ARDUINO

Ethernet

PROGRAMACIÓN

TP_19

Índice de contenido

Todos los derechos reservados
© by Firtec Capacitación

Lectura de un sensor DHT22 por Socket UDP.

Enviar información por redes TCP-IP a nivel de electrónica puede resultar un tanto problemático, todos los protocolos y estándares están establecidos para comunicaciones informáticas y una computadora no entiende nada de un sensor DHT22 y los datos que este genera.
Para poder enviar estos datos por la red debemos "convertirlos" en algo que los protocolos de red entiendan.
En el siguiente ejemplo tenemos un sensor DHT22 conectado a una placa Arduino. Una vez que se leen los datos del sensor se escalan y convierten en una única larga cadena de caracteres donde se han codificado tanto la temperatura como la humedad (los dos datos que maneja del sensor), esta cadena de caracteres esta separada en dos campos, uno para cada variable que el sensor maneja, todo esto es responsabilidad del código Arduino y se realiza de la siguiente forma.

```
dtostrf(t, 4, 1, Temperatura);                  // Lee desde el sensor la Temperatura
dtostrf(h, 4, 1, Humedad);                      // Lee desde el sensor la Humedad
sprintf(cadena,"%s,%s",Temperatura,Humedad);    // Ensambla la trama que forma el datagrama
if (cadena_old != cadena) {                     // Los datos son distintos?
strcpy(cadena_old, cadena);                     // OK, entonces actualiza los nuevos datos
udp.beginPacket(udpAddress, udpPort);           // Se conecta con el servidor
udp.print(cadena);                              // Pone los datos en el socket
udp.endPacket();                                // Fin de transacción
```

Observe que en `cadena` mediante la función *sprintf()* se pasan a formato *String* (`"%s,%s"`) el dato de *Temperatura* y *Humedad* separados por una coma, (*circulo rojo*), esta coma oficia de separador de campo y los separadores de campo son de mucha utilidad en Python ya que hay métodos específicos para reconocerlos y tratarlos. Es decir que me puede llegar una larga cadena de caracteres pero el separador me indica donde empieza y termina un dato específico pudiendo tomar estos datos y mostrar cada uno en el lugar correcto en una ventana.
Para separar los campos mediante un separador en Python tenemos el método *split()*, en las siguiente líneas podemos ver como trabaja.
Recuerda como se ensambló la trama, primero la temperatura y luego la humedad según se lee aquí.

```
sprintf(cadena,"%s,%s",Temperatura,Humedad);
```

Entonces sabemos que en el primer campo [0] está la temperatura, así que pasamos a una variable *temperatura* todo lo que este en ese campo hasta encontrar el separador **","** luego sabemos que el siguiente campo [1] es la humedad, leemos todo lo que este ahí hasta el separador, en las siguiente líneas del programa Python pude ver como se decodifica el separador y los datos.

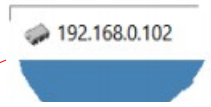
```
temperatura = dato.decode().split(",")[0]    # En temperatura lo que esté en [0]
label_dato_temp.config(text = temperatura)   # Mostrar en un cuadro de texto
humedad = dato.decode().split(",")[1]        # En humedad lo que esté en [1]
label_dato_hum.config(text = humedad)        # Mostrar en un cuadro de texto
```

Una vez que la temperatura y la humedad son leídos son mostrados en un cuadro de texto en la ventana Python.

Programa Arduino para el sensor DHT22 por Socket UDP.

Del lado del Aduino necesitamos el sensor DHT22 conectado de la misma forma que en el ejemplo antes visto, también una serie de bibliotecas necesarias para su funcionamiento como también para la red.

```
/**************************************************************************************
 * Placa Arduino: UNO-R3
 * Placa Ethernet: W5100
 * Sensor DHT22
 * Toolchain: Arduino IDE: 1.8.12
 *
 * Autor: daniel@firtec.com.ar
**************************************************************************************/
#include <Ethernet.h>         // Driver para el manejo de los procesos de red.
#include <EthernetUdp.h>      // Driver para el manejo de los sockets.
#include <Adafruit_Sensor.h>  // Driver para las comunicaciones con el sensor DHT22.
#include <DHT.h>              // Driver para el sensor DHT22.
#define DHTTYPE DHT22         // Tipo de sensor
const int DHTPin = 2;         // Pin para sensor DHT22
DHT dht(DHTPin, DHTTYPE);     // Se crea una instancia del sensor
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; // MAC de Ethernet
IPAddress ip(192, 168, 1, 200);         // Para usar una IP fija

static char Temperatura[5]="";
static char Humedad[5]="";
static char cadena[10];
static char cadena_old[10];

const char * udpAddress = "192.168.0.102";   // IP donde se encuentra el servidor
const int udpPort = 30000;                   // Puerto donde se reciben los datos
EthernetUDP udp;                             // Instancia para el socket
```



