Microcontroladores

Interface con un sensor DHT11

1

Preguntas previas:

- Profesor y las carpetas compartidas para el TF?
  - Voy a crearlas en el transcurso del día.
- Profesor el DD es un lab calificado?
  - Si, es el cuarto laboratorio calificado, a administrarse en la semana 15 y en horario de laboratorio.

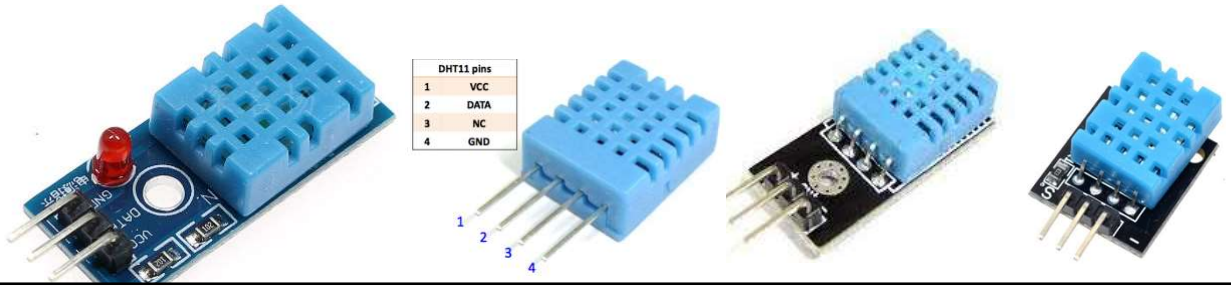
2

Agenda:

- El DHT11
- Comunicación con el DHT11
- Funciones en XC8 para el DHT11
- Creación de librerías en XC8
- Ejemplos de aplicaciones con el DHT11

El DHT11

- Aspectos iniciales:
  - El DHT11 es un sensor de temperatura y humedad con salida de datos digitales (serial).
  - Al revisar la hoja técnica del DHT11 podemos ver que este sensor tiene un rango de voltaje de operación de 3V a 5.5V. Por lo que podremos conectar directamente al microcontrolador PIC18F4550
  - 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:



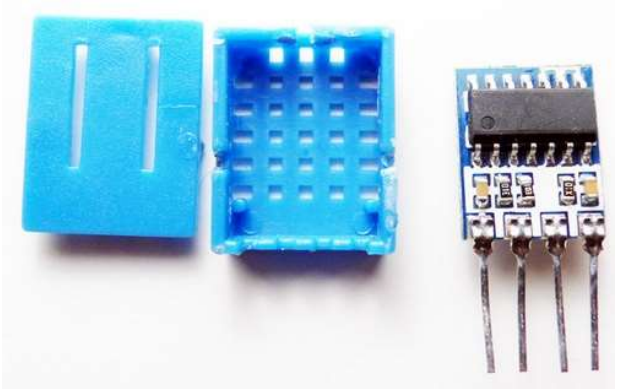
El DHT11

- Aspectos iniciales:
  - 20%-90% con precisión de ± 5% en humedad relativa ambiental
  - 0°C-50°C con precisión de ±2°C en temperatura ambiental
  - Muestreo de 1Hz como máximo (cada segundo se debe de obtener la información de la humedad y temperatura)
  - Consumo máximo de 2.5mA durante la conversión, 150uA en stand-by
  - Hoja técnica: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

5

El DHT11

- Vista interna del DHT11:



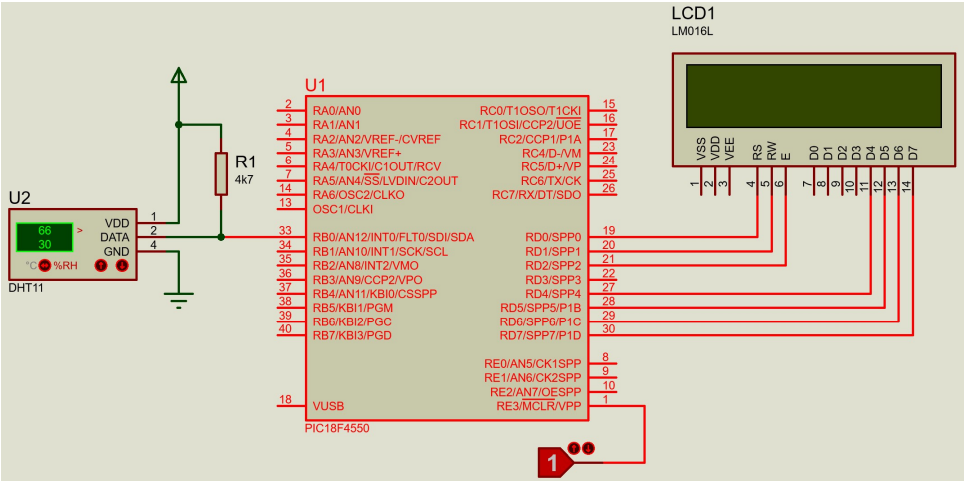
6

El DHT11

- Comunicación con el DHT11
  - Se emplea un solo pin para para la comunicación entre el microcontrolador y el sensor DHT11 (los datos se envían serialmente).
  - El pin de comunicación es bidireccional.
  - El microcontrolador es el que inicia la comunicación
  - El sensor envía 5 datos de 8 bits (total 40 bits) correspondientes a:
    - Humedad relativa parte entera
    - Humedad relativa parte decimal
    - Temperatura parte entera
    - Temperatura parte decimal
    - Checksum
  - Los datos enviados son MSBfirst (bit mas significativo se envía al inicio)
  - Tener en cuenta que el DHT11 solo maneja parte entera en las medidas.
  - El checksum es la suma de las cuatro medidas (Hum. Entera + Hum. Dec. + Temp. Entera + Temp. Dec.) y se emplea para verificar si hay integridad en la información recibida.

El DHT11

- Circuito de prueba: La salida de datos del DHT11 esta conectado a RB0



El DHT11

- Código de prueba para el LCD:
- ¿Qué es lo que se visualiza?

```
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"

#define _XTAL_FREQ 48000000UL

void LCD_Init(void){
    TRISD = 0x00;
    __delay_ms(50);
    LCD_CONFIG();
    __delay_ms(15);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void main(void) {
    LCD_Init();
    ESCRIBE_MENSAJE("Prueba", 6);
    unsigned char x=0;
    for(x=0;x<10;x++){
        ENVIA_CHAR('.');
        __delay_ms(200);
    }
    POS_CURSOR(2,1);
    ESCRIBE_MENSAJE("Todo OK!", 8);
    while(1);
}
```


El DHT11 – Funciones para el XC8

- Para el desarrollo de las presentes funciones se ha contemplado que el DHT11 se encuentra conectado al pin RB0
- Se han creado tres funciones para la comunicación:
  - DHT11_Start() – El microcontrolador le envía una señal de inicio al DHT11
  - DHT11_Check() – El microcontrolador espera respuesta del DHT11
  - DHT11_Read() – El microcontrolador recibe un dato de 8 bits
- Se tiene que realizar 5 veces el DHT11_Read() para obtener los 40 bits que envía el DHT11

El DHT11 – Función DHT11_Start()

```
void DHT11_Start(void){
    TRISBbits.RB0 = 0;        //Puerto RB0 como salida
    LATBbits.LB0 = 0;         //Mandamos cero
    __delay_ms(18);           //Esperamos 18 milisegundos
    LATBbits.LB0 = 1;         //Mandamos uno
    __delay_us(20);           //Esperamos 20 microsegundos
    TRISBbits.RB0 = 1;        //Puerto como entrada
}
```


El DHT11 – Función DHT11_Check()

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

El DHT11 – Función DHT11_Read()

```
unsigned char DHT11_Read(void){
    unsigned char x = 0, data = 0;
    for(x=0;x<8;x++){
        while(!PORTBbits.RB0);
        __delay_us(30);
        if(PORTBbits.RB0){
            data = ((data<<1) | 1);
        }
        else{
            data = (data<<1);
        }
        while(PORTBbits.RB0);
    }
    return data;
}
```

