

TUTORIAL SENSOR DE HUMEDAD Y TEMPERATURA EN LCD CON ATMEGA32

tutorial para www.comunidadatmel.com

Por **Óscar Razo Navarrete**

INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE ZAMORA

DPTO. DE INGENIERIA ELECTRÓNICA.

En este tutorial se va a llevar a cabo la integración de un sensor de humedad y temperatura, utilizando un microcontrolador ATMEGA32 y un sensor, con matricula HMZ433A1, de tal manera que se puedan ver los datos en una pantalla LCD.

Algunas de las características del sensor son:

- Es económico
- Entrega una salida lineal de voltaje para la lectura de humedad que va de 0 a 3.3 volts
- Tiene un termistor integrado del tipo NTC para medir la temperatura
- Tamaño reducido
- Se alimenta con 5 voltios
- Bajo consumo de corriente

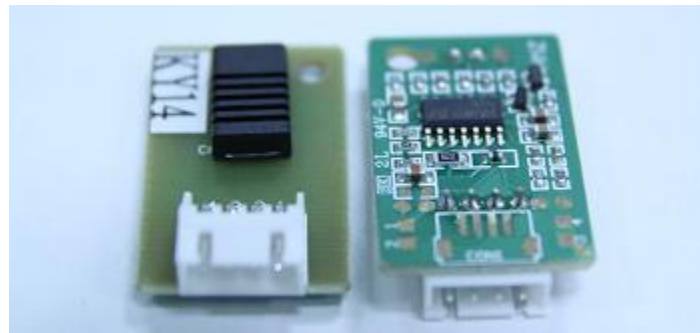


Imagen del sensor HMZ433A1

Humedad relativa

La **humedad relativa** es la humedad que contiene una masa de aire, en relación con la máxima humedad absoluta que podría admitir sin producirse condensación, conservando las mismas condiciones de temperatura y presión atmosférica. Esta es la forma más habitual de expresar la

$$RH = \frac{P(H_2O)}{P^*(H_2O)} \times 100\%$$

humedad ambiental. Se expresa en tanto por ciento. %

donde

$P(H_2O)$ es la [presión parcial](#) de vapor de agua en la mezcla de aire;

$P^*(H_2O)$ es la [presión de saturación](#) de vapor de agua a la temperatura en la mezcla de aire; y

RH es la humedad relativa de la mezcla de aire que se está considerando.

La importancia de esta manera de expresar la humedad ambiente estriba en que refleja muy adecuadamente la capacidad del aire de admitir más o menos vapor de agua, lo que, en términos de [comodidad ambiental](#) para las personas, expresa la capacidad de evaporar la [transpiración](#), importante regulador de la temperatura del cuerpo humano.

Introducción

Un Termistor es un resistor sensible a la temperatura. La palabra que mejor describe a los termistores es la sensibilidad. El termistor exhibe gran cambio en el parámetro resistencia en función de pequeños cambios de temperatura.

Los termistores están generalmente compuestos de materiales semiconductores y existen básicamente dos tipos: los de coeficiente negativo de temperatura (NTC) y los de coeficiente positivo de temperatura (PTC). Los primeros son los más usados y disminuyen su resistencia con el incremento de temperatura.

El precio que se paga por el incremento en la sensibilidad, es la pérdida de linealidad.

Efectivamente, el termistor es un dispositivo extremadamente no lineal y su curva característica varía según sea el fabricante.

La curva característica de un termistor individual puede ser aproximada a través del uso de la ecuación de Steinhart-Hart:

$$1/T = A + B(\ln R) + C(\ln R)^3$$

Donde:

T = Temperatura en °Kelvin

R = Resistencia del Termistor en Kohm

A, B, C = Constante de la curva de aproximación

Las constantes A, B, y C pueden ser calculadas seleccionando tres puntos de la tabla o curva que acompaña el termistor y resolviendo un sistema de ecuaciones simultáneas de tres incógnitas.

Propósito

Este trabajo tiene como propósito la realización de un Termómetro e higrómetro digital basado en un termistor 50Kohm +/- 1% @ 25°C y un ATMEGA32, el cual resulta apropiado por tener convertidores analógico digital de 10 bits de resolución y suficientes líneas de I/O para el manejo de un módulo LCD, aunque se puede utilizar cualquier ATMEGA que esté disponible.

