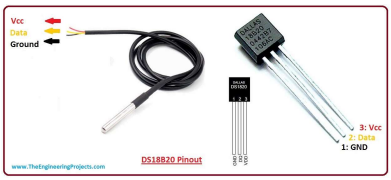
Microcontroladores

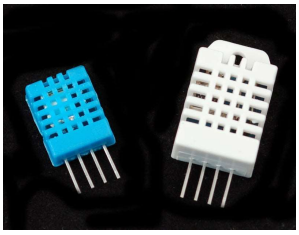
Semana 14

Por Kalun Lau

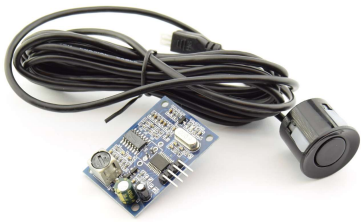
1

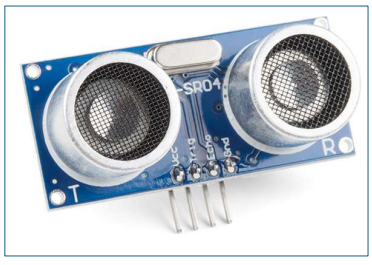
Agenda:





- Manejo del DHT11/DHT22
- Manejo del HC-SR04
- Funcionamiento del DS18B20 (1-wire)

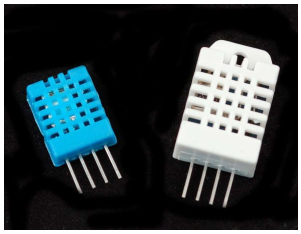




2

DHT11/DHT22

- Sensores para medir la humedad y temperature del ambiente.
- Ampliamente usados en proyectos de sistemas embebidos con Arduino.



DHT11

- Ultra low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

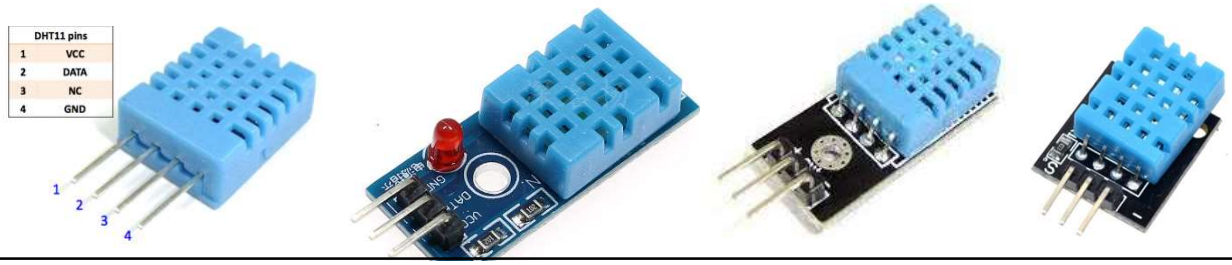
DHT22 / AM2302 **(Wired version)**

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 0-100% humidity readings with 2-5% accuracy
- Good for -40 to 80°C temperature readings ±0.5°C accuracy
- No more than 0.5 Hz sampling rate (once every 2 seconds)
- Body size 15.1mm x 25mm x 7.7mm
- 4 pins with 0.1" spacing

3

El DHT11

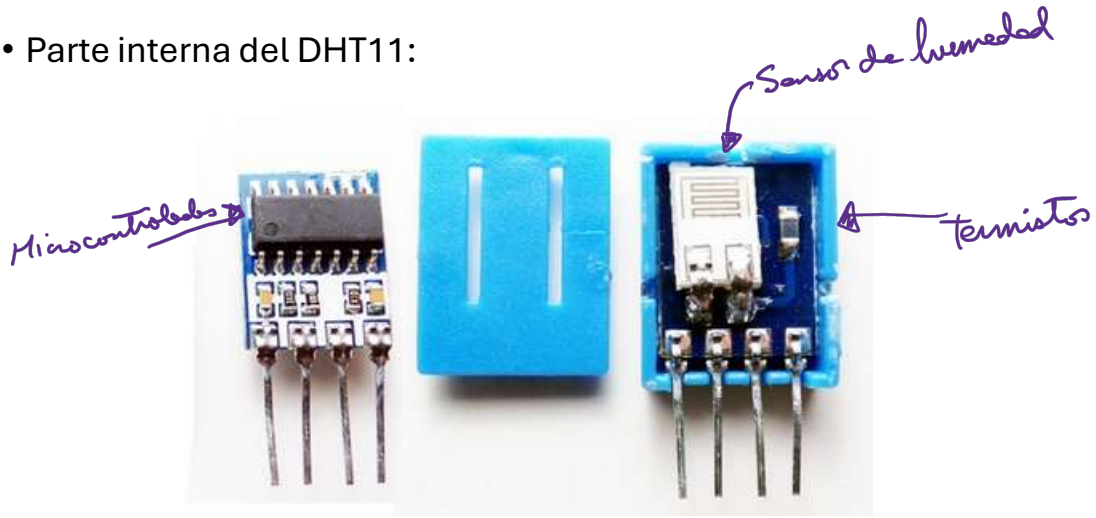
- Aspectos iniciales:
  - Al revisar la hoja técnica del DHT11 podemos ver que el DHT11 tiene un rango de voltaje de operación de 3V a 5.5V por lo que la conexión hacia el Curiosity Nano IC18F57Q43 será de manera directa.
  - Dependiendo del modelo de DHT11 puede que tenga integrado la resistencia de pull-up, sobre todo lo que tienen el sensor montado en una PCB:
  - Hoja técnica: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf



4

El DHT11

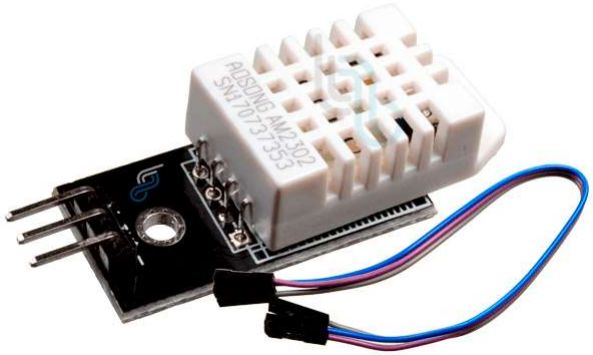
- Parte interna del DHT11:



5

El DHT22

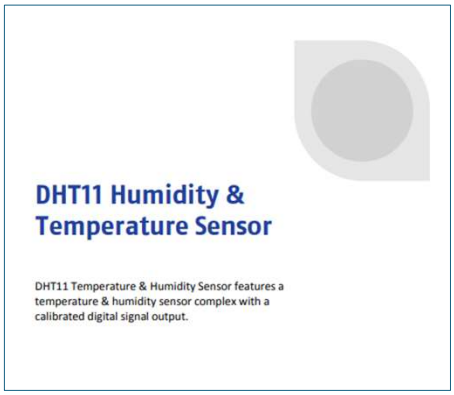
- También se le puede encontrar como AM2302 AOSONG
- De funcionamiento similar al DHT11, pero con mayor resolución en las medidas de temperatura y humedad



6

Hoja técnica (datasheet) del DHT11/DHT22

- El DHT11 es producido en China
- La hoja técnica originalmente esta en chino y fue traducido de manera artesanal.



