SENSOR DE TEMPERATURA

Raquel Roca Pereira 55431
Ines Rico Peng 55426
Iria Touriño Villanueva 55493

ÍNDICE

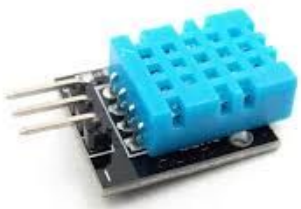
1. DESCRIPCIÓN

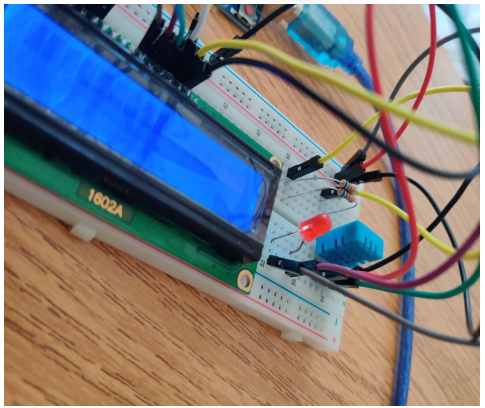
Sistema para controlar la temperatura en un espacio cerrado. Esto nos sirve para evitar que en nuestro lugar de trabajo alcancemos una temperatura que no sea adecuada para desempeñar nuestras tareas. El sensor detecta la temperatura cada cierto tiempo y mediante un código de leds y a través de una pantalla recibiremos la información, donde además será almacenada.

2. MATERIAL

Sensor de temperatura DHT11

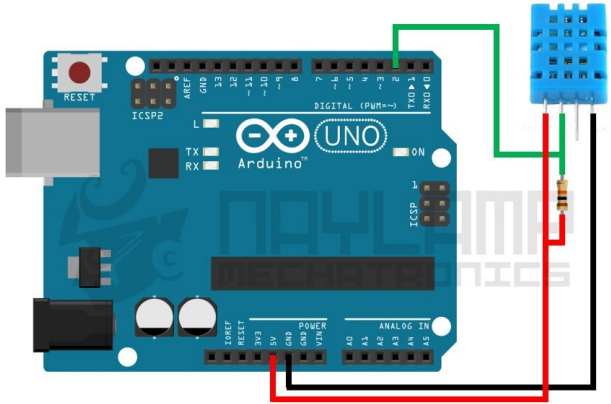
Este sensor trabaja con un rango de medición de temperatura de 0 a 50 °C con precisión de ±2.0 °C y un rango de humedad de 20% a 90% RH con precisión de 4% RH. Los ciclos de lectura debe ser como mínimo 1 o 2 segundos.





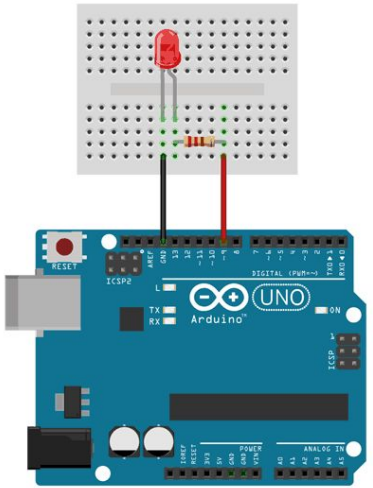
-Código:

```
#include "DHT.h"

#define DHTPIN 2      // Pin donde está conectado el sensor

//#define DHTTYPE DHT11   // Descomentar si se usa el DHT 11
#define DHTTYPE DHT22    // Sensor DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("Iniciando...");
  dht.begin();
}
void loop() {
  delay(2000);
  float h = dht.readHumidity();  //Leemos la Humedad
  float t = dht.readTemperature(); //Leemos la temperatura en grados Celsius
    float  f  =  dht.readTemperature(true);  //Leemos  la  temperatura  en  grados
Fahrenheit
  //--------Enviamos las lecturas por el puerto serial-------------
  Serial.print("Humedad ");
  Serial.print(h);
  Serial.print(" %t");
  Serial.print("Temperatura: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(f);
  Serial.println(" *F");
}
```



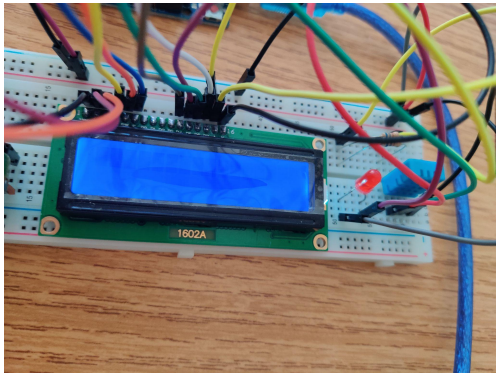
Luces LED

- Para indicar si la temperatura está entre los rangos indicados utilizaremos luces LED.
- Diagrama de conexiones: con el arduino.



