

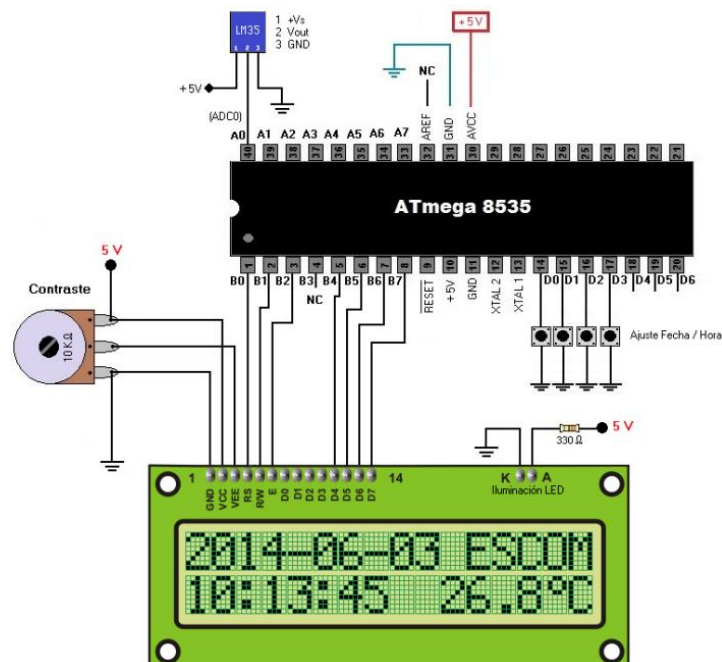


INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS EN CHIP

PRACTICA No. 18

"LCD"



ALUMNO: RENTERIA ARRIAGA JOSUE.

GRUPO: 6SCM1

PROFESOR: FERNANDO AGILAR SÁNCHEZ.

08/DICIEMBRE/2022

I. OBJETIVO GENERAL:

Al término de la sesión, los integrantes del equipo contarán con la habilidad para manejar una pantalla LCD.

II. MATERIAL Y EQUIPO EMPLEADO

- ✓ CodeVision AVR.
- ✓ AVR Studio 4.
- ✓ Microcontrolador ATmega 8535.
- ✓ 1 Pantalla LCD 16x2.
- ✓ 1 Potenciómetro 10K Ω .
- ✓ 1 Resistor de 330 Ω .

III. INTRODUCCIÓN TEÓRICA

En esta práctica podremos saber utilizar correctamente y aplicar la implementación del código en el lenguaje de programación C de los conceptos de realizar un programa con un LCD y ver como los datos ingresados se observan en el LCD.

En esta práctica se estudiará la implementación de un Programa y Circuito para programar el puerto D como entrada de nuestros botones para poder modificar los datos de la fecha y hora y el puerto B donde se mostrarán los datos de nuestro LCD. En este caso de la práctica se ocuparán algunos conceptos obtenidos anteriormente en el curso como lo es la implementación de código en C para poder programar nuestro microcontrolador, algunos conceptos de Fundamentos de Diseño Digital (Implementación de un LCD), concepto de un convertidor analógico a Digital que se vio en Arquitectura y de Electrónica Analógica para poder realizar correctamente el armado de nuestro circuito y que este funcione correctamente.

Ocuparemos algunos elementos como lo dice el formato de la practica ya que ocuparemos el simulador Code Vision para poder implementar el código para después programar nuestro ATmega8535, en mi caso se ocupará el programa Khazama AVR con su programador para poder programar nuestro ATmega con el archivo .hex que se generó anteriormente. Para la parte del armado del circuito será algo sencillo ya que solo se implementará un LCD, un termómetro y botones para modificar el LCD.

Algunos conceptos que se necesitan para poder entender algunos conceptos de la practica son los siguientes:

Tome en cuenta que las siguientes funciones sólo se pueden usar si se inicializó la LCD en el CodeVision:

FUNCIÓN	DESCRIPCIÓN															
lcd_gotoxy(x,y);	Esta función permite indicarle donde colocar el cursor, que es donde empezará a escribir el mensaje. x es la columna, y es la fila.															
lcd_putsf("mensaje");	Con la función anterior le indicamos el mensaje a escribir en la LCD.															
lcd_clear();	Esta función borra el mensaje en el display.															
lcd_putchar('A');	Esta función escribe en pantalla una sólo letra o un carácter.															
lcd_putchar(0x40)	Esta función escribe en pantalla una sólo letra o un carácter en código ASCII.															
 _lcd_ready(); _lcd_write_data(0xFF);	Los bits para controlar el cursor y la pantalla son: Dato=0xFF=00001DCB D=1 Enciende pantalla, D=0 coloca en stand by la pantalla (bajo consumo) C=1 Cursor on, C=0 Cursor Off B=1 Cursor parpadea, B=0 Cursor fijo															
 _lcd_ready(); _lcd_write_data(0xFF);	Los bits para controlar el desplazamiento del cursor y la pantalla son: Dato=0001 S/C R/L 00 Y los bits S/C y R/L tienen la siguiente descripción															
	<table><tr><th>S/C</th><th>R/L</th><th>Descripción</th></tr><tr><td>0</td><td>0</td><td>Desplaza el cursor a la izquierda</td></tr><tr><td>0</td><td>1</td><td>Desplaza el cursor a la derecha</td></tr><tr><td>1</td><td>0</td><td>Desplaza la pantalla y cursor a la izquierda</td></tr><tr><td>1</td><td>1</td><td>Desplaza la pantalla y cursor a la derecha</td></tr></table>	S/C	R/L	Descripción	0	0	Desplaza el cursor a la izquierda	0	1	Desplaza el cursor a la derecha	1	0	Desplaza la pantalla y cursor a la izquierda	1	1	Desplaza la pantalla y cursor a la derecha
S/C	R/L	Descripción														
0	0	Desplaza el cursor a la izquierda														
0	1	Desplaza el cursor a la derecha														
1	0	Desplaza la pantalla y cursor a la izquierda														
1	1	Desplaza la pantalla y cursor a la derecha														

Tabla 2. Funciones para la LCD reconocidas en CodeVision.

Esta práctica nos ayudara a entender cómo se pueden implementar los circuitos con un microcontrolador y todo el proceso que conlleva realizar la programación del microcontrolador.

Con esta introducción se podrá comprender y realizar el desarrollo de la práctica que se desarrolla más adelante.

IV. DESARROLLO EXPERIMENTAL

1. Con la información que a continuación se menciona, diseñe un programa para visualizar en una pantalla LCD 16x2 la fecha, la hora y la temperatura actual, tal y como se muestra en la figura 3. Los botones son para el ajuste de fecha y hora.

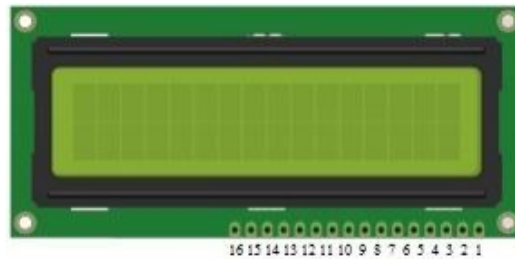


Figura 1. Asignación de pines de la pantalla LCD 16x2.

LCD PIN #	DESCRIPCIÓN	CONEXIÓN
1	VSS	GND
2	VDD	+5V
3	VO	Terminal de ajuste de contraste
4	Register Select (RS)	Conecte al bit del Puerto como indica CodeVision
5	Read/Write (R/W)	Conecte al bit del Puerto como indica CodeVision
6	Clock Enable (E)	Conecte al bit del Puerto como indica CodeVision
7	Data Bit 0	No Conectar
8	Data Bit 1	No Conectar
9	Data Bit 2	No Conectar
10	Data Bit 3	No Conectar
11	Data Bit 4	Conecte al bit del Puerto como indica CodeVision
12	Data Bit 5	Conecte al bit del Puerto como indica CodeVision
13	Data Bit 6	Conecte al bit del Puerto como indica CodeVision
14	Data Bit 7	Conecte al bit del Puerto como indica CodeVision
15	Backlight Anode (+)	+5V
16	Backlight Cathode (-)	GND

Tabla 1. Asignación y conexión de pines de la pantalla LCD 16x2.



Fig.1_Circuito Armado y Simulado en Proteus.

CIRCUITO EN FISICO ARMADO.