Ventana Python

Variables del programa

```
void setup(){
if (Ethernet.begin(mac) == 0) {
    Serial.println("No puedo configurar Ethernet usando DHCP");
  }
```

```
  //Ethernet.begin(mac, ip); // Configura Ethernet con IP fija (No es este el caso)
  Serial.begin(9600);         // Configura el serial para el terminal UART
  while (!Serial) {
    ;
  }
  if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println(F("ERROR en la placa Ethernet(")); // Verifica el hardware de red
    while (true) {
      delay(1); // Opsss!!!! No hay red, no hacer nada!!!!
    }
  }
  if (Ethernet.linkStatus() == LinkOFF) {
    Serial.println(F("El cable de red no esta conectado."));
  }
  Serial.print("IP asignada: ");
  Serial.println(Ethernet.localIP());   // Muestra la IP asignada
  udp.begin(udpPort);                   // Configura el socket
  dht.begin();                          // Inicia la electrónica del sensor
}
void loop(){
  float t = dht.readTemperature();      // Lee sensor temperatura
  float h = dht.readHumidity();         // Lee sensor humedad

  if (isnan(h) || isnan(t)) {           // Verifica si error de lectura
              Serial.println("ERROR!"); // Imprime cartel de error
              }
            else{                       // No hay error, procesa y envía los datos

   dtostrf(t, 4, 1, Temperatura);       // Procesa la temperatura
   dtostrf(h, 4, 1, Humedad);           // Procesa la humedad
   sprintf(cadena,"%s,%s",Temperatura,Humedad);  // Arma la trama que se monta en el socket.
   if (cadena_old != cadena) {                   // Los datos son distintos?
          strcpy(cadena_old, cadena);            // NO, entonces actualizar datos.
          udp.beginPacket(udpAddress, udpPort);  // Inicia el socket.
          udp.print(cadena); // Pone los datos en el socket y los transmite.
          udp.endPacket();   // Fin de transacción
```

```
      }
    }
  delay(3000); // Espera 3 segundos antes de nueva medición, requisito del sensor
}
```

Para recordar.

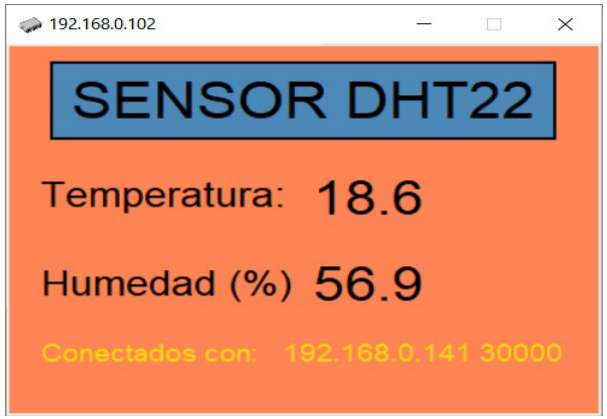
- El método *decode()* es necesario para decodificar correctamente los caracteres *Unicode*. Python maneja los datos con formato *Unicode* y los componentes electrónicos manejan *Bytes*.
- En los programas *ino* de Arduino los comentarios se inician con // esto es un comentario. Pero en Python con # esto es un comentario.

El programa Python para leer los datos del sensor DHT22 es el siguiente.

```
import socket
from tkinter import Frame
from tkinter import Text
from tkinter import Label
import sys
import select
import errno
import time

from tkinter import *
from tkinter import ttk
```

Bibliotecas usadas en el programa.