-Código:

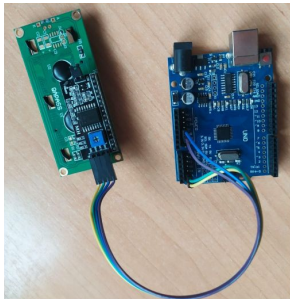
```
const int LED=13;
void setup()
{
pinMode(LED,OUTPUT);
}
void loop()
{
digitalWrite(LED,HIGH);
delay(1000);
digitalWrite(LED,LOW);
delay(1000);
}
```

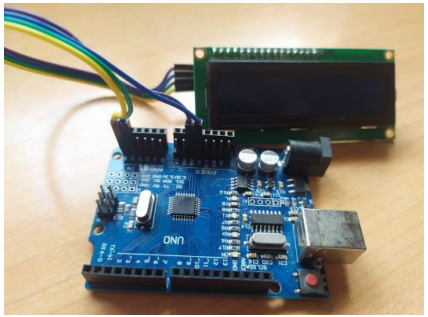


Pantalla LCD 16x2

- Para enseñar los datos a través del arduino, se va a usar un display de 16 carácteres alfanuméricos por 2 columnas, la LCD HD44780, como esta pantalla tiene una configuración de pines bastante compleja, decimos usar el adaptador de pantalla I2C PCF8574, quedándose las conexiones reducidas a tierra (GND), fuente de voltaje (Vcc 5V), y dos pines analógicos (A4 y A5).

- Para poder usar este adaptador ha sido necesaria descargarse una librería que será adjuntada a Github. El código para poner la pantalla en marcha es el siguiente:





```
 void setup()
 {
 lcd.begin(16, 2);
lcd.clear();
 lcd.print("Temperatura=");
sensor=analogRead(A0); //indica el sensor de donde se lee
lcd.setCursor(0,1);
lcd.print(“temperatura”);
 delay(1000);
 }
void loop()
{
// en función de las mediciones de los sensores la pantalla tendrá determinadas respuestas
 }
```

Placa ARDUINO UNO R3

Arduino es una placa basada en un microcontrolador ATMEL con circuitos integrados donde se pueden grabar instrucciones programadas en el entorno Arduino IDE. Este microcontrolador posee tanto una interfaz de entrada como de salida, donde podemos conectar distintos tipos de periféricos. Por ejemplo, en nuestro proyecto conectaremos a la interfaz de entrada un sensor de temperatura DS18B20 y a la interfaz de salida tanto la pantalla LCD como las luces leds que transmitirán al exterior la información detectada por el sensor.

La placa Arduino UNO R3 tiene 14 pines digitales de entrada y salida, entre los cuales 6 se pueden usar como salidas PWM y otros 6 como entradas analógicas. Esta placa se diferencia del resto en que cuenta con el Atmega8U2 programado como convertidor de USB a serie. Arduino posee convertidores ADC (analógico a digital) que nos permiten realizar lecturas analógicas del mundo exterior, imprescindibles en la inmensa mayoría de los proyectos.

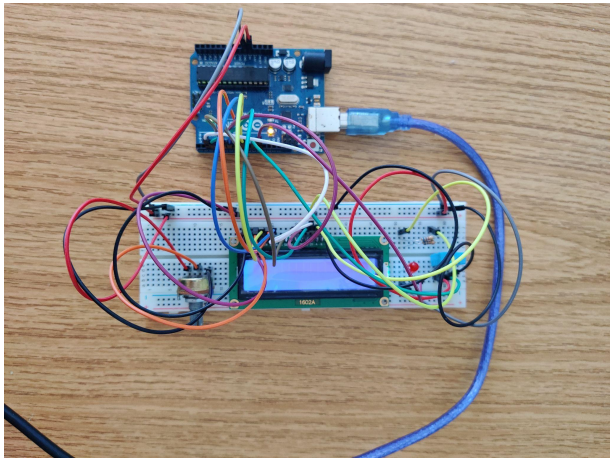
El problema de los Arduino es que tienen una precisión y una estabilidad muy justa. Los ADC internos de Arduino no tienen una mala resolución, el problema es que en algunos proyectos pueden ser un tanto imprecisos.

Las placas de Arduino UNO poseen 10 bits, es decir, 2 10=1024 valores de resolución y si comparamos estos con los 5V que nos suministra Arduino UNO, nos encontramos con la siguiente resolución:

5 V/1024= 4.8 mV de resolución.

Esta precisión puede ser más que suficiente en la mayoría de los proyectos simples, aunque en otros se queda corta.