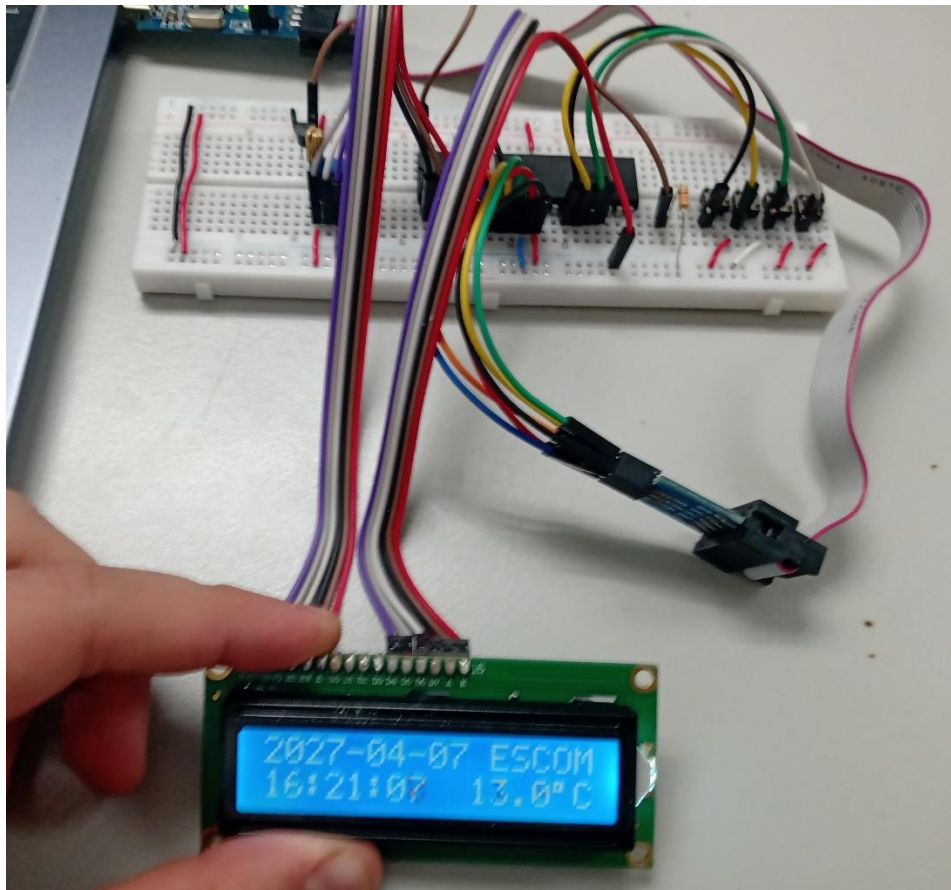


Fig.2_Circuito Armado en Físico (LCD).

V. CÓDIGO

```
#include <mega8535.h>
#include <delay.h>

// Alphanumeric LCD functions
#include <alcd.h>

#define cambio PIND.0
#define ha PIND.1
#define mm PIND.2
#define sd PIND.3

// 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
float cel;
int tem;
int desplz;
int cont_antidelay,time_antidelay;
bit btnp,btna;
unsigned char unidades,decenas,decimas,cn,seg=0,min=0,hor=0,dia=30,mes=12,change;
unsigned short ye=19,ar=99;
const char car=48; //codigo ascii

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

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS: PORTB Bit 0
// RD: PORTB Bit 1
// EN: PORTB Bit 2
// D4: PORTB Bit 4
// D5: PORTB Bit 5
// D6: PORTB Bit 6
// D7: PORTB Bit 7
// Characters/line: 16
lcd_init(16);
desplz=0;
cont_antidelay=0;
time_antidelay=20;
while (1)
{
    delay_ms(50);

    if(cambio==0) btna=0;
    else btna=1;

    if ((btnp==1)&&(btna==0)) {
        if(change==0) change=1;
        else change=0;
    }

    btnp=btna;

    lcd_gotoxy(11,0);
    lcd_putsf("ESCOM");
    cn=read_adc(0);
    cel=50 * cn / 255;
    if(cel>99) cel=99;

    tem=cel*10;
    decenas=tem/100;
    tem%=100;
    decimas=tem%10;
    unidades=tem/10;
    lcd_gotoxy(10,1);
    lcd_putchar(decenas+car);
    lcd_gotoxy(11,1);
    lcd_putchar(unidades+car);
    lcd_gotoxy(12,1);
    lcd_putchar('.');
}