DHT11 Humidity & Temperature Sensor

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output.

Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connect and special packages can be provided according to users' request.

2. Technical Specifications:

Overview:

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

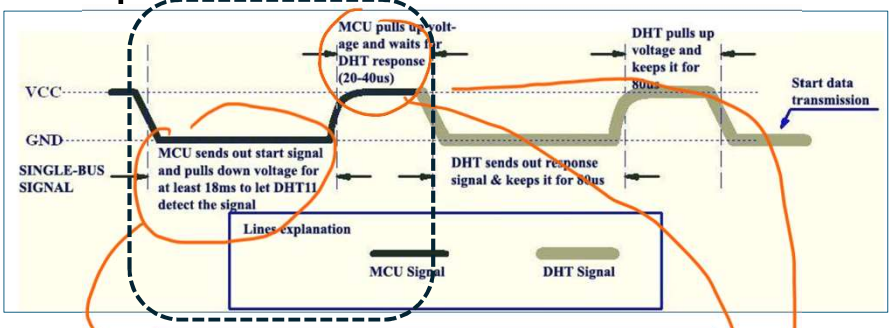
Page | 3

Detailed Specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH 8 Bit	1%RH

7

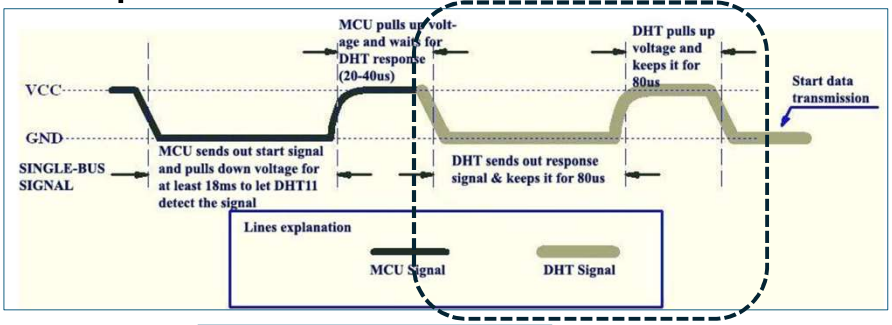
Inicio del proceso de comunicación con el DHT11




```
void DHT11_Start(void){
    TRISBbits.TRISB0 = 0;       //Puerto RB0 como salida
    ANSELBbits.ANSELB0 = 0;     //Para que sea digital
    LATBbits.LATB0 = 0;         //Mandamos cero
    __delay_ms(18);             //Esperamos 18 milisegundo
    LATBbits.LATB0 = 1;         //Mandamos uno
    __delay_us(40);             //Esperamos 40 microsegundos
    TRISBbits.TRISB0 = 1;       //Puerto como entrada
}
```


8

Inicio del proceso de comunicación con el DHT11




```
void DHT11_Check(void){
    while(PORTBbits.RB0);
    __delay_us(80);
    while(!PORTBbits.RB0);
    __delay_us(80);
    while(PORTBbits.RB0);
}
```

while (PORTBbits.RB0 == 1)

while (PORTBbits.RB0 != 1)

9

Proceso de envío de datos del DHT11 al MCU

- En el proceso de envío de datos (son 5 bytes ó 40 bits)

Data consists of decimal and integral parts. A complete data transmission is **40bit**, and the sensor sends **higher data bit** first.
Data format: 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum. If the data transmission is right, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

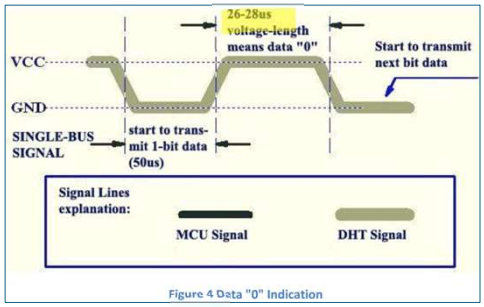


Figure 4 Data "0" Indication

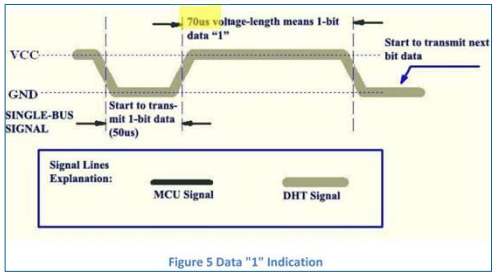


Figure 5 Data "1" Indication

```
unsigned char DHT11_Read(void){
    unsigned char x = 0, data = 0;
    for(x=0;x<8;x++){
        while(!PORTBbits.RB0);
        __delay_us(35);
        if(PORTBbits.RB0){
            data = ((data<<1) | 1);
        }
        else{
            data = (data<<1);
        }
        while(PORTBbits.RB0);
    }
    return data;
}
```

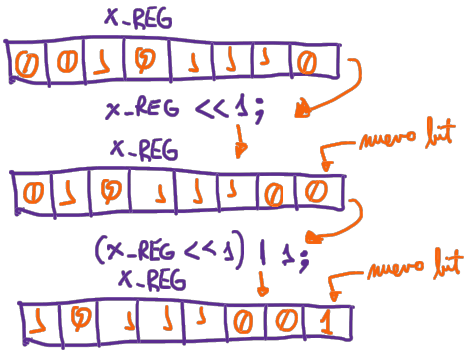
Lee un dato de ocho bits

Haciendo cinco veces!