Otro problema muy común es que el Arduino UNO no está bien aislado de interferencias que pueden existir a la hora de realizar la conversión y son demasiado sensibles al ruido externo.



3.CÓDIGO VISUAL STUDIO

- Con este programa realizamos la comunicación principal entre el proyecto y el usuario.
- Es la parte encargada de recibir qué acción se quiere realizar y comunicarle dicha elección a arduino mediante determinadas funciones.
- Imprime por pantalla el menú y recibe por teclado la opción seleccionada.
- Manda y recibe mensaje a Arduino mediante cadenas de caracteres en formato String.
- Guarda las temperaturas devueltas por Arduino en una estructura ‘registro’ y tiene la opción de generar un fichero con esta información para que se quede guardada en la memoria del ordenador.
- Se encarga de mandar el mensaje a Arduino para que active/desactive la alarma.

```c
#include <stdio.h>                          //Incluimos las bibliotecas necesarias para el correcto funcionamiento del programa
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include <conio.h>
#include "SerialClass/SerialClass.h"
#define MAX_BUFFER 200                      //Definimos el varios valores necesarios
#define PAUSA_MS 200
#define LONGCAD 50
#define N 100

typedef struct                             //Creamos una estructura de tipo registro donde
{                                          //guardaremos un vector de temperaturas correspondiente
    int dia;                               //a una fecha y una hora caracter♦sticas
    int mes;
    int anio;
    int hora;
    int minutos;
    float temperatura[N];
}registro;
//Declaramos las distintas funciones a las que iremos llamando a lo largo de todo el programa
int menu(void);                                          //Imprime las opciones y devuelve la opcion escogida
void verifica_sensores(Serial*, char*);                  //Lee una temperatura para comprovar el funcionamiento del sensor
registro monitorizar_sensor_temperatura(Serial*, int*); //Realiza una series de lecturas de t durante cierto tiempo
void activar_alarma_temperatura(Serial* Arduino);        //Envia la se♦al para activar o desactivar la alarma(led)
void comprobar_mensajes(Serial*);                        //Comprueba los mensajes recibidos desde Arduino
float leer_sensor_temperatura(Serial*);                  //Envia la se♦al para leer una temperatura y la devuelve a la variable
int Enviar_y_Recibir(Serial*, const char*, char*);       //Funcion principal para la comunicacion entre visual y arduino
float float_from_cadena(char* cadena);                   //Transforma cadenas de caracteres en formato float
void registrar_temperatura(registro*, int);              //Guarda en el vector registro las temperaturas captadas
void escribir_fichero_registro(registro* r, int nr, int nt);//Genera un fichero con los registros generados
void cambiar_t_crit(Serial* Arduino);
```

```c
int main(void)      //FUNCION PRINCIPAL
{
    Serial* Arduino;                //Variable de comunicacion com arduino
    char puerto[] = "COM3";         //Puerto serie al que est conectado Arduino
    int opcion_menu, nr = 0, nt = 0; //Variables: opcion escogida, numero de registros, numero de temperaturas
    registro r;                     //Variable de tipo registro


    setlocale(LC_ALL, "es-ES");          //Funcion para que no haya errores en la impresion de caracteres
    Arduino = new Serial((char*)puerto); //Conexion con arduino en el puerto indicado
    do   //Bucle para que el programa funcione hasta que se marque 5 en el menu
    {
        comprobar_mensajes(Arduino);    //Comprobar mensajes recibidos de arduino
        opcion_menu = menu();           //llamada a la funcion menu para guardar la opcion escogida por el usuario
        switch (opcion_menu)            //Llamada a la funcion correspondiente en cada opcion
        {
        case 0: break;                          //0->Vuelve a pedir otra opcion
        case 1:
            verifica_sensores(Arduino, puerto); //1->Hace una lectura del sensor para comprobar su correcto funcionamiento
            break;
        case 2:
            r = monitorizar_sensor_temperatura(Arduino, &nt); //2->hace varias lecturas del sensor y lo guarda en la estructuta registro
            nr++;                                             //Ademas cuenta el numero de registros realizados