```

```

lcd_gotoxy(13,1);
lcd_putchar(decimas+car);
lcd_gotoxy(14,1);
lcd_putchar(car+175);
lcd_gotoxy(15,1);
lcd_putchar('C');

if(change==1) {
    if(ha==0) {
        if(cont_antidelay>time_antidelay) { cont_antidelay=0; hor++; }
        else cont_antidelay++;
    }

    if(mm==0) {
        if(cont_antidelay>time_antidelay){cont_antidelay=0; min++;}
        else cont_antidelay++;
    }

    if(sd==0) {
        if(cont_antidelay>time_antidelay){ cont_antidelay=0; seg++; }
        else{ cont_antidelay++; }
    }
} else {
    if(ha==0) {
        if(cont_antidelay>time_antidelay){
            cont_antidelay=0;
            ar++;
            if(ar>99){ ye++; ar=0; }
        } else {
            cont_antidelay++;
        }
    }

    if(mm==0) {
        if(cont_antidelay>time_antidelay){ cont_antidelay=0; mes++; }
        else{ cont_antidelay++; }
    }

    if(sd==0) {
        if(cont_antidelay>time_antidelay){ cont_antidelay=0; dia++; }
        else{ cont_antidelay++; }
    }
}

if(desplz>49) {
    desplz=0;
    seg++;
} else {
    desplz++;
}

if(seg>59){ min++; seg=0; }
if(min>59){ hor++; min=0; seg=0; }

```

```

    if(hor>23){ dia++; hor=0; seg=0; min=0; }
    if(dia>31){ mes++; dia=0; }
    if(mes>12){
        ar++;
        mes=0;
        if(ar>99){ ye++; ar=0; }
    }

    lcd_gotoxy(0,1);
    lcd_putchar(hor/10+car);
    lcd_gotoxy(1,1);
    lcd_putchar(hor%10+car);
    lcd_gotoxy(2,1);
    lcd_putchar(':');
    lcd_gotoxy(3,1);
    lcd_putchar(min/10+car);
    lcd_gotoxy(4,1);
    lcd_putchar(min%10+car);
    lcd_gotoxy(5,1);
    lcd_putchar(':');
    lcd_gotoxy(6,1);
    lcd_putchar(seg/10+car);
    lcd_gotoxy(7,1);
    lcd_putchar(seg%10+car);

    lcd_gotoxy(0,0);
    lcd_putchar(ye/10+car);
    lcd_gotoxy(1,0);
    lcd_putchar(ye%10+car);
    lcd_gotoxy(2,0);
    lcd_putchar(ar/10+car);
    lcd_gotoxy(3,0);
    lcd_putchar(ar%10+car);
    lcd_gotoxy(4,0);
    lcd_putchar('-');
    lcd_gotoxy(5,0);
    lcd_putchar(mes/10+car);
    lcd_gotoxy(6,0);
    lcd_putchar(mes%10+car);
    lcd_gotoxy(7,0);
    lcd_putchar('-');
    lcd_gotoxy(8,0);
    lcd_putchar(dia/10+car);
    lcd_gotoxy(9,0);
    lcd_putchar(dia%10+car);

    delay_ms(50);
}
}

```

VI. OBSERVACIONES Y CONCLUSIONES

Gracias a la realización de la práctica podemos observar cómo se utiliza e implementa un circuito con un microcontrolador (ATMega8535) y como gracias a su programador pudimos logra programar el integrado con nuestro código en lenguaje C. En esta práctica al momento de armar el circuito y simularlo pudimos observar cómo se implementaría un LCD y que formato le daría en mi caso implemente como se decía en la práctica, que realmente no fue tan difícil gracias al video explicativo.

Tras la investigación y explicación de la práctica previa me quedo un poco más claro como poder implementar un LCD y programar el ATMega8535. También al observar el data set del LCD que estaba en la documentación y se comprendido mejor como conectarlo en físico este componente.

Para concluir esta práctica podemos decir que fue una experiencia interesante ya que decidí yo programar la forma de los botones y como cambiarla en el código para el LCD. Gracias a la anterior practica ya fue más fácil de hacerlo, ya que es casi lo mismo que esta.

VII. BIBLIOGRAFÍA

- ✓ CodeVisionAVR. (2018). CodeVisionAVR. 20/Septiembre/2022, de CodeVisionAVR Sitio web: <http://www.hpinfotech.ro/cvavr-download.html>
- ✓ Wikipedia. (2018). Vúmetro: qué es y cómo se puede utilizar este dispositivo. 05/Noviembre/2022, de Wikipedia, Sitio web: <https://www.hwlibre.com/vumetro/>