13

El DHT11: Código fuente completo

```
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"

#define _XTAL_FREQ 48000000UL

unsigned char RH_Entera, RH_Decimal, Temp_Entera, Temp_Decimal, Checksum;
unsigned char centena=0, decena=0, unidad=0;

void DHT11_Start(){
    TRISBbits.RB0 = 0;      //Puerto RB0 como salida
    LATBbits.LB0 = 0;       //Mandamos cero
    __delay_ms(18);         //Esperamos 18 milisegu
    LATBbits.LB0 = 1;       //Mandamos uno
    __delay_us(20);         //Esperamos 20 microseg
    TRISBbits.RB0 = 1;      //Puerto como entrada
}

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

unsigned char DHT11_Read(){
    unsigned char x = 0, data = 0;
    for(x=0;x<8;x++){
        while(!PORTBbits.RB0);
        __delay_us(30);
        if(PORTBbits.RB0){
            data = ((data<<1) | 1);
        }
        else{
            data = (data<<1);
        }
        while(PORTBbits.RB0);
    }
    return data;
}
```

```
void LCD_Init(void){
    TRISD = 0x00;
    __delay_ms(50);
    LCD_CONFIG();
    __delay_ms(15);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void convierte(unsigned char numero){
    centena = (numero % 1000) / 100;
    decena = (numero % 100) / 10;
    unidad = numero % 10;
}
```

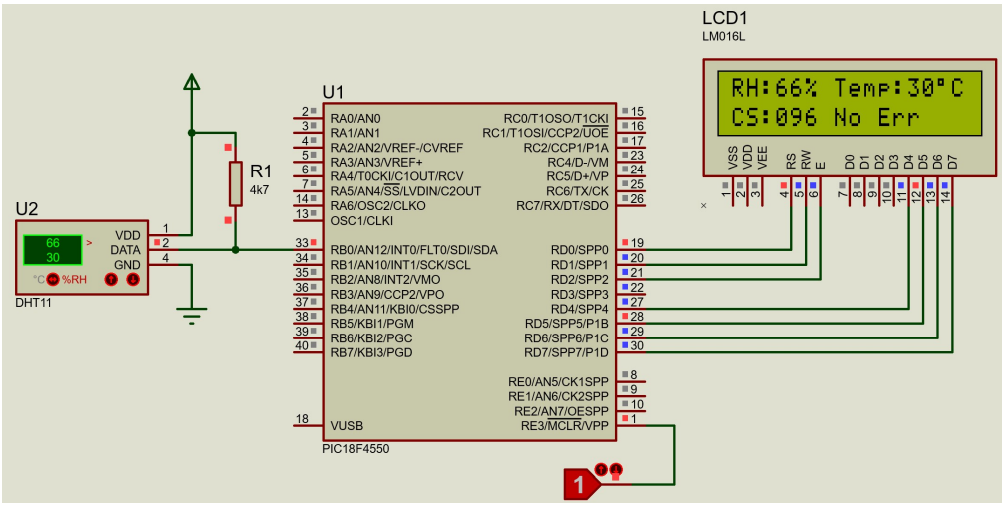
```
void main(void) {
    LCD_Init();
    ESCRIBE_MENSAJE("Prueba", 6);
    unsigned char x=0;
    for(x=0;x<10;x++){
        ENVIA_CHAR('.');
        __delay_ms(200);
    }
    POS_CURSOR(2,1);
    ESCRIBE_MENSAJE("Todo OK!", 8);
    __delay_ms(3000);
    BORRAR_LCD();
    while(1){
        DHT11_Start();
        DHT11_Check();

        RH_Entera = DHT11_Read();
        RH_Decimal = DHT11_Read();
        Temp_Entera = DHT11_Read();
        Temp_Decimal = DHT11_Read();
        Checksum = DHT11_Read();
        CURSOR_HOME();
        ESCRIBE_MENSAJE("RH:",3);
        convierte(RH_Entera);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
        ESCRIBE_MENSAJE("% Temp:",7);
        convierte(Temp_Entera);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
        ENVIA_CHAR(0xDF);
        ENVIA_CHAR('C');
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE("CS:",3);
        convierte(Checksum);
        ENVIA_CHAR(centena+0x30);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
        ENVIA_CHAR(' ');
        if(Checksum != (RH_Entera+RH_Decimal+Temp_Entera+Temp_Decimal)){
            BORRAR_LCD();
            ESCRIBE_MENSAJE("Hay Error!!",11);
        }
        else{
            ESCRIBE_MENSAJE("No Err",6);
        }
        __delay_ms(1000);
    }
}
```