            break;
        case 3:
            activar_alarma_temperatura(Arduino);   //3->Activa o desactiva la alarma en funcion de su estado anterior
            break;
        case 4:
            escribir_fichero_registro(&r, nr, nt); //4->Genera un fichero con las temperaturas captadas en el caso dos guardadas en registro
            break;
        case 5:
            cambiar_t_crit(Arduino);               //5->Cambia la t crititca
            break;
        case 6:                                    //6->Cierra el programa
            break;
        default: printf("\nOpci\x82n incorrecta\n");  //CUalquier otro numero imprime el mensaje anterior
        }
    } while (opcion_menu != 6);
    return 0;
}
```

```c
int menu(void)                                          //Esta funcion imprime el menu de opciones por pantalla y devuelve
{                                                       //a la funcion principal la escogida por el usuario
    static int opcion = -1;

    if (opcion != 0)
    {
        printf("\n");
        printf("Menú Principal\n");
        printf("==============\n");
        printf("1 - Verificar sensores.\n");
        printf("2 - Monitorizar sensores.\n");
        printf("3 - Activar/Desactivar alarma por distancia\n");
        printf("4 - Generar fichero de temperaturas\n");
        printf("5 - Cambiar temperatura crítica\n");
        printf("6 - Salir de la aplicación\n");
        printf("Opción:");
    }

    if (_kbhit())
    {
        opcion = (int)_getch() - '0';
        printf("%d\n", opcion);
    }
    else
        opcion = 0;

    return opcion;
}
```

```c
void verifica_sensores(Serial* Arduino, char* port)                         //funcion case 1
{
    float temperatura;                                                      //inicializa variable donde se guarda la temp. leida

    if (Arduino->IsConnected())                                             //Comprueba que arduino esta conectado
    {
        temperatura = leer_sensor_temperatura(Arduino);                     //guarda en la variable la temperatura devuelta por la funcion
        if (temperatura != -1)                                              //Si es distinto de -1 imprime por pant5alla la temperatura
            printf("\nTemperatura: %f\n", temperatura);
    }
    else                                                                    //En el caso de que sea -1 imprime un error de conexion
    {
        printf("\nNo se ha podido conectar con Arduino.\n");
        printf("Revise la conexión, el puerto %s y desactive el monitor serie del IDE de Arduino.\n", port);
    }
}


float leer_sensor_temperatura(Serial* Arduino)          //funcion encargada de mandar la señal para que el sensor lea una temperatura
{
    float temperatura;                  //Declaracion de variables
    int bytesRecibidos;
    char mensaje_recibido[MAX_BUFFER];
                                                                                //Almacena en la variable bytesRecibidos el espacio que
    bytesRecibidos = Enviar_y_Recibir(Arduino, "GET_TEMPERATURA\n", mensaje_recibido); //ocupa el mensaje devuelto desde arduino, el cual se alamcena en
                                                                                //la variable mensaje_recibido. Tambien envia a arduino un mensaje
    if (bytesRecibidos <= 0)                                //En este caso el programa no ha recibido mensaje de respuesta de arduino
    {
        printf("\nNo se ha recibido respuesta a la petición\n");
        temperatura = -1;
    }
    else                                    //en el caso que si reciba una respuesta de arduino imprime por patalla el mensaje y lo que ocupa
    {
        printf("\nLa respuesta recibida tiene %d bytes. Recibido=%s\n", bytesRecibidos, mensaje_recibido);
        temperatura = float_from_cadena(mensaje_recibido);  //Ademas adjudica a la variable temperatura el valor que entrega arduino pero antes
    }                                                       //lo pasa por una funcion que lo transforma de tipo string a float
    return temperatura;    //Devuelve la temperatura
}
```

```c
float float_from_cadena(char* cadena)               //Funcion encargada de pasar un elemento de tipo char* a float