Cálculo de las constantes de la ecuación de Steinhart-Hart

El termistor usado tiene la siguiente tabla característica:

TEMPERATURA (°C)		0	10	20	25	30	40	50	60
RESISTENCIA (KΩ)		160.56	98.714	62.328	50	40.356	26.756	18.138	12.554

Para el cálculo de las constantes se resuelve el sistema de ecuaciones simultáneas de tres incógnitas para los puntos 0°C, 25°C y 50°C. Recuerde que °Kelvin = °C + 273.15.

$$1 + \ln 160.56 + (\ln 160.56)^3 = 1/(0 + 273)$$

$$1 + \ln 50 + (\ln 50)^3 = 1/(25 + 273)$$

$$1 + \ln 18.138 + (\ln 18.138)^3 = 1/(50 + 273)$$

Este sistema de ecuaciones se puede resolver en MATLAB o en una calculadora gráfica, los cuales entregaron el siguiente resultado:

$$A= 0.00237531 \quad B= 0.00024632 \quad C= 0.00000028$$

Entonces, la ecuación de Steinhart-Hart para este termistor es:

$$T^{\circ}\text{C} = \left(\frac{1}{0.00237531 + 0.00024632 \ln(R) + 0.00000028 \ln(R)^3} \right) - 273$$

Interfaz Termistor-ATMEGA

La interfaz Termistor-ATMEGA, consiste en un divisor de voltaje realizado con el termistor (Rterm) y una resistencia de 50K (R1), tal como se muestra en la fig. 1. Como voltaje de referencia se utiliza +5V.

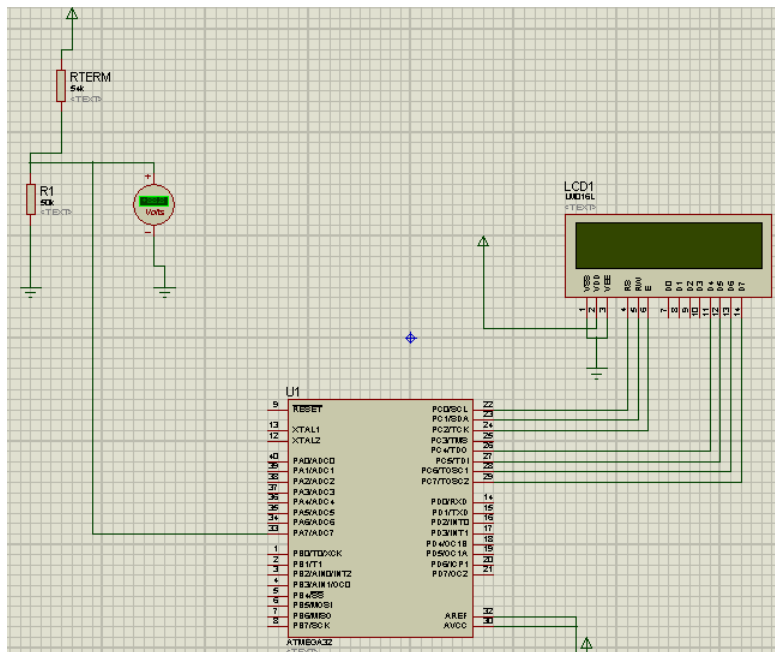


Fig. 1 interfaz termistor Atmega

Como se explica anteriormente, se debe conectar un resistor de 50K en el pin número 4 para hacer un divisor de voltaje, ya que la señal de temperatura que nos entrega el sensor no es lineal, a diferencia de la señal de humedad, el fabricante también recomienda colocar un capacitor de 0.1 μ F en el pin 2 que es la señal de humedad.

Para obtener el resistor de 50K yo hice un arreglo con 2 resistores de 100K en paralelo.

NOTA: Es muy recomendable revisar la hoja de datos del sensor para entender mejor su funcionamiento.

