A dark green vertical bar is positioned on the left side of the page. A green arrow-shaped banner points to the right, containing the date. In the bottom left corner, there are several thin, curved lines in dark green and light gray, resembling stylized grass or reeds.

14-11-2021

# Vúmetro

Martínez Coronel Brayan Yosafat

3CM17 Introducción a los microcontroladores  
FERNANDO AGUILAR SÁNCHEZ

## Objetivo

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

## Introducción teórica

Un conversor o convertidor de señal analógica a digital (Conversor Analógico Digital, CAD; Analog-to-Digital Converter, ADC) es un dispositivo electrónico capaz de convertir una señal analógica, ya sea de tensión o corriente, en una señal digital mediante un cuantificador y codificándose en muchos casos en un código binario en particular. Donde un código es la representación unívoca de los elementos, en este caso, cada valor numérico binario hace corresponder a un solo valor de tensión o corriente.

En la cuantificación de la señal se produce pérdida de la información que no puede ser recuperada en el proceso inverso, es decir, en la conversión de señal digital a analógica y esto es debido a que se truncan los valores entre 2 niveles de cuantificación, mientras mayor cantidad de bits mayor resolución y por lo tanto menor información perdida. Se utiliza en equipos electrónicos como computadoras, grabadores de sonido y de vídeo, y equipos de telecomunicaciones.

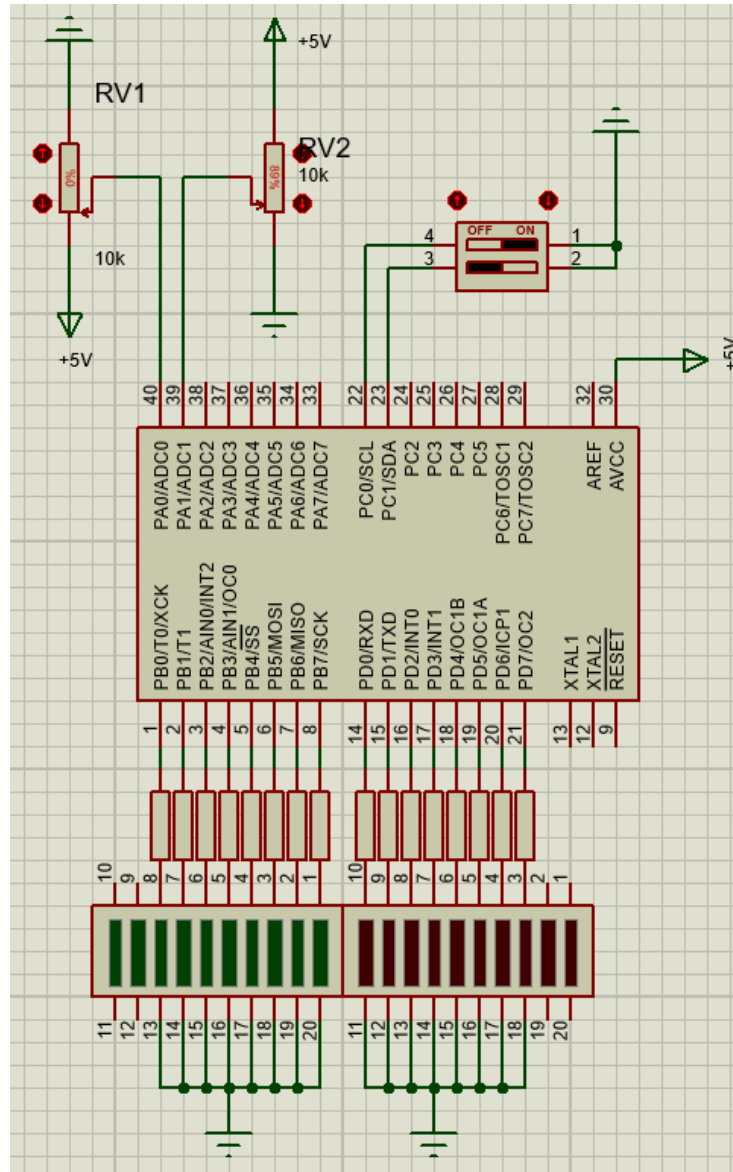
Por otro lado, el vúmetro es un dispositivo indicador en equipos de audio para mostrar el nivel de señal en unidades de volumen, también es llamado indicador del volumen.

## Materiales y Equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- 16 LED
- 16 resistores de 330
- 2 potenciómetro de 10 K

## 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.



## Estructura del programa

/\*\*\*\*\*

This program was created by the CodeWizardAVR V3.46a

Automatic Program Generator

© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.

<http://www.hpinfotech.ro>

Project :

Version :

Date : 31/10/2021

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 btn_tomar_a PINC.0
```

```
#define btn_hold PINC.1
```

```
#define btn_punto PINC.2
```

```
#define btn_barra PINC.3
```

```
char anterior_a, anterior_b, cantidad_a, cantidad_b;
```

```
bit saltar = 0;
```

```
// 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;
}

```

```

unsigned char calcular_intensidad(unsigned char valor, unsigned char cantidad, char modo) {
    char num_de_unos = valor / 31;
    unsigned char intensidad = 0;
    char i;

```

```

    num_de_unos = cantidad < num_de_unos ? num_de_unos - cantidad : 0;

```

```

    for (i = 0; i < num_de_unos; i++) {
        if (modo == 0) {
            if (i == 0)
                intensidad = (intensidad << 1) | 0x01;
            else
                intensidad = intensidad << 1;
        } else { //Barra
            intensidad = (intensidad << 1) | 0x01;
        }
    }
}

```

```

    return intensidad;
}

```

```

// Declare your global variables here

```

```

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)
{
    unsigned char valor;
    unsigned char intensidad;
    unsigned char modo;

    modo = 0; //Barra
    if (!btn_punto) modo = 1;
    if (!btn_hold) modo = 2;

    if (saltar) {
        saltar = ~saltar;
        delay_ms(30);
        continue;
    }

    if (btn_tomar_a) {
        valor = read_adc(0);

        if (valor == anterior_a) {
            if (cantidad_a < 8) cantidad_a++;
        } else
            cantidad_a = 0;
    }
}

```

```

    intensidad = calcular_intensidad(valor, cantidad_a, modo);
    PORTB = intensidad;
    anterior_a = valor;
} else {
    valor = read_adc(1);

    if (valor == anterior_b) {
        if (cantidad_b < 8) cantidad_b++;
    } else
        cantidad_b = 0;

    intensidad = calcular_intensidad(valor, cantidad_b, modo);
    PORTD = intensidad;
    anterior_b = valor;
}

saltar = ~saltar;
delay_ms(30);
}
}

```

## Observaciones y Conclusiones

Fue bastante complicado al inicio, estaba pensando en cómo resolverlo totalmente y no llegaba a nada, me basé en la práctica anterior y empecé a ver subproblemas, me pareció muy divertido ver cómo avancé y descubría otro problema.

## Bibliografía

Vúmetro: <https://es.wikipedia.org/wiki/V%C3%BAmetro>

Convertor de señal analógica a digital:  
[https://es.wikipedia.org/wiki/Convertor\\_de\\_se%C3%B1al\\_anal%C3%B3gica\\_a\\_digital](https://es.wikipedia.org/wiki/Convertor_de_se%C3%B1al_anal%C3%B3gica_a_digital)