Se importan todos los módulos de Tkinter.




```
def conseguir_ip():
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        try:
            s.connect(('10.255.255.255', 0))
            IP = s.getsockname()[0]
        except:
            IP = '127.0.0.1'
        finally:
            s.close()
        return IP
```

Obtiene el IP asignada por el servidor DHCP.

```
def leer_sensor():
    global sock
    try:
        data,addr = sock.recvfrom(1500)      # Recibe tanto los datos del sensor como la IP
    except socket.error as e:
        err = e.args[0]
        if err == errno.EAGAIN or err == errno.EWOULDBLOCK:
            time.sleep(0.01)
            ventana.after(2, leer_sensor)
    else:
        temperatura = data.decode().split(",")[0]
        label_dato_temp.config(text = temperatura)
        humedad = data.decode().split(",")[1]
        label_dato_hum.config(text = humedad)
        label_IP_remoto.config(text = addr)
        ventana.after(2, leer_sensor)
```

Decodifica y muestra la información enviada desde Arduino.

```
class Aplicacion():
    def __init__(self):
        global sock
        global ventana
        global label_dato_temp
        global label_dato_hum
        global label_IP_remoto
        ventana = Tk()                      # Crea la ventana
        ventana.iconbitmap('11.ICO')        # Asigna un icono a la ventana
        ventana.config(bg="Steel Blue") # Color de fondo
        ventana.resizable(0,0)              # La ventana de tamaño fijo
        UDP_PORT = 30000                    # Puerto asignado al socket
        Dir_IP = conseguir_ip()             # Obtiene la IP asignada por el DHCP
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # Socket tipo UDP
        sock.bind((Dir_IP, UDP_PORT))   # Configura el socket con el IP y el puerto
        sock.setblocking(0)                 # Socket del tipo NO BLOQUEANTE
        ventana.title(Dir_IP)               # La ventana principal lleva como nombre la IP
        #----- Dibujar un rectangulo  ----------------------------
        self.canvas = Canvas(width=390, height=290, bg='coral') # Tamaño del rectángulo
```

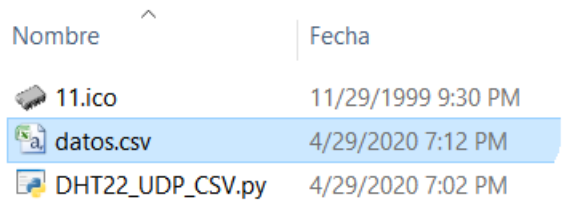
Variables globales para ser accedidas desde afuera de la clase.

```
        self.canvas.pack(expand=YES, fill=BOTH)                  # Ajustado a la ventana
        self.canvas.create_rectangle(30, 15, 363, 75, width=2, fill='Steel Blue')
        #-----------------------------------------------------------

        label_dht22 = Label(ventana, text="SENSOR DHT22", bg="Steel Blue", fg="black",
                            font=("Helvetica", 30))
        label_dht22.place(x=40, y=20)
        label_temperatura = Label(ventana, text="Temperatura:", bg="coral", fg="black",
                            font=("Helvetica", 20))
        label_temperatura.place(x=20, y=100)
        label_humedad = Label(ventana, text="Humedad (%)", bg="coral", fg="black",
                            font=("Helvetica", 20))
        label_humedad.place(x=20, y=170)
        label_dato_temp = Label(ventana, text="",bg="coral",fg="black",font=("Helvetica", 28))
        label_dato_temp.place(x=200, y=98)
        label_dato_hum = Label(ventana, text="", bg="coral",fg="black",font=("Helvetica", 28))
        label_dato_hum.place(x=200, y=166)
        label_IP_remoto = Label(ventana, text= "",bg="coral",fg="gold",font=("Helvetica", 14))
        label_IP_remoto.place(x=180, y=230)

        label_remoto = Label(ventana, text="Conectados con:", bg="coral", fg="gold",
                            font=("Helvetica", 14))
        label_remoto.place(x=20, y=230)

        leer_sensor()         # Se lee el sensor DHT22
        ventana.mainloop()    # Bucle de la ventana principal

def main():
    #cuenta = 0
    mi_app = Aplicacion()
    return 0
if __name__ == '__main__':
    main()
```