5

10

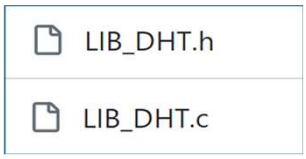
Entendiendo las operaciones de desplazamiento de datos en un registro



11

La nueva librería DHT (LIB_DHT)

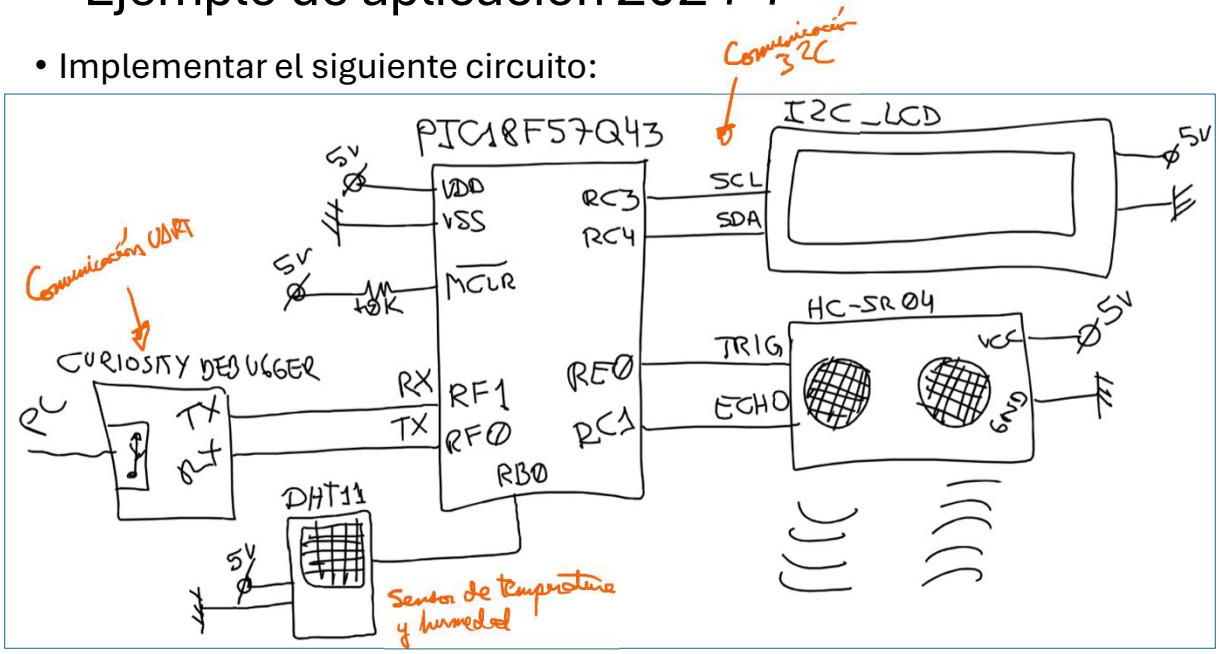
- Referencia: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/ELEC225%202024-1%20Examples/Prueba_DHT22_I2CLCD.X
- Soporta dispositivos DHT11 y DHT22



12

Ejemplo de aplicación 2024-1

- Implementar el siguiente circuito:



13

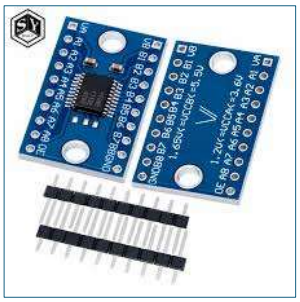
Uso de las librerías I2C_LCD y LIB_DHT

```
#include "I2C_LCD.h"
#include "LIB_DHT.h"

unsigned int temperatura=0;
unsigned int humedad=0;

void configuro(void){
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;
    OSCEN = 0x40;
}
void main(void) {
    configuro();
    I2C_LCD_INIT();
    while(1){
        temperatura = DHT_GetTemp(DHT22);
        __delay_ms(500);
        humedad = DHT_GetHumid(DHT22);
        __delay_ms(500);
        I2C_POS_CURSOR(1,0);
        I2C_ESCRIBE_MENSAJE2("Temp: ");
        I2C_LCD_ESCRIBE_VAR_INT(temperatura, 3, 1);
        I2C_ENVIA_LCD_DATA(0xDF);
        I2C_ENVIA_LCD_DATA('C');
        I2C_POS_CURSOR(2,0);
        I2C_ESCRIBE_MENSAJE2("Humid: ");
        I2C_LCD_ESCRIBE_VAR_INT(humedad, 3, 1);
        I2C_ESCRIBE_MENSAJE2("% RH");
    }
}
```


14

El sensor HC-SR04

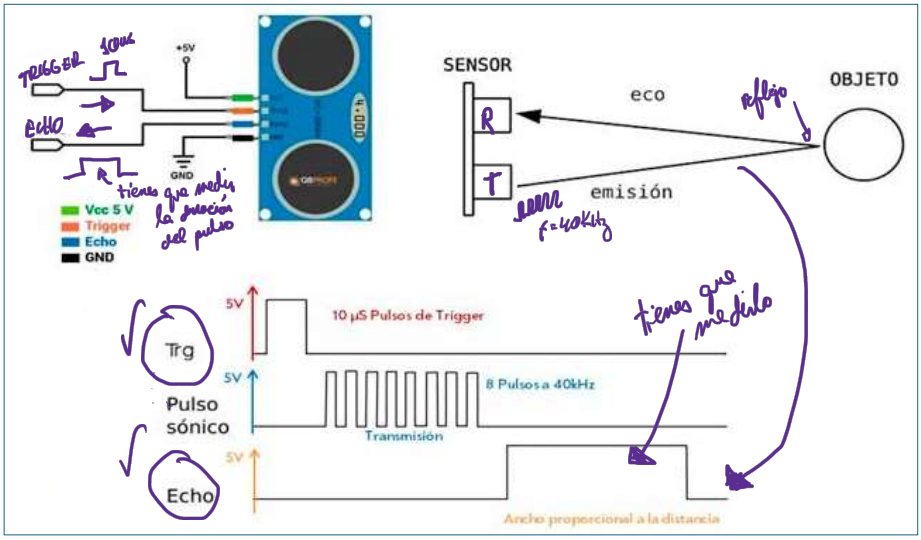
- El sensor HC-SR04 es un módulo que permite medir la distancia entre éste y un objeto entre un rango de 3cm y 300cm.
- Trabaja con alimentación de 5V por lo que se necesitará de un conversor de niveles lógicos 3.3V-5V, se recomienda emplear el TXS0108 en caso de que el microcontrolador trabaje en LVTTL
- Los transductores trabajan en 40KHz de frecuencia de sonido (rango ultrasónico – no es audible)