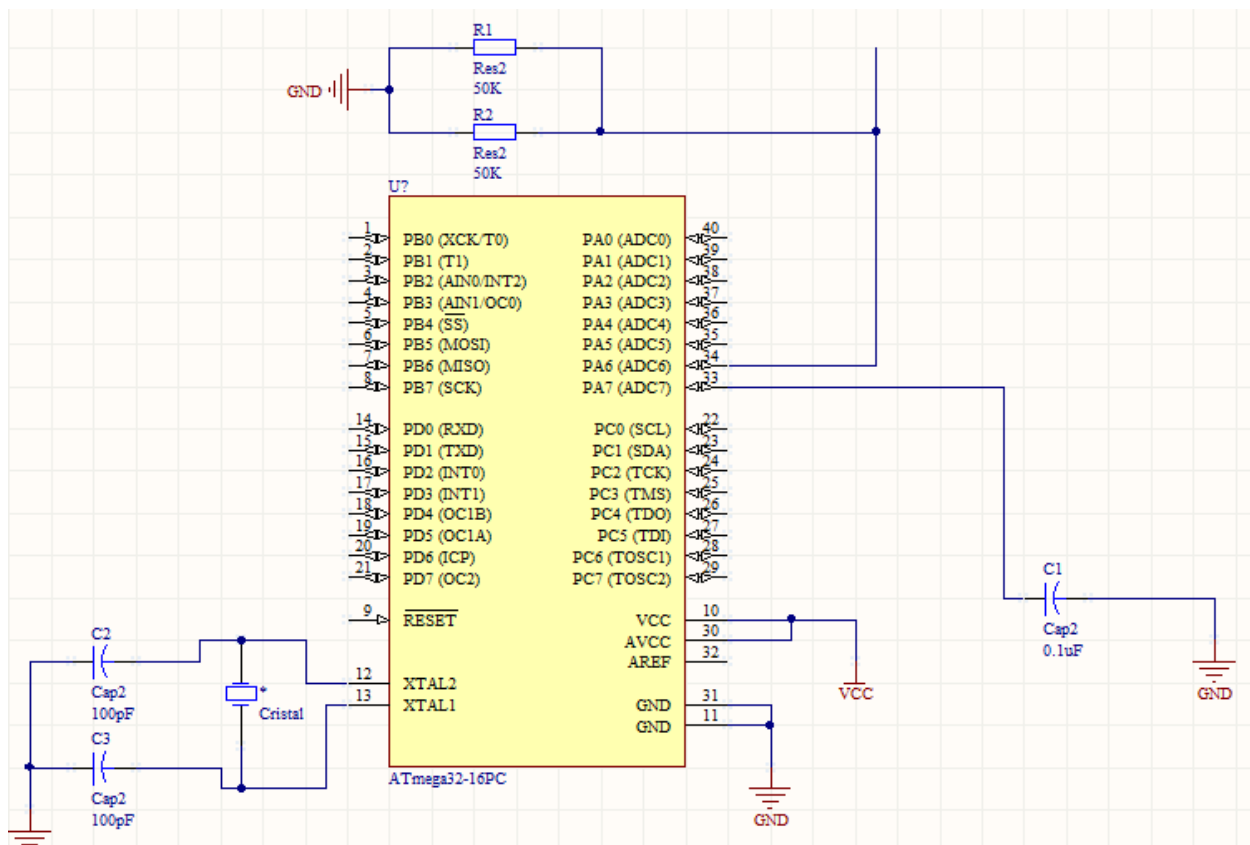


Diagrama esquemático

Se debe conectar la señal de humedad del sensor al ADC7 y la señal de temperatura al ADC6, o a cualquier otro ADC de otro ATMEGA si así se desea.

A continuación lo que se debe hacer es generar nuestro programa en el codevision para usar el ATMEGA a 8MHz sin dividir la frecuencia, el ADC en AVCC ref CON 10 BITS DE RESOLUCION, la pantalla LCD.

Este es el programa generado y lo que está en color azul es lo que se debe agregar:

/******

This program was produced by the

CodeWizardAVR V2.04.4a Advanced

Automatic Program Generator

© Copyright 1998-2009 Pavel Haiduc, HP InfoTech s.r.l.

<http://www.hpinfotech.com>

Project : RH% y TEMP

Version :

Date : 5/8/2010

Author : NeVaDa

Company : Hewlett-Packard

Comments:

sensor de humedad relativa y temperatura

Chip type : ATmega32

Program type : Application

AVR Core Clock frequency: 8.000000 MHz

Memory model : Small

External RAM size : 0

Data Stack size : 512

*****/

`#include <math.h> //se agrega esta libreria para hacer operaciones logaritmicas`

```

#include <mega32.h>

#include <stdio.h>

#include <delay.h>

int var1;

unsigned int var2=50000;

float Ln_Resist;

float Resist;

float Temp;

float Ln_Resist3;

void humedad();

void despliega();

void despliega2();

void calcular_temp();

// Alphanumeric LCD Module functions

#asm

.equ __lcd_port=0x15 ;PORTC

#endasm

#include <lcd.h>

//#define ADC_VREF_TYPE 0x00 //para usar AREF

#define ADC_VREF_TYPE 0x40 //para usar AVCC

// Read the AD conversion result

unsigned int read_adc(unsigned char adc_input)

{