14

El DHT11

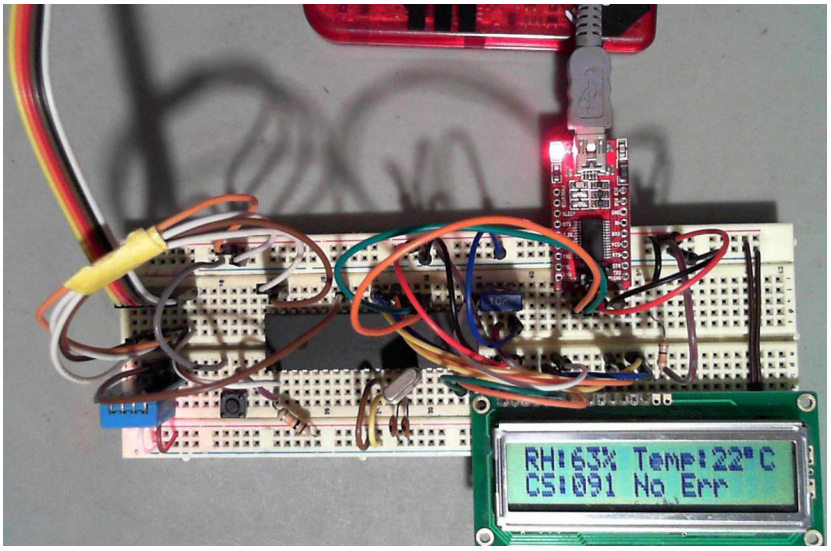
- Simulación en Proteus



15

El DHT11

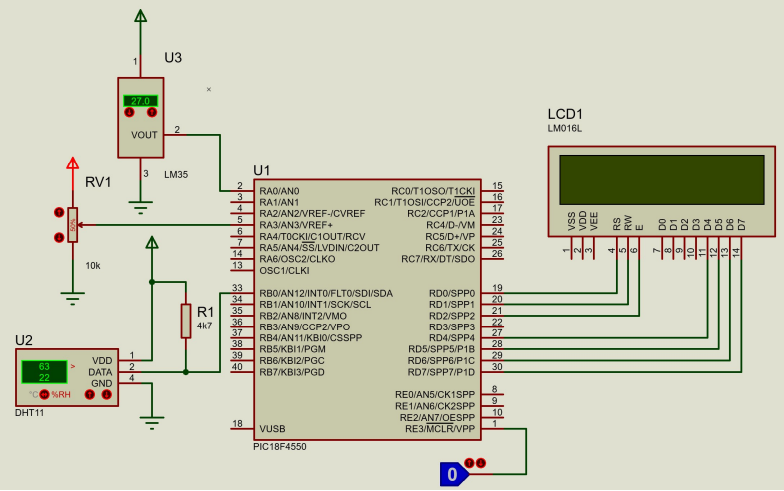
- Circuito implementado:



16

Ejemplo:

- Desarrollar un circuito que muestre las medidas de un DHT11 y de un LM35DZ en un display 16x2



(cont...)