{
    float numero = 0;                 //Declaracion de variables
    int i, divisor = 10, estado = 0;


    for (i = 0; cadena[i] != '\0' && estado != 3 && i < MAX_BUFFER; i++)
        switch (estado)
        {
        case 0:// Antes del número
            if (cadena[i] >= '0' && cadena[i] <= '9')
            {
                numero = cadena[i] - '0';
                estado = 1;
            }
            break;
        case 1:// Durante el número
            if (cadena[i] >= '0' && cadena[i] <= '9')
                numero = numero * 10 + cadena[i] - '0';
            else
                if (cadena[i] == '.' || cadena[i] == ',')
                    estado = 2;
                else
                    estado = 3;
            break;
        case 2: // Parte decimal
            if (cadena[i] >= '0' && cadena[i] <= '9')
            {
                numero = numero + (float)(cadena[i] - '0') / divisor;
                divisor *= 10;
            }
            else
                estado = 3;
            break;
        }
    return numero;
}
```

```
int Enviar_y_Recibir(Serial* Arduino, const char* mensaje_enviar, char* mensaje_recibir)      //Funcion encargada de intercambiar mensajes entre visual
{                                                                                             //y arduino
    int bytes_recibidos = 0, total = 0;     //Declaracion de variables
    int intentos = 0, fin_linea = 0;


    Arduino->WriteData((char*)mensaje_enviar, strlen(mensaje_enviar));  //Funcion que manda a arduino un mensaje
    Sleep(PAUSA_MS);                                                    //espera x tiempo

    bytes_recibidos = Arduino->ReadData(mensaje_recibir, sizeof(char) * MAX_BUFFER - 1); //Recibe un mensaje de arduino

    while ((bytes_recibidos > 0 || intentos < 5) && fin_linea == 0)     //lee el mensaje mandado por arduino hasta llegar al final
    {
        if (bytes_recibidos > 0)
        {
            total += bytes_recibidos;
            if (mensaje_recibir[total - 1] == 13 || mensaje_recibir[total - 1] == 10)
                fin_linea = 1;
        }
        else
            intentos++;
        Sleep(PAUSA_MS);
        bytes_recibidos = Arduino->ReadData(mensaje_recibir + total, sizeof(char) * MAX_BUFFER - 1);
    }
    if (total > 0)
        mensaje_recibir[total - 1] = '\0';

    return total; //Devuelve el espacio que ocupa el mensaje recibido
}
```

```
registro monitorizar_sensor_temperatura(Serial* Arduino, int* nt)       //Funcion case 2:
{
    float frecuencia, temperatura;                                      //Declaracion de variables
    //int i=0;
    char tecla;
    registro r;
    int i = 0;
    do
    {
        printf("Establezca frecuencia de muestreo (0,5 Hz - 2,0 Hz):");//Pide al usuario la frecuencia de muestreo de las temperaturas
        scanf_s("%f", &frecuencia);
    } while (frecuencia < 0.5 || frecuencia>2.0);

    printf("Pulse una tecla para finalizar la monitorización\n");     //lee temperaturas desde el sensor hasta que el usuario pulse una tecla cualquiera
    registrar_temperatura(&r, i);                                       //llamada a la funcion que inicializa el elemento registro
    do                                          //Bucle que lee temperaturas desde el sensor hasta pulsar tecla
    {
        if (Arduino->IsConnected())             //Comprueba que arduino esta conectado
        {
            temperatura = leer_sensor_temperatura(Arduino);  //Guarda en la variable la temperatura que la funcion manda a arduino leer
            if (temperatura != -1)
            {
                printf("%.2f ", temperatura);          //Imprime las temperaturas por pantalla
                r.temperatura[i] = temperatura;        //Guarda las temperaturas en el vector de la estructura registro
                i++;                                   //aumenta en 1 el numero de temperaturas leidas
                *nt = i;
            }

            else
                printf("XXX ");             //Si no recibe una temperatura imprme XXX por pantalla
        }
        else
            printf("\nNo se ha podido conectar con Arduino.\n");   //Imprime error de conexion
        if ((1 / frecuencia) * 1000 > PAUSA_MS)
            Sleep((1 / frecuencia) * 1000 - PAUSA_MS);
    } while (_kbhit() == 0);