```

```
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);  
  
// Delay needed for the stabilization of the ADC input voltage  
delay_us(10);  
  
// Start the AD conversion  
ADCSRA|=0x40;  
  
// Wait for the AD conversion to complete  
while ((ADCSRA & 0x10)==0);  
  
ADCSRA|=0x10;  
return ADCW;  
}
```

```
//LCD putint
```

```
unsigned char buff[33];
```

```
void lcd_putint(unsigned int dat)
```

```
{
```

```
    sprintf(buff,"%d ",dat);
```

```
    lcd_puts(buff);
```

```
}
```

```
// Declare your global variables here
```

```
void main(void)
```

```
{
```

```
// Declare your local variables here
```

```
// Input/Output Ports initialization
```

```
// Port A initialization
```

```
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
```

```
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
```

```
PORTA=0x00;
```

```
DDRA=0x00;
```

```
// Port B initialization
```

```
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
```

```
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
```

```
PORTB=0x00;
```

```
DDRB=0x00;
```

```
// Port C initialization
```

```
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
```

```
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
```

```
PORTC=0x00;
```

```
DDRC=0x00;
```

```
// Port D initialization
```

```
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
```

```
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
```

```
PORTD=0x00;
```

```
DDRD=0x00;
```

```
// Timer/Counter 0 initialization
```

```
// Clock source: System Clock
```

```
// Clock value: Timer 0 Stopped
```

```
// Mode: Normal top=FFh
```

```
// OCO output: Disconnected
```

```
TCCR0=0x00;
```

```
TCNT0=0x00;
```

```
OCR0=0x00;
```

```
// Timer/Counter 1 initialization
```

```
// Clock source: System Clock
```

```
// Clock value: Timer1 Stopped
```

```
// Mode: Normal top=FFFFh
```

```
// OC1A output: Discon.
```

```
// OC1B output: Discon.
```

```
// 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=0x00;
```

```
TCCR1B=0x00;
```

```
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=FFh
```

```
// OC2 output: Disconnected
```

```
ASSR=0x00;
```

```
TCCR2=0x00;
```

```
TCNT2=0x00;
```

```
OCR2=0x00;
```

```
// External Interrupt(s) initialization
```

```
// INT0: Off
```

```
// INT1: Off
```

```
// INT2: Off
```

```
MCUCR=0x00;
```

```
MCUCSR=0x00;
```

```
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization6
// ADC Clock frequency: 500.000 kHz
// ADC Voltage Reference: AREF pin
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// LCD module initialization
lcd_init(16);

while (1)
{
    // Place your code here
```

```
    calcular_temp();
```

```
    lcd_gotoxy(8,1);
```

```
    lcd_putchar(0xdf);
```

```
    lcd_putsf("C");
```

```
    lcd_gotoxy(0,1);
```

```
    lcd_putsf("TEMP:");
```

```
    despliega2();
```

```
    lcd_gotoxy(9,0);
```

```
    lcd_putsf("%");
```

```
    lcd_gotoxy(0,0);
```

```
    lcd_putsf("RH[%]:");
```

```
    humedad();
```

```
    despliega();
```

```
    delay_ms(500);
```

```
};
```

```
}
```

```
void humedad()
```

```
{
```

```
var1=((read_adc(7))*5/3.3)/10.2;
```

```
}
```

```
void despliega()
```

```
{
```

```
lcd_putint(var1);
```

```
}
```

```
void despliega2()
```

```
{
```

```
lcd_putint(Temp);
```

```
}
```

```
void calcular_temp()
```

```
{
```

```
var2=1024-read_adc(6);
```

```
Resist=(54*var2)/(1024-var2);
```

```
Ln_Resist =log(Resist);
```

```
Ln_Resist3=Ln_Resist*Ln_Resist*Ln_Resist;
```

```
Temp = (1/(0.00237531+0.00024632*Ln_Resist+0.00000028*Ln_Resist3)-273);
```

```
}
```

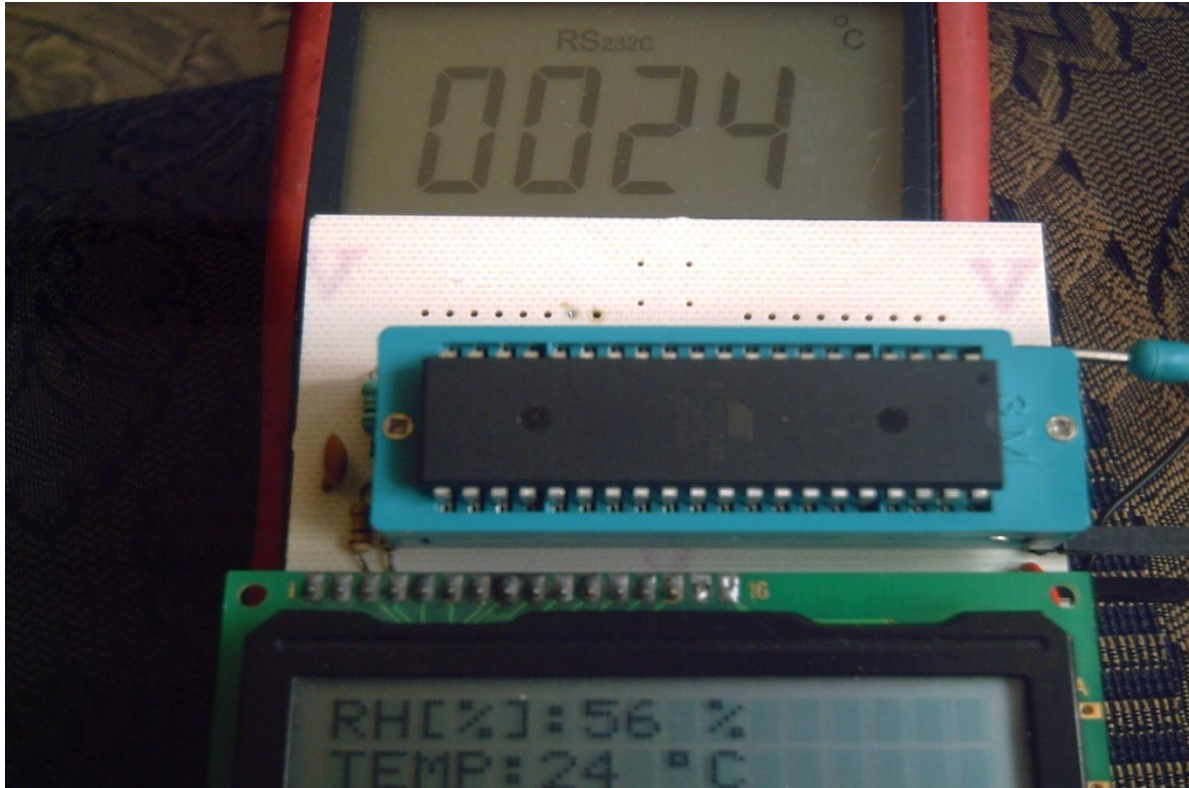
Bueno, para este programa fué necesario agregar una librería llamada math.h que nos va ayudar a hacer las operaciones logarítmicas, también debemos declarar nuestras variables como se muestra en el programa, así como las funciones que se van a utilizar para desplegar datos y hacer las operaciones correspondientes, tal y como se ha hecho en otros tutoriales.

Como se vio anteriormente la parte más difícil es hacer los cálculos para el termistor ya que no es lineal, pero ya teniendo los valores correspondientes solamente aplicamos la ecuación de STEINHART dentro de nuestra función de calcular temperatura, para la humedad solamente se hacen los cálculos para acoplar el valor del ADC que va de 0 a 5 volts y el sensor que va de 0 a 3.3 volts.

Una aplicación para este circuito podría ser en un invernadero, en donde a veces es necesario mantener ciertos niveles de temperatura y humedad para ciertos alimentos o plantas, también se pueden regular los niveles de oxígeno, solamente es cuestión de acoplar un sensor para oxígeno.



Esta es una foto del circuito ya funcionando, el circuito está montado en una baquelita fabricada por medio de ALTIUM DESIGNER que es un programa para realizar estos diseños.



Aquí se muestra una fotografía en donde se hace una analogía con un multímetro el cual tiene integrado un termómetro, en donde se aprecia claramente la precisión de nuestro diseño.

Ya para finalizar solamente me resta agradecer su atención e invitar a los lectores a aportar más tutoriales para compartir el conocimiento.

Muchas gracias.

Atte. Óscar Razo Navarrete

INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE ZAMORA

INGENIERÍA ELECTRÓNICA

www.comunidadatmel.com

JULIO DE 2010