15

El sensor HC-SR04

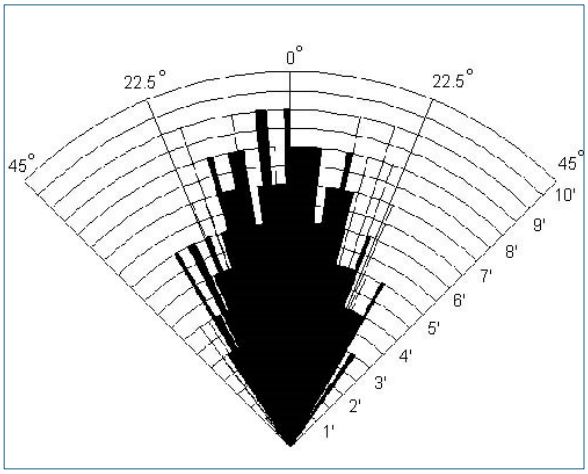
- El sensor envía una ráfaga de 8 pulsos a 40KHz.
- El sonido viaja a una distancia doble con respecto al objeto donde se reflejó el sonido
- El sonido viaja a 340m/s a nivel del mar y a 20°C



16

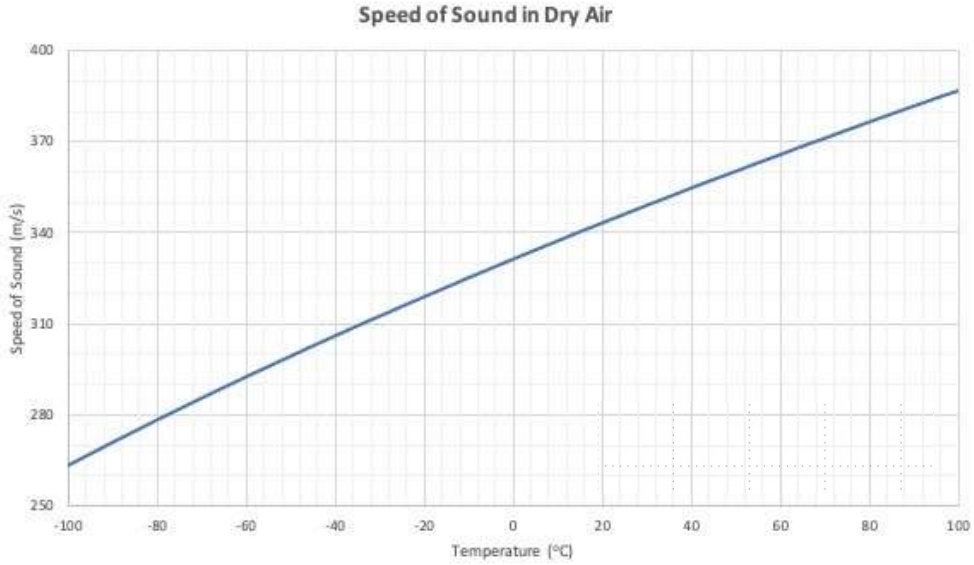
El sensor HC-SR04

- Tener en cuenta que la emisión ultrasónica no es muy directiva, a mayor distancia que recorre el sonido emitivo mayor será la dispersión.
- Mínima distancia a medir: 2cm
- Máxima distancia a medir: 4m (sin interferencias)



17

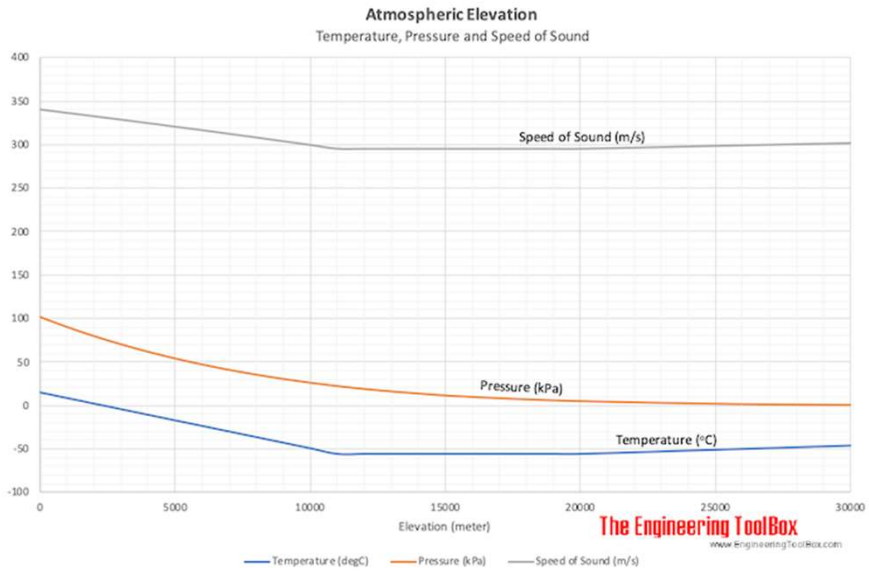
Velocidad del sonido con respecto a la temperatura



18

Velocidad del sonido con respecto a la temperatura y la presión atmosférica

Referencia:
https://www.engineeringtoolbox.com/elevation-speed-sound-air-d_1534.html



El sensor HC-SR04

- Cálculo de la distancia entre el sensor y el obstáculo:

1 m = 1000 cm

34300 cm/s

1 s = 1 x 10^6 µs

0.0343

Velocidad del sonido: 343 m/s (a nivel del mar y a 20°C)

↳ 0.0343 cm/µs ← convertido según resolución del Timer1 y las distancias a medir (de 2 cm a 4000 cm)

4 metros en condiciones ideales

$$\text{Distancia} = \frac{(\text{velocidad del sonido}) \times \text{duración}}{2}$$