- Librería dht11_lib

dht11_lib.h

```
#ifndef XC_HEADER_TEMPLATE_H
#define XC_HEADER_TEMPLATE_H

#include <xc.h> // include processor

#define _XTAL_FREQ 48000000UL

void DHT11_Start(void);
void DHT11_Check(void);
unsigned char DHT11_Read(void);

#endif  /* XC_HEADER_TEMPLATE_H */
```

dht11_lib.c

```
#include "dht11_lib.h"

void DHT11_Start(void){
    TRISBbits.RB0 = 0;      //Puerto RB0 como salida
    LATBbits.LB0 = 0;       //Mandamos cero
    __delay_ms(18);         //Esperamos 18 milisegundos
    LATBbits.LB0 = 1;       //Mandamos uno
    __delay_us(20);         //Esperamos 20 microsegundos
    TRISBbits.RB0 = 1;      //Puerto como entrada
}

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

unsigned char DHT11_Read(void){
    unsigned char x = 0, data = 0;
    for(x=0;x<8;x++){
        while(!PORTBbits.RB0);
        __delay_us(30);
        if(PORTBbits.RB0){
            data = ((data<<1) | 1);
        }
        else{
            data = (data<<1);
        }
        while(PORTBbits.RB0);
    }
    return data;
}
```

(cont…)

- Código principal

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#include "dht11_lib.h"

#define _XTAL_FREQ 48000000UL

void ADC_init(void){
    ADCON2 = 0xA4;      //Tiempo de conversion
    ADCON1 = 0x1B;      //Seleccion de los canales analogicos
    ADCON0 = 0x01;      //Encender el modulo A/D
}

void LCD_init(void){
    TRISD = 0x00;
    __delay_ms(50);
    LCD_CONFIG();
    __delay_ms(15);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

float LM35_read(void){
    unsigned int lm35raw;      //variable de funcion LM35_read
    float n_temp_c;
    ADCON0bits.GODONE = 1;      //toma de una muestra en AN0
    while(ADCON0bits.GODONE == 1);
    lm35raw = (ADRESH << 8) + ADRESL; //ADRESH:ADRESL
    n_temp_c = lm35raw / 10.24;
    return n_temp_c;
}
```

```c
void ESCRIBE_VARIABLECHAR_LCD(unsigned char numero){
    unsigned char centena, decena, unidad;  //variables de funcion convierte
    centena = (numero % 1000) / 100;
    decena =  (numero % 100) / 10;
    unidad = numero % 10;
//    ENVIA_CHAR(centena+0x30);
    ENVIA_CHAR(decena+0x30);
    ENVIA_CHAR(unidad+0x30);
}

void main(void) {
    unsigned char RH_Ent, RH_Dec, Temp_Ent, Temp_Dec, Chksum;
    ADC_init();
    LCD_init();
    while(1){
        CURSOR_HOME();
        ESCRIBE_MENSAJE("T1:",3);
        ESCRIBE_VARIABLECHAR_LCD(LM35_read());
        ENVIA_CHAR(0xDF);
        ESCRIBE_MENSAJE("C ", 2);
        DHT11_Start();
        DHT11_Check();
        RH_Ent = DHT11_Read();
        RH_Dec = DHT11_Read();
        Temp_Ent = DHT11_Read();
        Temp_Dec = DHT11_Read();
        Chksum = DHT11_Read();
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE("T2:",3);
        ESCRIBE_VARIABLECHAR_LCD(Temp_Ent);
        ENVIA_CHAR(0xDF);
        ESCRIBE_MENSAJE("C RH:", 5);
        ESCRIBE_VARIABLECHAR_LCD(RH_Ent);
        ENVIA_CHAR('%');
        __delay_ms(1000);
    }
}
```


(cont…)

- Función **LM35_read()** se encarga de realizar una lectura al sensor LM35 y hacer el escalamiento y conversiones necesarias para obtener una variable con rango 0-100°C
- Función **ESCRIBE_VARIABLECHAR_LCD(unsigned char numero)** se encarga de individualizar los dígitos centena, decena y unidad de la variable que se recibe (en este caso la variable numero), formatearlos según ROM de caracteres del 44780 (en este caso sumarle 0x30) e imprimirlos en el display.

(cont…)

- Simulación: