación del vumetro, a pesar de que hoy en día no es algo común sigue mostrando de manera clara como es que se recibe el sonido por el micrófono y como puede ser modifica su interpretación para que se adecué a lo que el usuario requiera. Cabe aclarar que esta configuración no esta hecha para recibir sonidos del ambiente, ya que se requiere una configuración adicional para alcanzar dicho objetivo.

Referencias

[1] Kamath, A. y Bhat, S. (2017). Implementación de VU Meter usando Microcontrolador. Revista Internacional de Investigación Avanzada en Ingeniería y Tecnología Informática (IJARCET), 6(8), 2259-2263.

[2] Horowitz, P. y Hill, W. (2015). El arte de la electrónica. Prensa de la Universidad de Cambridge.