Timer1

$$\text{Distancia} = \frac{(0.0343)\ \text{cm/µs} \times \text{duración en microsegundos}}{2 \leftarrow \text{ida y vuelta}} = 0.01715 \times \text{duración} \Rightarrow \frac{\text{duración}}{58.31}$$

lo pasamos a cociente

Timer1
1 µs x ciclo

$$\text{Distancia (cm)} = \frac{\text{Tiempo (µs)}}{58.31}$$

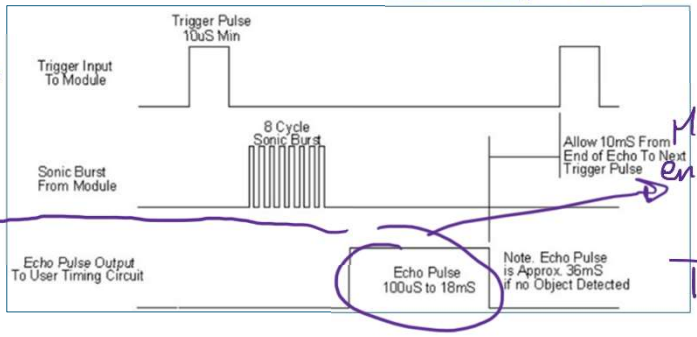
El sensor HC-SR04

- Posee dos puertos:
  - Trigger: El microcontrolador host le envía al puerto "trigger" del HC-SR04 un pulso activo en alto de 10us para que este último envíe ocho pulsos de sonido de 40KHz.
  - Echo: Luego del envío de los ocho pulsos de sonido de 40KHz el HC-SR04 enviará un pulso al microcontrolador host con determinado ancho, este ancho representará el tiempo del eco y mediante un cálculo matemático se obtendrá la distancia entre el módulo y el objeto.

```
T1CLK = 0x01;       //Timer1 con fuente de reloj FOSC/4
T1CON = 0x32;       //presc 1:8, rd16=1, async, timer1 OFF = 1us x cta
```

base de tiempo

```
LATEbits.LATE0 = 1;
__delay_us(10);
LATEbits.LATE0 = 0;
while(PORTCbits.RC0 == 0);
T1CONbits.ON = 1;
while(PORTCbits.RC0 == 1);
T1CONbits.ON = 0;
```



Medir el tiempo en este pulso

Timer1

21

El sensor HC-SR04

- Código propuesto para leer la distancia usando el HC-SR04

```
#include <xc.h>
#include "cabecera.h"
#include "I2C_LCD.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;
    OSCEN = 0x40;
    //Configuracion de E/S para los HCSR04
    TRISE = 0xFE;       //RE0 como salida
    ANSELE = 0xFE;      //RE0 como digital
    TRISCbits.TRISC1 = 1;   //RC1 como entrada
    ANSELCbits.ANSELC1 = 0;   //RC1 como digital
    T1CLK = 0x01;       //Timer1 con fuente de reloj FOSC/4
    T1CON = 0x32;       //presc 1:8, rd16=1, async, timer1 OFF = 1us x cta
}
```

```
unsigned int Read_HCSR04(void){
    TMR1H = 0;
    TMR1L = 0;
    LATEbits.LATE0 = 1;
    __delay_us(10);
    LATEbits.LATE0 = 0;
    while(PORTCbits.RC1 == 0);
    T1CONbits.ON = 1;
    while(PORTCbits.RC1 == 1);
    T1CONbits.ON = 0;
    return (TMR1);
}
```

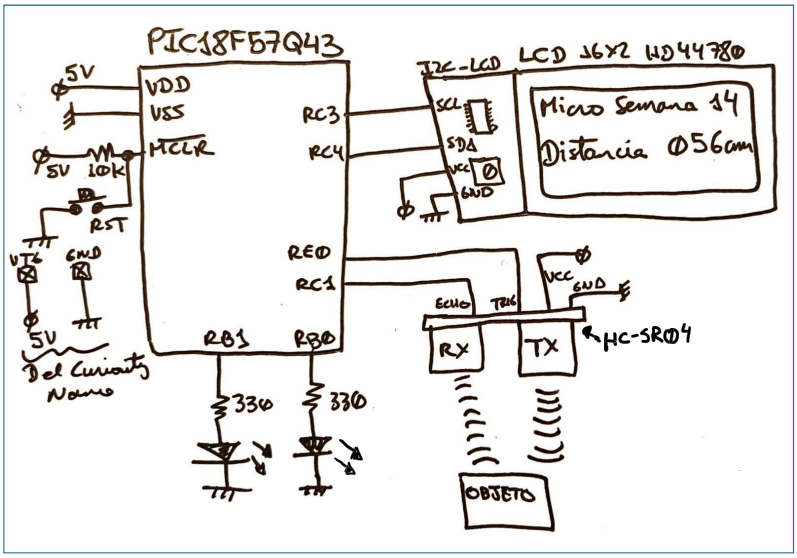
```
unsigned int calculo(unsigned int valor){
    float tiempo;
    unsigned int distancia;
    tiempo= (float)valor;
    tiempo= tiempo / 5.8;
    distancia = (int)tiempo + 1;
    return distancia;
}
```

```
void main(void) {
    configuro();
    //Inicializacion del I2C LCD
    I2C_LCD_INIT();
    I2C_POS_CURSOR(1,0);
    I2C_ESCRIBE_MENSAJE2("Medidor HC-SR04");
    while(1){
        I2C_POS_CURSOR(2,0);
        I2C_ESCRIBE_MENSAJE2("Dist: ");
        I2C_LCD_ESCRIBE_VAR_INT(calculo(Read_HCSR04()),3,1);
        I2C_ESCRIBE_MENSAJE2(" cm");
        __delay_ms(100);
    }
}
```


22

Ejemplo 2024-2

- Desarrollar un medidor de distancia empleando el sensor HC-SR04




```c
#include <xc.h>
#include "cabecera.h"
#include "I2C_LCD.h"
#define _XTAL_FREQ 32000000UL

unsigned int tiempo = 0;

void configuro(void){
    //conf de la fuente de reloj
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;
    OSCEN = 0x40;
    //conf las E/S
    TRISEbits.TRISE0 = 0;   //RE0 salida
    ANSELEbits.ANSELE0 = 0; //RE0 digital
    TRISCbits.TRISC1 = 1;   //RC1 entrada
    ANSELCbits.ANSELC1 = 0; //RC1 digital
    TRISB = 0xFC;   //RB0 y RB1 como salidas
    ANSELB = 0xFC;  //RB0 y RB1 como digitales
    //condiciones iniciales
    LATEbits.LATE0 = 0;
    //inicializacion del I2C_LCD
    I2C_LCD_INIT();
}

void pantallazo(void){
    I2C_POS_CURSOR(1,0);
    I2C_ESCRIBE_MENSAJE2("Medidor de dist.");
    I2C_POS_CURSOR(2,0);
    I2C_ESCRIBE_MENSAJE2("Hecho por Kalun");
    __delay_ms(3000);
    I2C_BORRAR_LCD();
}
```

Ejemplo 2024-2

```c
unsigned int HCSR04_software(void){
    unsigned int tiempo;
    //primero: pulso positivo de 10us en RE0
    LATEbits.LATE0 = 1;
    __delay_us(10);
    LATEbits.LATE0 = 0;
    while(PORTCbits.RC1 == 0);  //esperar a que ECHO sea 1
    tiempo = 0;
    do{
        tiempo++;
        __delay_us(1);
    }
    while(PORTCbits.RC1 == 1);
    return tiempo;
}

void main(void) {
    configuro();
    pantallazo();
    while(1){
        tiempo = HCSR04_software();
        I2C_POS_CURSOR(1,0);
        I2C_ESCRIBE_MENSAJE2("Ultrasonico!");
        I2C_POS_CURSOR(2,0);
        I2C_ESCRIBE_MENSAJE2("Tiempo: ");
        I2C_LCD_ESCRIBE_VAR_INT(tiempo,5,0);
        __delay_ms(1000);
    }
}
```

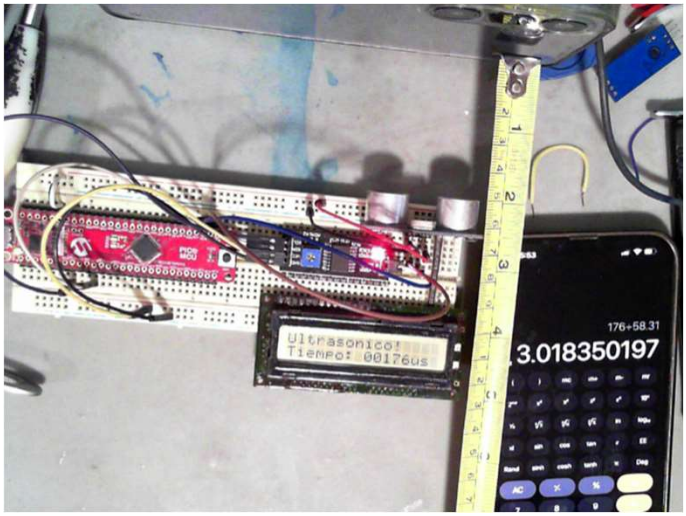
temporizado empleando incremento de variable y retardo de 1 uS

ECHO

medir el tiempo

I2C_ESCRIBE_MENSAJE2("us");

- No sale exacto!
- La base de tiempo no es precisa en la rutina de temporizado empleando do..while
- La base de tiempo es mas de 1uS por el tiempo de ejecución de las instrucciones



25

Ejemplo 2024-2

- Utilizando el Timer1 como base de tiempo

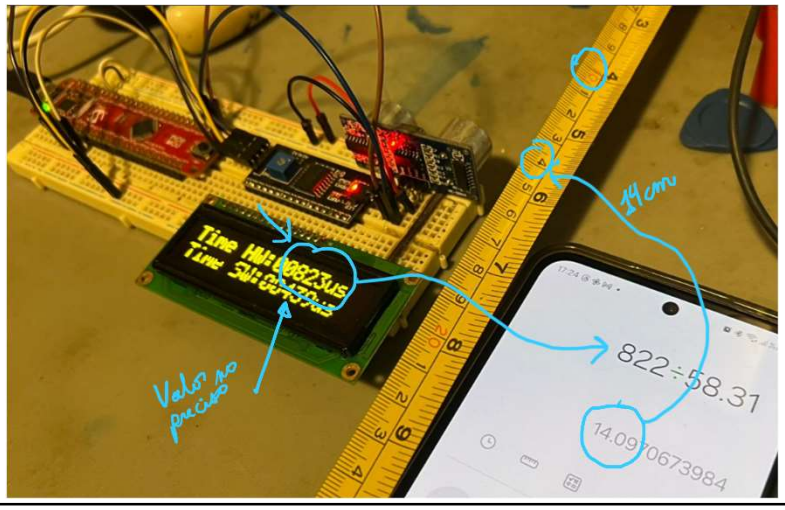
```
#include ...3 lines
#define _XTAL_FREQ 32000000UL

unsigned int tiempo = 0;

void configuro(void){
    //conf de la fuente de reloj
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;
    OSCEN = 0x40;
    //conf las E/S
    TRISEbits.TRISE0 = 0;   //RE0 salida
    ANSELEbits.ANSELE0 = 0; //RE0 digital
    TRISCbits.TRISC1 = 1;   //RC1 entrada
    ANSELCbits.ANSELC1 = 0; //RC1 digital
    TRISB = 0xFC;   //RB0 y RB1 como salidas
    ANSELB = 0xFC;  //RB0 y RB1 como digitales
    //conf del Timer1 para la base de tiempo de 1us
    T1CLK = 0x01;   //Timer1 con fosc/4
    T1CON = 0x32;   //presc 1:8, Timer1 en OFF!
    //condiciones iniciales
    LATEbits.LATE0 = 0;
    //inicializacion del I2C_LCD
    I2C_LCD_INIT();
}

void pantallazo(void){...8 lines }

unsigned int HCSR04_software(void){...18 lines }
```

```
unsigned int HCSR04_hardware(void){
    //rutina empleando el Timer1 como base de tiempo de 1us
    //primero: pulso positivo de 10us en RE0
    LATEbits.LATE0 = 1;
    __delay_us(10);
    LATEbits.LATE0 = 0;
    TMR1H = 0;
    TMR1L = 0;
    while(PORTCbits.RC1 == 0);
    T1CONbits.ON = 1;   //encendemos Timer1 para que cuente
    while(PORTCbits.RC1 == 1);
    T1CONbits.ON = 0;
    return TMR1;
}
return TMR1;
void main(void) {
    configuro();
    pantallazo();
    while(1){
        I2C_POS_CURSOR(1,0);
        I2C_ESCRIBE_MENSAJE2("Time HW:");
        I2C_LCD_ESCRIBE_VAR_INT(HCSR04_hardware(),5,0);
        I2C_ESCRIBE_MENSAJE2("us");
        I2C_POS_CURSOR(2,0);
        I2C_ESCRIBE_MENSAJE2("Time SW:");
        I2C_LCD_ESCRIBE_VAR_INT(HCSR04_software(),5,0);
        I2C_ESCRIBE_MENSAJE2("us");
        __delay_ms(1000);
    }
}
```

26

Ejemplo 2024-2

- Utilizando el valor de tiempo obtenido con el Timer1 y empleando la ecuación para hallar la distancia en una calculadora, se puede obtener la distancia con una calculadora.
- Se puede evidenciar el gran error de temporizado con la rutina do..while



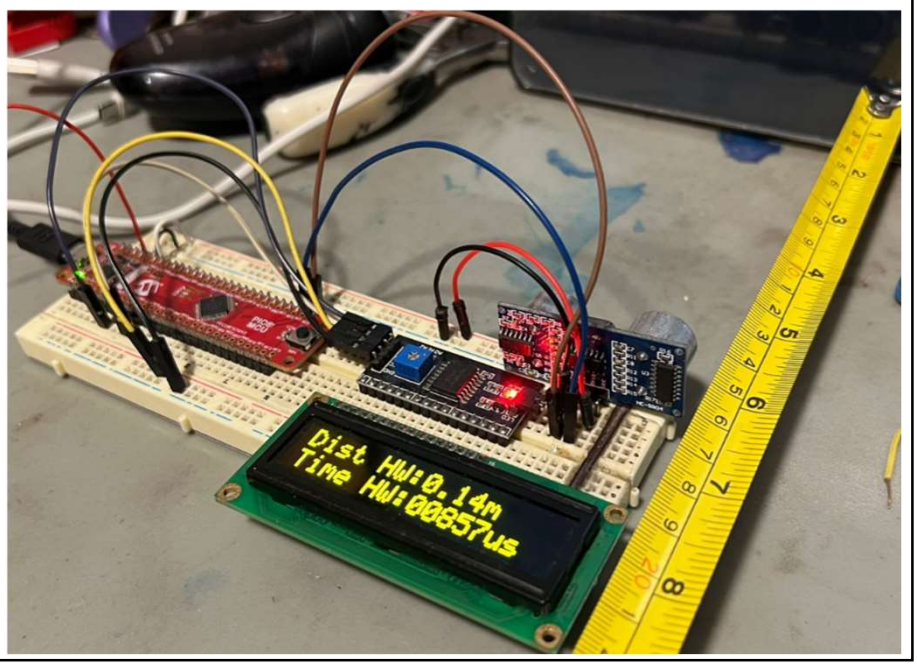
Ejemplo 2024-2

- Convirtiendo a distancia:

```
78  unsigned int calculo(unsigned int valor){
79      float tiempo;
80      unsigned int distancia;
81      tiempo = (float)valor;
82      tiempo = tiempo / 58.31;
83      distancia = (int)tiempo;
84      return distancia;
85  }
86
87  void main(void) {
88      configuro();
89      pantallazo();
90      while(1){
91          I2C_POS_CURSOR(1,0);
92          I2C_ESCRIBE_MENSAJE2("Dist HW:");
93          I2C_LCD_ESCRIBE_VAR_INT(calculo(HCSR04_hardware()),3,0);
94          I2C_ESCRIBE_MENSAJE2("cm");
95          I2C_POS_CURSOR(2,0);
96          I2C_ESCRIBE_MENSAJE2("Dist SW:");
97          I2C_LCD_ESCRIBE_VAR_INT(calculo(HCSR04_software()),3,0);
98          I2C_ESCRIBE_MENSAJE2("cm");
99          __delay_ms(1000);
100     }
101 }
```

Ejemplo 2024-2

- Convirtiendo a distancia:



29

2024-2

- Mejorando resolución hasta el milímetro

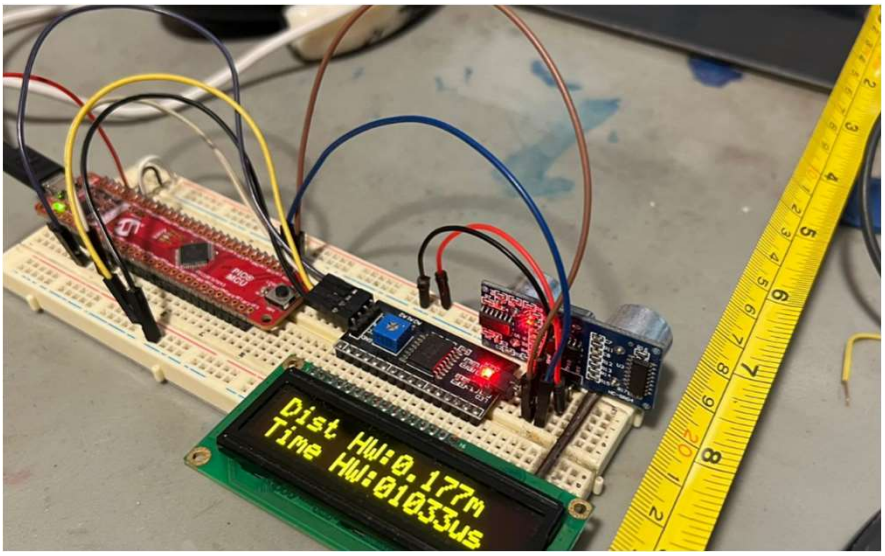
```
#include <xc.h>
#include "cabecera.h"
#include "I2C_LCD.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){...20 lines }

void pantallazo(void){...8 lines }

unsigned int HCSR04_hardware(void){
    //rutina empleando el Timer1 como base de tiempo de lus
    //primero: pulso positivo de 10us en RE0
    LATEbits.LATE0 = 1;
    __delay_us(10);
    LATEbits.LATE0 = 0;
    TMR1 = 0;
    while(PORTCbits.RC1 == 0);
    T1CONbits.ON = 1;   //encendemos Timer1 para que cuente
    while(PORTCbits.RC1 == 1);
    T1CONbits.ON = 0;   //apagamos el Timer1
    return TMR1;
}

unsigned int HCSR04_conversion(unsigned int valor){
    //calculo para obtener la distanca en cm a partir
    //del temporizado obtenido en el TMR1
    float tiempo;
    unsigned int distancia;
    tiempo = (float)valor;
    //tiempo = tiempo / 58.31;    //distancia en cm
    tiempo = tiempo / 5.831;    //distancia en mm
    distancia = (int)tiempo;
    return distancia;
}
```

```
void main(void) {
    configuro();
    pantallazo();
    while(1){
        I2C_POS_CURSOR(1,0);
        I2C_ESCRIBE_MENSAJE2("Dist HW:");
        //I2C_LCD_ESCRIBE_VAR_INT(HCSR04_conversion(HCSR04_hardware()),3,2);
        I2C_LCD_ESCRIBE_VAR_INT(HCSR04_conversion(HCSR04_hardware()),4,3); //
        I2C_ESCRIBE_MENSAJE2("m");
        I2C_POS_CURSOR(2,0);
        I2C_ESCRIBE_MENSAJE2("Time HW:");
        I2C_LCD_ESCRIBE_VAR_INT(HCSR04_hardware(),5,0);
        I2C_ESCRIBE_MENSAJE2("us");
        __delay_ms(1000);
    }
}
```

30

Ejemplo 2024-2

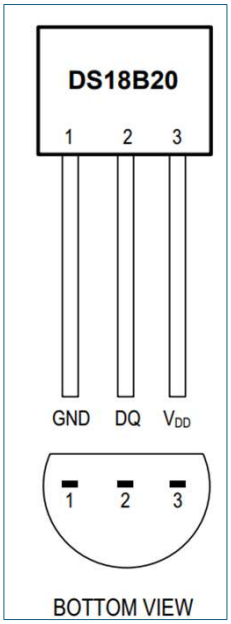
- Mejorando resolución hasta el milímetro



31

El sensor DS18B20

- Creado por Dallas Semiconductor
- Dallas Semiconductor fué absorbida por Maxim Integrated en el año 2002
- Analog Devices absorbe a Maxim Integrated en el año 2021
- Emplea comunicación 1-wire (un solo cable para la transmission de datos en forma bidireccional y también para energía)
- Se pueden conectar diferentes dispositivos 1-wire a microcontrolador empleando un solo cable (topología tipo bus)

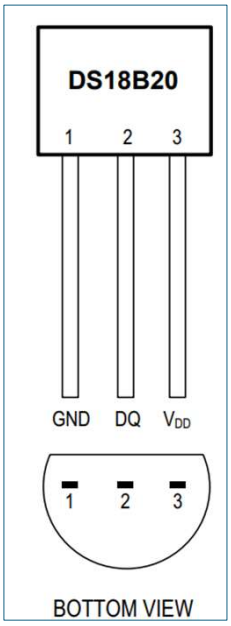


32

El sensor DS18B20

Benefits and Features

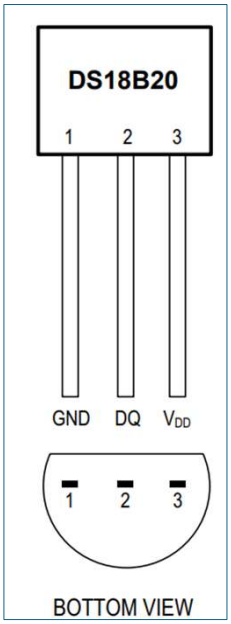
- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages



33

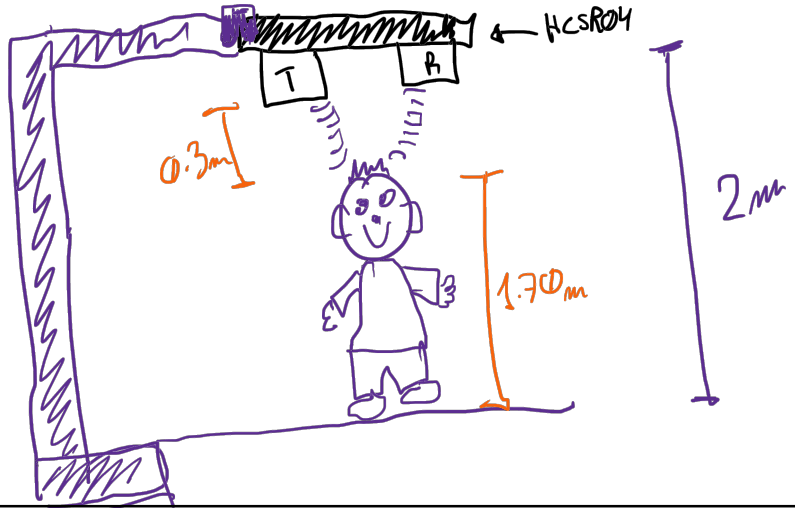
El sensor DS18B20

- El principal inconveniente es que la rutina para poder comunicarnos con el DS18B20 lo tienes que hacer por software, es bien escaso encontrar microcontroladores que Tengan implementado en hardware la comunicación 1-wire.
- En los microcontroladores generalmente puedes encontrar periféricos de comunicación serial solo I2C y UART.
- Afortunadamente hay implementaciones (librerías) que se pueden portar al PIC18F57Q43



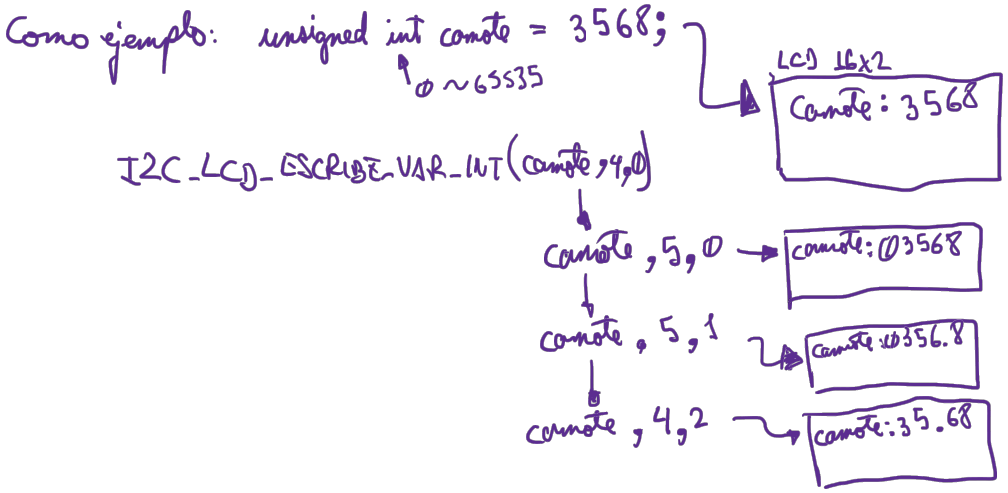
34

¿Cómo hago para hacer un dispositivo que mida la estatura de una persona?



35

¿Cómo funciona el I2C_LCD_ESCRIBE_VAR_INT()?



36

Fin de la sesión

37