    tecla = _getch();
    return r;                       //devuelve el elemento registro a la funcion principal
}
```

```c
void activar_alarma_temperatura(Serial* Arduino)            //Funcion manda mensaje a arduino para activar/desactivar alarma
{
    int bytesRecibidos;
    char mensaje_recibido[MAX_BUFFER];

    bytesRecibidos = Enviar_y_Recibir(Arduino, "SET_MODO_ALARMA\n", mensaje_recibido); //Funcion de comunicacion entre programas
    if (bytesRecibidos <= 0)
        printf("\nNo se ha recibido confirmación\n");   //Imprime error
    else
        printf("\n%s\n", mensaje_recibido);            //Imprime el mensaje recibido
}


void comprobar_mensajes(Serial* Arduino)        //Funcion que lee los mensaje mandado de arduino a visual
{
    int bytesRecibidos, total = 0;
    char mensaje_recibido[MAX_BUFFER];

    bytesRecibidos = Arduino->ReadData(mensaje_recibido, sizeof(char) * MAX_BUFFER - 1);
    while (bytesRecibidos > 0)
    {
        Sleep(PAUSA_MS);
        total += bytesRecibidos;
        bytesRecibidos = Arduino->ReadData(mensaje_recibido + total, sizeof(char) * MAX_BUFFER - 1);
    }
    if (total > 0)
    {
        mensaje_recibido[total - 1] = '\0';
        printf("\nMensaje recibido: %s\n", mensaje_recibido);   //Imprime por pantalla el mensaje recibido
    }
}
```

```c
void cambiar_t_crit(Serial* Arduino)                                    ////////2.3////////
{
    int bytesRecibidos, i = 0;
    char t_crit[MAX_BUFFER];
    char mensaje_recibido_t[MAX_BUFFER];

    bytesRecibidos = Enviar_y_Recibir(Arduino, "CAMBIAR_T_CRIT\n", mensaje_recibido_t);
    if (bytesRecibidos <= 0)
        printf("\nNo se ha recibido confirmación\n");
    else
    {
        printf("\n%s\n", mensaje_recibido_t);
        scanf_s("%c", t_crit[i]);
        while (t_crit[i] != '\n')
        {
            i++;
            scanf_s("%c", t_crit[i]);
        }
        //scanf_s("%s\n", t_crit[MAX_BUFFER]);
        //gets_s(t_crit,MAX_BUFFER);
        bytesRecibidos = Enviar_y_Recibir(Arduino, t_crit, mensaje_recibido_t);
        //Arduino->WriteData((float)t_crit, sizeof(float)*1);

        if (bytesRecibidos <= 0)
            printf("\nNo se ha recibido confirmación\n");
        else
            printf("\n%s\n", mensaje_recibido_t);
    }
}
```

```c
void escribir_fichero_registro(registro* r, int nr, int nt)   //Funcion que genera un fichero con la informacion guardada
{                                                             //en el elemento registro
    int i, j;
    FILE* pf;
    errno_t e;
    e = fopen_s(&pf, "Temperatura.txt", "wt");    //Abrimos el fichero temperatura.txt en modo escritura
    if (e == 0)                                   //Si el fichero se ha podido crear
    {
        fprintf(pf, "Numero de registros: %d\n", nr);                          //Escribe en el fichero el nº de registros realizado
        fprintf(pf, "Numero de temperaturas en registro %d: %d\n", nr + 1, nt); //Imprime el numero de temperaturas captadas
        for (i = 0; i < nr; i++)
        {
            fprintf(pf, "%d /", (r + i)->dia);          //Se imprimen el resto de datos que identifican las tmperaturas leidas en una fecha determinada

            fprintf(pf, "%d /", (r + i)->mes);
            fprintf(pf, "%d\n", (r + i)->anio);
            fprintf(pf, "%d :", (r + i)->hora);
            fprintf(pf, "%d\n", (r + i)->minutos);
            for (j = 0; j < nt; j++)
            {
                fprintf(pf, "%.3f\n", (r + i)->temperatura[j]);  //Imprime en el fichero las temperaturas leidas
            }
        }
        fclose(pf);             //Se cierra el fichero
    }
    else
        printf("Se ha producido un problema a la hora de grabar el fichero de usuarios\n"); //Imprime error por pantalla
}
```

```c
void registrar_temperatura(registro* r, int i)      //Funcion que pide al usuario que intorduca la fecha y hora del registro
{                                                   //en el que se van a guardar las temperaturas
    char intro;

    printf("Nuevo registro\n");
    printf("Dia:");
    scanf_s("%d", &r->dia);
    printf("Mes:");
    scanf_s("%d", &r->mes);
    printf("Anio:");
    scanf_s("%d", &r->anio);
    printf("hora:");
    scanf_s("%d", &r->hora);
    printf("minutos:");
    scanf_s("%d", &r->minutos);

    intro = getchar();
}
```

