Instituto Politécnico Nacional

Escuela Superior de Cómputo

Departamento de Ciencias e Ingeniería de la Computación

Sistemas en Chip

Práctica 14

"Vúmetro"

Profesor: Fernando Aguilar Sánchez

Grupo: 6CM1

Equipo 3

Alumnos:

Ocampo Téllez Rodolfo

Patlani Mauricio Adriana

Ruvalcaba Flores Martha Catalina

Sandoval Hernández Eduardo

Fecha de entrega: 15/01/2023

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

El microcontrolador ATMEGA8535 fabricante ATMEL. En la figura 1 se muestra el microcontrolador ATMEGA8535 la cual maneja datos de 8 bits es decir su bus de datos de 8 bits. Aunque este contiene tres registros los cuales son el *x*, *y* y *z*, los cuales manejan datos de 16 bits.

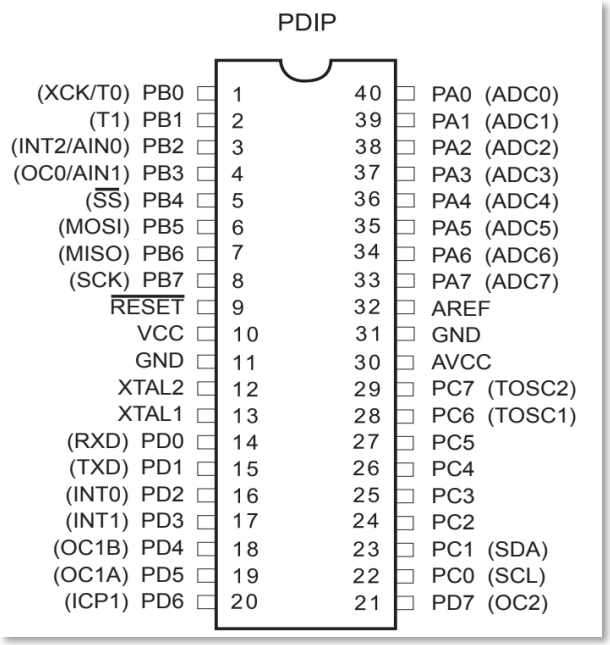

Figura 1. Configuración de pines ATmega8535

La comunicación interna del microcontrolador se categoriza en 4 puertos, en la figura 1 se puede analizar la etiquetación de los puertos PA0 al PA7, PB0 al PB7, PC0 al PC7 y PD0 al PD7. Cabe recordar que maneja datos de 8 bits.

Posee un oscilador interno de 1MHz, sin embargo, como es un oscilador RC, es susceptible a variar la frecuencia con respecto a las variaciones de temperatura. Por otro lado, puede conectarse un oscilador de 0 a 8 MHz o de 0 a 16 MHz.

Tiene 8 canales que sirven a un ADC de 10 bits, los 8 canales son de una sola terminal, 7 canales son diferenciales para únicamente paquetes TQFP y 2 de esos canales diferenciales permiten ganancia programable de 1x. 10x, o 200x.

Conversión Analógica-Digital

La conversión analógica-digital consiste en la transcripción de señales analógicas en señal digital, con el propósito de facilitar su procesamiento (codificación, comprensión, etcétera) y hacer la señal resultante (digital) más inmune al ruido y otras interferencias a las que son más sensibles las señales analógicas.

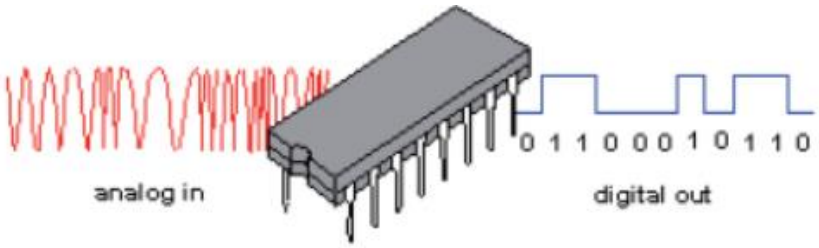

Figura 2. Conversión de información analógica a una digital.

Para realizar esa tarea, se utiliza un ADC (Analog-to-Digital Converter – Conversor Analógico-Digital) que tiene que efectuar los siguientes procesos:

- Muestreo de la señal analógica, que es la medición periódica de la amplitud de una señal para poder evaluar su nivel. Sin embargo, este proceso no se contempla desde el punto de vista matemático, puesto que es un recurso técnico sin modelo matemático y con limitaciones prácticas. Durante el muestreo, la señal sigue siendo analógica ya que ésta puede tomar cualquier valor.
- Cuantización o cuantificación, en este proceso lo que se mide es el voltaje de cada muestra, asignándoles un margen de valor de una señal analizada a un único nivel de salida. La desventaja es que siempre el resultado lleva una distorsión llamada "ruido de cuantificación".
- Codificación del resultado, codificar es la acción por la cual los valores que se obtuvieron en la cuantificación se traducen en un código binario o en otro tipo de códigos similares. La longitud de la palabra binaria obtenida depende de la resolución (en bits) que se decida utilizar.

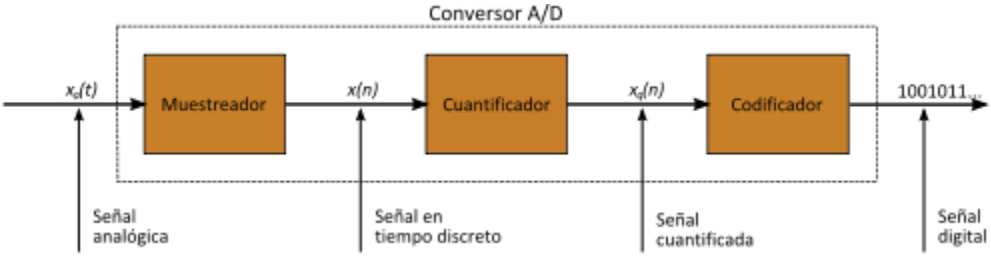

Figura 3. Diagrama del proceso de conversión analógica-digital.

Vúmetro

El vúmetro fue desarrollado originalmente en 1939 por Bell Labs para la medición y la normalización de los niveles en las líneas telefónicas. Actualmente suelen incluirse en equipos de audio para mostrar un nivel de señal en unidades de volumen.

La idea para desarrollar el vúmetro surgió a partir de la necesidad de proporcionar el nivel de audio suficiente sin llegar a percibir una distorsión en la señal ofrecida en la audiencia "Unknown Soldier on Armistice Day " celebrada en Arlington, New York y San Francisco en el año 1921.

En las pruebas de sonido previas a la audiencia se dieron cuenta de que el mismo volumen de salida provocaba una distorsión a través de un receptor telefónico y no en unos altavoces. Este problema dió lugar al desarrollo del Vúmetro, en ingles SVI (Stándar Volumen Indicator) ya que se tuvo la idea de crear un dispositivo que ofreciera información de forma visual sobre niveles de audio que podían entregarse sin llegar a ciertos límites los cuales provocaban esta distorsión y así poder controlar dicho nivel a través de potenciómetros o faders.

En la actualidad, estos dispositivos están presentes en multitud de aparatos de música, entre ellos las mesas de mezclas, implementados por software en algunos dispositivos de sonido como reproductores de MP3, ecualizadores, etc. Muestran las variaciones de tensión de la señal de audio, consiguiendo que el usuario pueda ver el nivel de volumen en cada momento.

Su medición es lenta a propósito para reflejar de una manera clara al usuario los máximos y mínimos en la señal entregada. Como norma general esa señal no debe superar la franja marcada con 0VU, ya que esto podía ocasionar pérdidas en la calidad de sonido ofreciendo una respuesta en frecuencia más pobre y una saturación en la grabación, ocasionando problemas en un futuro tratamiento de la señal en equipos de audio digital. Por otro lado, si la señal es demasiado baja la relación entre el ruido y la señal es más notable en la grabación lo cual también dificulta su posterior tratamiento si lo hubiese.

El vúmetro a base de LEDs es muy sencillo de construir, ya que apenas requiere de unos pocos componentes electrónicos, y en internet se pueden encontrar muchos esquemas diferentes con distintos circuitos integrados o transistores.

Figura 4. Vúmetros Digitales.

Materiales y Equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- 16 LEDS
- 16 Resistores de 330 Ω a ¼ W
- 2 Resistores de 100 KΩ a ¼ 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.

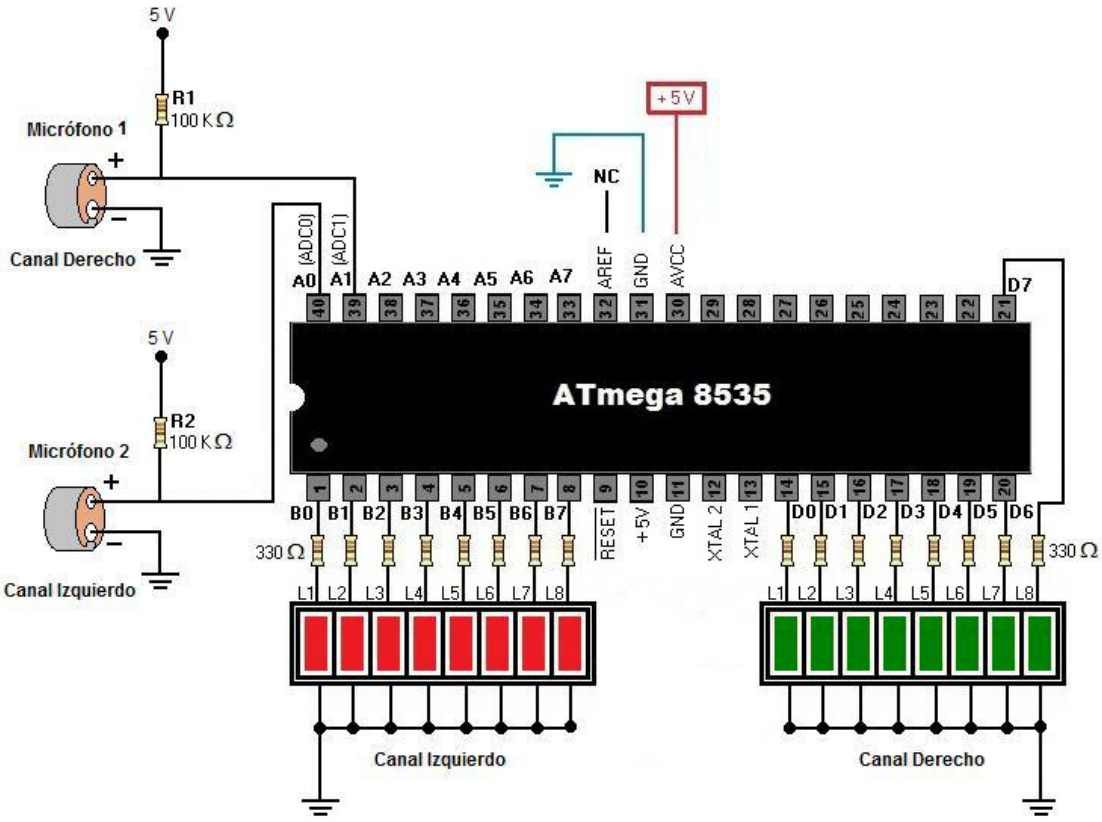

Figura 5. Circuito para el Vúmetro.

Estructura del programa

Código de la configuración de los periféricos utilizados y código del programa principal en C, proporcionados por el IDE de CodeVision.

```
#include <mega8535.h>

#include <delay.h>
// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (1<<ADLAR))

unsigned char hold1;
unsigned char hold2;
unsigned char max1 = 0x00;
unsigned char max2 = 0x00;
int count = 0;
unsigned char valorADC1;
unsigned char valorADC2;

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;
}

void main(void)
{
// Input/Output Ports initialization
// Port A initialization
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1)
| (0<<DDA0);
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2)
| (0<<PORTA1) | (0<<PORTA0);
// Port B initialization
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<DDB1)
| (1<<DDB0);
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2)
| (0<<PORTB1) | (0<<PORTB0);
// Port C initialization
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1)
| (0<<DDC0);
PORTC=(1<<PORTC7) | (1<<PORTC6) | (1<<PORTC5) | (1<<PORTC4) | (1<<PORTC3) | (1<<PORTC2)
| (1<<PORTC1) | (1<<PORTC0);
// Port D initialization
DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) | (1<<DDD2) | (1<<DDD1)
| (1<<DDD0);
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2)
| (0<<PORTD1) | (0<<PORTD0);
…
…
…
while (1)
      {
      valorADC1 = read_adc(0)/32;
      valorADC2 = read_adc(1)/32;
      if(PINC.0 == 0 && PINC.1 == 0){
            hold1 = 0;
            hold2 = 0;
            switch(valorADC1){
```

```
                case 0:     PORTB = 0x00;
                      break;
                case 1:     PORTB = 0x01;
                      break;
                case 2:     PORTB = 0x03;
                      break;
                case 3:     PORTB = 0x07;
                      break;
                case 4:     PORTB = 0x0f;
                      break;
                case 5:     PORTB = 0x1f;
                      break;
                case 6:     PORTB = 0x3f;
                      break;
                case 7:     PORTB = 0x7f;
                      break;
                case 8:     PORTB = 0xff;
                      break;
                default:    PORTB = 0x00;
                      break;
          }
          switch(valorADC2){
                case 0:     PORTD = 0x00;
                      break;
                case 1:     PORTD = 0x01;
                      break;
                case 2:     PORTD = 0x03;
                      break;
                case 3:     PORTD = 0x07;
                      break;
                case 4:     PORTD = 0x0f;
                      break;
                case 5:     PORTD = 0x1f;
                      break;
                case 6:     PORTD = 0x3f;
                      break;
                case 7:     PORTD = 0x7f;
                      break;
                case 8:     PORTD = 0xff;
                      break;
                default:    PORTD = 0x00;
                      break;
          }
     }
     else if(PINC.0 == 1 && PINC.1 == 0){
          switch(valorADC1){
```

```
                case 0:     hold1 = 0x00;
                      break;
                case 1:     hold1 = 0x01&0b00000001;
                      break;
                case 2:     hold1 = 0x03&0b00000010;
                      break;
                case 3:     hold1 = 0x07&0b00000100;
                      break;
                case 4:     hold1 = 0x0f&0b00001000;
                      break;
                case 5:     hold1 = 0x1f&0b00010000;
                      break;
                case 6:     hold1 = 0x3f&0b00100000;
                      break;
                case 7:     hold1 = 0x7f&0b01000000;
                      break;
                case 8:     hold1 = 0xff&0b10000000;
                      break;
                default:    hold1 = 0x00;
                      break;
          }
          switch(valorADC2){
                case 0:     hold2 = 0x00;
                      break;
                case 1:     hold2 = 0x01&0b00000001;
                      break;
                case 2:     hold2 = 0x03&0b00000010;
                      break;
                case 3:     hold2 = 0x07&0b00000100;
                      break;
                case 4:     hold2 = 0x0f&0b00001000;
                      break;
                case 5:     hold2 = 0x1f&0b00010000;
                      break;
                case 6:     hold2 = 0x3f&0b00100000;
                      break;
                case 7:     hold2 = 0x7f&0b01000000;
                      break;
                case 8:     hold2 = 0xff&0b10000000;
                      break;
                default:    hold2 = 0x00;
                      break;
          }
          PORTB = hold1;
          PORTD = hold2;
     }
```

```
    else if(PINC.0 == 0 && PINC.1 == 1){
        if(count == 10){
            count = 0;
            max1 = 0x00;
            max2 = 0x00;
        }else{
            if(max1<hold1)
                max1 = hold1;
            if(max2<hold2)
                max2 = hold2;
            count++;
        }
        PORTB = max1;
        PORTD = max2;
        delay_ms(50);
        switch(valorADC1){
            case 0:     PORTB = 0x00;     hold1 = 0x00;
                break;
            case 1:     PORTB = 0x01;     hold1 = 0x01&0b00000001;
                break;
            case 2:     PORTB = 0x03;     hold1 = 0x03&0b00000010;
                break;
            case 3:     PORTB = 0x07;     hold1 = 0x07&0b00000100;
                break;
            case 4:     PORTB = 0x0f;     hold1 = 0x0f&0b00001000;
                break;
            case 5:     PORTB = 0x1f;     hold1 = 0x1f&0b00010000;
                break;
            case 6:     PORTB = 0x3f;     hold1 = 0x3f&0b00100000;
                break;
            case 7:     PORTB = 0x7f;     hold1 = 0x7f&0b01000000;
                break;
            case 8:     PORTB = 0xff;     hold1 = 0xff&0b10000000;
                break;
            default:    PORTB = 0x00;     hold1 = 0x00;
                break;
        }
        switch(valorADC2){
            case 0:     PORTD = 0x00;     hold2 = 0x00;
                break;
            case 1:     PORTD = 0x01;     hold2 = 0x01&0b00000001;
                break;
            case 2:     PORTD = 0x03;     hold2 = 0x03&0b00000010;
                break;
            case 3:     PORTD = 0x07;     hold2 = 0x07&0b00000100;
                break;
```

```
                case 4:     PORTD = 0x0f;     hold2 = 0x0f&0b00001000;
                      break;
                case 5:     PORTD = 0x1f;     hold2 = 0x1f&0b00010000;
                      break;
                case 6:     PORTD = 0x3f;     hold2 = 0x3f&0b00100000;
                      break;
                case 7:     PORTD = 0x7f;     hold2 = 0x7f&0b01000000;
                      break;
                case 8:     PORTD = 0xff;     hold2 = 0xff&0b10000000;
                      break;
                default:    PORTD = 0x00;     hold2 = 0x00;
                      break;
            }
      }
      delay_ms(50);
      }
}
```

Simulaciones

La simulación se realizó en el software Proteus Design Suite, previamente cargado el programa en formato ".hex" obtenido del software CodeVision AVR, el cual es un software para el microcontrolador Microchip AVR y sus contrapartes XMEGA.

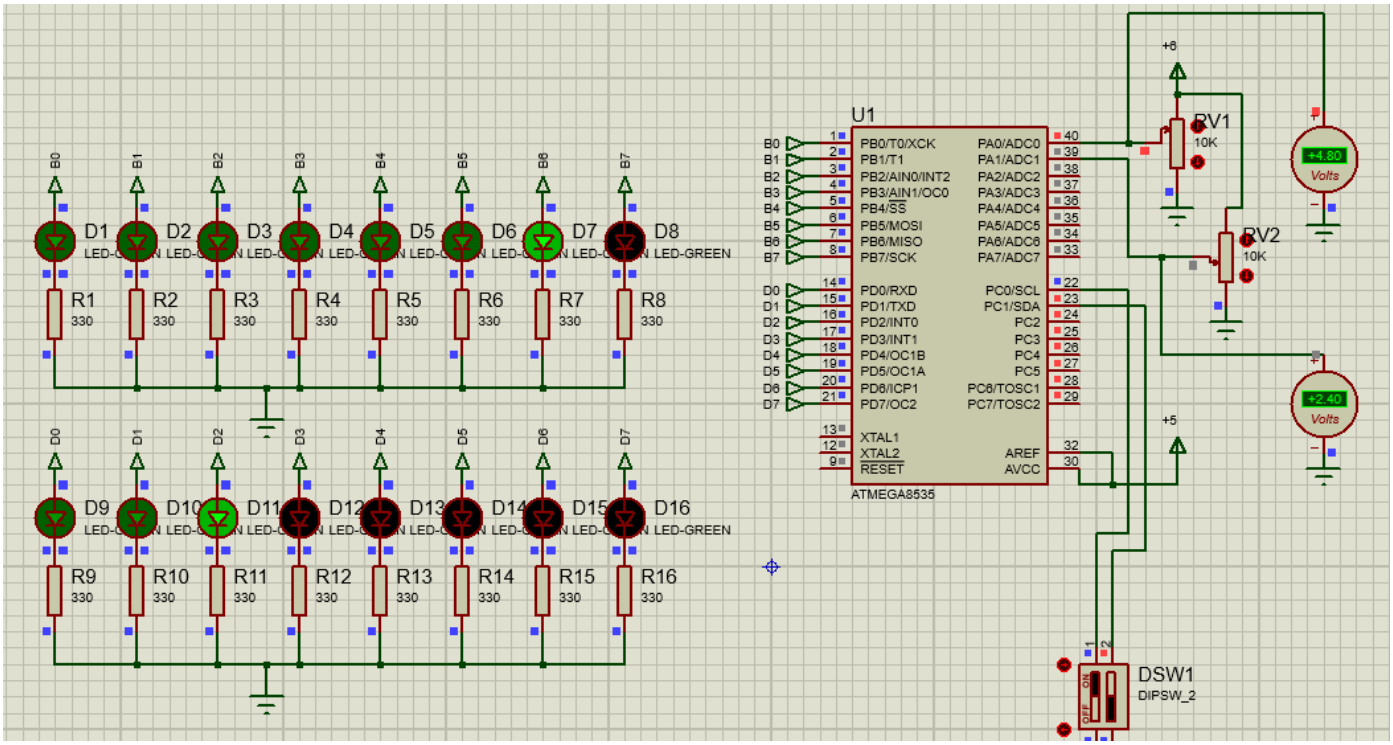

Figura 6. Simulación de la práctica 14.

Fotografía del circuito armado

A continuación, se muestra la figura 7 la evidencia sobre la realización y prueba de la practica 14.

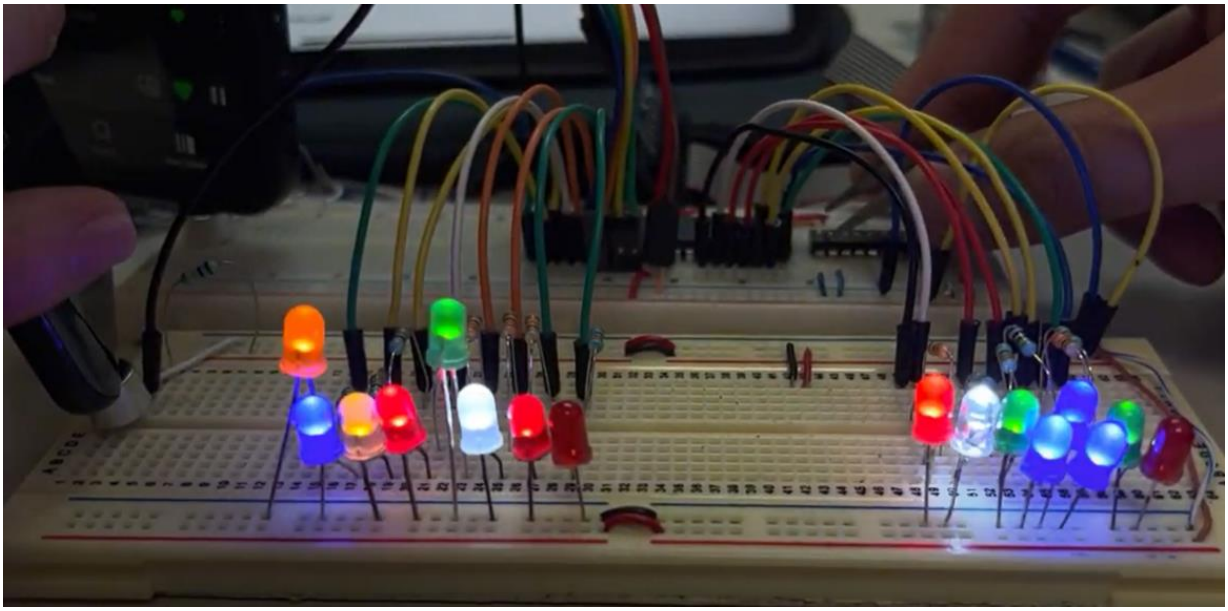

Figura 7. Circuito armado y funcionamiento de la práctica 14.

Observaciones y conclusiones Individuales

Ocampo Téllez Rodolfo

En esta práctica se realizó una aplicación concreta en la que se emplea el ADC del Atmega 8535, pues se necesitó convertir los valores analógicos del sonido captado por los micrófonos valores digitales para representar la intensidad a la que se estaba reproduciendo dicho sonido. A diferencia de la práctica pasada donde únicamente se mostraba un valor binario, ahora se asignaron los valores de la conversión mediante una serie de switch-case para que los LEDs se mostraran según el modo seleccionado manualmente de entre los 3 diferentes: que una vez alcanzara el valor digital permaneciera encendido (barra), que únicamente el último LED que indicaba el máximo estuviera encendido (punto) o una combinación de ambos (hold). Debido a la velocidad en que la música variaba de un tono a otro, que daba ese efecto de cambio constante, se debió agregar un delay adicional para una mejor visibilidad con los LEDs en ambos canales del vúmetro.

Patlani Mauricio Adriana

Continuando con la práctica 13, en esta práctica se hace uso del convertidor análogo a digital, ya que en esta práctica se requirió captar el sonido (señal análoga) usando micrófonos y convertirlos a una señal digital para medir su intensidad y que se representaran mediante LEDs utilizando algún modo que se eligiera: barra, hold o punto para visualizar las frecuencias y su intensidad de volumen.

Ruvalcaba Flores Martha Catalina

Para la realización de esta práctica se utilizó el código de la practica 13, puesto que se requiere de un convertidor analógico digital del microcontrolador para que este sea implementado como un vúmetro de

dos canales. Para ello, se tuvo que programar tres modos para la representación del audio en los leds, logrando obtener las variaciones de tensión de la señal de audio ya sea para el modo barra, el modo hold y el modo punto, de tal forma que, se pudo ver el nivel de volumen en cada momento.

Sandoval Hernández Eduardo

Similar a la práctica anterior, hicimos uso del convertidor analógico digital del microcontrolador, solo que en esta ocasión la entrada fue un par de micrófonos y el resultado del adc en los leds se mostró de forma incremental y no binaria, fue un circuito sencillo de armar, aunque en esta ocasión la codificación fue más difícil debido a que se tuvo que implementar 3 formas distintas de mostrar el vúmetro. Otra complicación encontrada fue que los micrófonos no eran muy sensibles y por tanto era necesario acercar demasiado la fuente de audio para que el microcontrolador pudiera identificar una variación en la entrada del adc.

Bibliografía

• Atmel Corporation. (2006). 8-bit Microcontroller with 8K Bytes In-System Programmable Flash. [Online]. Disponible en: https://drive.google.com/file/d/1acpQaDlsyLHr3w3ReSgrlXRbshZ5Z-lb/view

• Menna. (s.f.) Cómo funciona un convertidor analógico a digital. [Online]. Disponible en: https://como-funciona.co/como-funciona-un-convertidor-analogico-a-digital/

• Isaac. (s.f). Vúmetro: qué es y cómo se puede utilizar este dispositivo. [Online]. Disponible en: https://www.hwlibre.com/vumetro/

• Electrónica de Invierno. (Mayo, 2011). Vúmetro a base de leds. [Online]. Disponible en: https://electronicavm.wordpress.com/2011/05/28/vumetro-monofonico-a-base-de-leds/

• Ingeniatic. (2011). Vúmetro. [Online]. Disponible en: https://www.etsist.upm.es/estaticos/ingeniatic/index.php/tecnologias/item/660-v%C3%BAmetro.html