4. CÓDIGO ARDUINO

- Programa fundamental para la comunicación entre el ordenador y el montaje físico que recibe las señales del exterior.
- Arduino recibe mensajes provenientes de Visual que determinarán qué función debe realizar: leer las señales recogidas por el sensor dht11, activar la alarma que saltara en función de las temperaturas recogidas, desactivar la alarma…
- Deben declararse los pines correspondientes a lo que están conectados los distintos elementos a la placa arduino y el puerto de comunicación entre ordenador y placa.

```
#include <DHT.h>      // INCLUIR LIBRERIA
#include <LiquidCrystal.h>           //importar libreria

enum Estados {MODO_MANUAL, MODO_ALARMA_ON,MODO_ALARMA_OFF,CAMBIAR_T_CRIT};
Estados estado=MODO_MANUAL;

const int DHTPin = 10;     //Pin al que conectamos el sensor
const int LedPin = 11;     //Pin al que conectamos la alarma en forma de led
LiquidCrystal lcd(8, 7, 6, 5, 4, 3, 2);    //Pines a los que conectamos la pantalla LCD
DHT dht(10, DHT11);    // Creamos el objeto del sensor usado, con su pin correspondiente

String mensaje_entrada;  //Mensaje en formato string pasado de Visual a Arduino
String mensaje_salida;   //Mensaje en formato string pasado de Arduino a Visual

void setup()
{
   delay(5000);   //5 segundos de espera
   Serial.begin(9600);
   dht.begin();        // Inicializacion del sensor DHT11
   lcd.begin(16, 1);   // inicializar la pantalla con la cantidad de (1) fila y (16)columna que esta tiene
   pinMode(LedPin, OUTPUT); //Inicializamos el Led e indicamos que funciona como OUTPUT
}
```

```
void loop ()
{
  delay(1000);  //Señal de espera de 1 segundo

  float t_crit=26.0;

  procesar_mensajes();   //Funcion que procesa el mensaje introducido desde Visual
                         //Es la funcion base para la comunicacion Visual-Arduino

  switch (estado)        //Estados por los que pasa el sistema:
  {
    case MODO_MANUAL:  //Modo en el que permanece mientras no se le introduce desde visual el mensaje necesario para
                       //activar la alarma.
        break;
    case MODO_ALARMA_ON:               //ALARMA ACTIVA, saltará cuando se supere la Tmax durante más tiempo del indicado
        modo_alarma(t_crit,5.0,2000,1); //Llamada a la funcion que determina si se enciende el led o cambia de
                                        //estado en funcion de la temperatura recibida comparandolo con los parametros mandados a la funcion
                                        //Dichos parametros son: temperatura critica, tiempo necesario superando dicha temperatura, tiempo
                                        //que estara encendido el led, y el modo de funcionamiento del led.
        break;
    case MODO_ALARMA_OFF:               //ALARMA INACTIVA, la alarma esta apagada, por lo que aunque se supere la tcrit no se encendera el led
        modo_alarma(t_crit,5.0,2000,0);  //Llamada a la misma funcion que antes, pasandole los mismos parámetros pero con el led apagado.
                                         //Si recibe la señal para encender la alarma pasa a MODO_ALARMA_ON, sino se pasa al MODO_MANUAL hasta
                                         //recibir dicha señal.
        estado=MODO_MANUAL;
        break;
    case CAMBIAR_T_CRIT:               //EStado en el que cambianos la t critica
        t_crit = cambiar_t_crit();     //Funcion para cambiar la temperatura critica
        break;
  }
}
```

```
void procesar_mensajes(void)        //FUNCION PARA IDENTIFICAR MENSAJES RECIBIDOS DESDE VISUAL
{
  if( Serial.available()> 0)
  {
    mensaje_entrada = Serial.readStringUntil('\n');   //Adjudica a la variable mensaje_entrada una cadena de caracteres mandada desde visual
    if (mensaje_entrada.compareTo("GET_TEMPERATURA")==0)   //Compara la cadena recibida con "GET_TEMPERATURA" para saber si el sensor
                                                           //debe empezar a leer temperaturas
    {
      float TemC = dht.readTemperature();            // Adjudica a la variable de tipo float la temperatura leida por el sensor

      if (isnan(TemC))   // Si no recibe un numero, imprime mensaje de error
      {
          lcd.setCursor(0, 0);            //Coloca en posicion el cursor de la pantalla lcd
          lcd.print("Revisar conexion");  //Imprime por pantalla el error correspondiente
      }
      else   //En el caso de que si que reciba una temperatura
      {
          lcd.setCursor(0, 0);
          lcd.print("Tem:" + String(TemC, 1) + "C  "); //Imprime en la pantalla LCD la temperatura correspondiente en grados centigrados
      }
      mensaje_salida=String("Tem:" + String(TemC, 1) + "C  " );  //y además, adjudica a la variable mensaje_salida dicha temperatura
    }                                                            //para enviarla de vuelta a visual y el programa haga la funcion correspondiente
                                                                 //con esta informacion.
    else
      if (mensaje_entrada.compareTo("SET_MODO_ALARMA")==0)   //Compara el mensaje enviado desde visual con "SET_MODO_ALARMA" y en el caso de que
                                                             //coincida , su funcionamiento dependera del estado en que se encuetre
      {
        if (estado==MODO_ALARMA_OFF || estado==MODO_MANUAL)  //Si estaba en modo manual(predeterminado) o con la alarma apagada, pasara
        {                                                    //a activar la alarma la cual saltara cuando se cumplan los parametros asignados
```

```
    {                                                   //a activar la alarma la cual saltara cuando se cumplan los parametros asignados
        estado=MODO_ALARMA_ON;
        mensaje_salida=String("ALARMA_OPERATIVA");      //Manda de vuelta a visual la situacion de la alarma
        lcd.print("ALARMA_OPERATIVA");                  //e imprime por pantalla el mismo mensaje
    }
    else
    {
      estado=MODO_ALARMA_OFF;                           //En el caso de encontrarse en el estado de alarma activa, dicha alarma
      mensaje_salida=String("ALARMA_INACTIVA");         //pasara a estar desactivada(MODO_ALARMA_OFF)
      lcd.print("ALARMA_INACTIVA");
     }
  }
  else
    if (mensaje_entrada.compareTo("CAMBIAR_T_CRIT")==0)   //SI manda cambiar tcrit se pasa al estado correspondiente
    {                                                     // y manda el mensaje a visual
      estado=CAMBIAR_T_CRIT;
      mensaje_salida=String("Introduce la nueva temperatura crítica:");
      lcd.print("CAMBIA TCRIT");
    }
    else
    {
     mensaje_salida="COMANDO DESCONOCIDO";          //En el caso de que haya un erro y no se corresponda el mensaje con ninguno de los anteriores
    }
     Serial.println(mensaje_salida);                //imprimira "COMANDO DESCONOCIDO""
                                                    //y con Serial.println devuelve a arduino el mensaje de salida.
}
}
```

```
void modo_alarma(float tem_max,int ms_sup, int ms_led, int modo)      //definimos la funcion modo alarma, la cual tiene 4 entradas y no
{                                                                     //devuelve ninguna variable
  static int  estado_alarma=0;            // Inicializamos el estado actual de la alarma
  static unsigned long  tiempo_alarma=0;  // Inicializamos el tiempo de la alarma
  static unsigned long tiempo_led_on=0;   // Inicializamos el tiempo led on
  float tem=0.0; // Inixializamos la temperatura captada por el sensor
  int i,rango;   //Declaramos dos variables de tipo entero

  if (modo==0)        //Cuando el modo (variable referente al estado del led) es igual a 0, el led esta apagado
  {
    estado_alarma=0;  //Por lo que el estado de la alarma se mantiene en 0
    digitalWrite(LedPin,LOW); //Indicamos que el led está apagado
  }
  else                                //En el caso de que el modo no sea 0 (SERA = 1):
    {
      float tem = dht.readTemperature();  //Asignamos a la variable la temperatura leida por el sensor
      rango = tem>=tem_max?1:0;           //En el caso de que la temperatura sea mayor o igual que la t critica, rango es 1, sino sera 0.
      switch (estado_alarma)              //A partir de aqui va pasando por distinto estados (inicio: estado 0) en funcion de las señales
      {                                   //que reciba.
        case 0:         // Estado inicial
         if (rango==1) // Si la temperatura supera la tcrit
         {
           tiempo_alarma=millis(); // Referencia temporal
           estado_alarma=1; //Pasa del estado 0 al 1
         }
         break;


        case 1: // En este estado, se ha superado ahce poco la t critica y detecta si se supera durante el tiempo suficiente
          if (millis()-tiempo_alarma>ms_sup) // Si la superación ha permanecido el tiempo mínimo requerido
          {
            estado_alarma=2;            //Cambia al estado 2
            digitalWrite(LedPin,HIGH); // El led pasa a estar encendido (HIGH)
```

```
        Serial.println("ATENCION: SE HA SUPERADO LA TEMPERATURA CRITICA"); //Manda a visual el mensaje anterior
        lcd.print("ATENCION: SE HA SUPERADO LA TEMPERATURA CRITICA");      //Imprime en la pantalla lcd que se supero la t critica
      }
      else
      if (rango==0)       //En el caso de que no se supere la t critica
        estado_alarma=0; //Regresa al estado inicial
    break;


    case 2:          // En este estado la alarma esta activa con el led encendido
     if (rango==0)  // Cuando deje de superarse la t critica
     {
       tiempo_led_on=millis(); // Empiesa el contador del tiempo que se mantiene encendido el led sin estar superando la t
       estado_alarma=3; // Cambia al estado 3
     }
     break;


     case 3: // El led sigue encendido pero la temperatura ya no es superada
     if (millis()-tiempo_led_on>ms_led) //Al pasar el tiempo en el que el led se mantiene encendido en estas circunstancias
     {
       digitalWrite(LedPin,LOW); //Manda una señal para apagar el led
       estado_alarma=0;          //y se vuelve al estado inicial
       Serial.println("ATENCION: YA NO ESTAMOS EN ESTADO DE ALERTA");
     }
     else
      if (rango==1) // Si se vuelve a superar tcrit
        estado_alarma=2; // Regresa a estado 2
  }
 }
}
```

```
float cambiar_t_crit()          //Funcion encargada de mandarle un mensaje a visual
{                               //para que el usuario introduzca la nueva t critica
    float t_crit;

    if(Serial.available()>0)
    {
      mensaje_entrada=Serial.readStringUntil('\n');
      float t_crit = mensaje_entrada.toFloat();
      estado=MODO_MANUAL;
      mensaje_salida=String("Se ha cambiado la temperatura critica");
    }
    return t_crit;
}
```