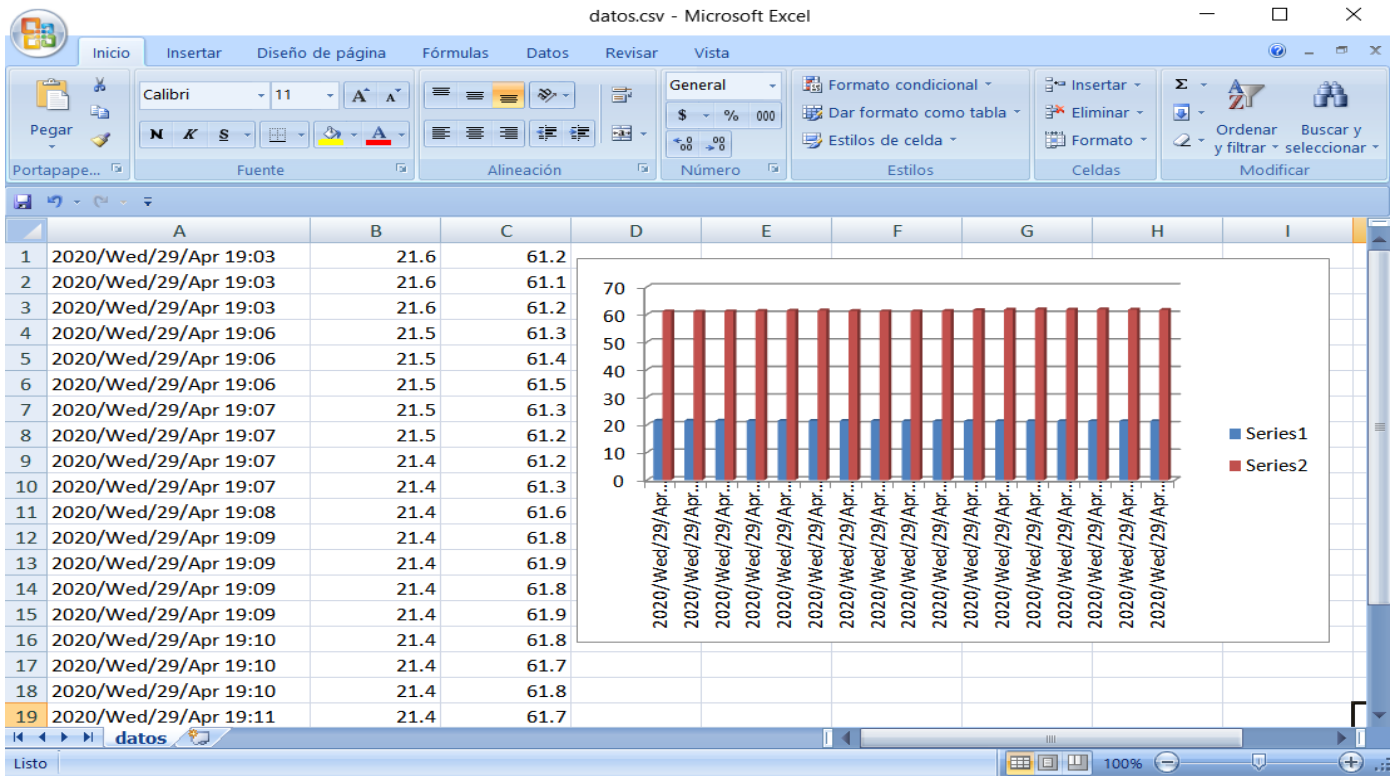
Almacenar datos en hoja de cálculo.

Siguiendo con el sensor DHT22 vamos a crear un programa Python que lea la temperatura y humedad del sensor, envíe los datos por un socket y almacene estos datos en un archivo compatible con una hoja de cálculo. El programa deberá cumplir con algunas condiciones.

- *Solo se enviará por la red información cuando alguno de las dos variables ha cambiado.*
- *Se guardará junto con la información del sensor la hora, minutos, día, mes, fecha y año.*
- *El archivo tendrá el formato CSV (**C**oma **S**eparate **V**alues).*



En el ejemplo propuesto el archivo se llamará *datos.csv*, y se creará en la misma carpeta donde se encuentre la fuente del código Python en la siguiente imagen se puede ver como se organizan los datos en el archivo generado.



La fecha y hora se extrae del propio sistema operativo usando una biblioteca de Python lo mismo que la creación del archivo *CSV* . Python tiene una biblioteca específica para la creación de este tipo de archivos.
Para este ejemplo vamos a necesitar tres bibliotecas que son los pilares de su funcionamiento.

- `import socket` *M*anejo de las comunicaciones por socket.
- `import csv` *E*ncargada del manejo de archivos ***CSV***.
- `import datetime` *S*e encarga del manejo del calendario y la hora, datos que extrae del sistema operativo.

En este ejemplo también vamos crear una ventana que este por encima de todas las demás, esto puede ser interesante cuando estamos recibiendo datos y es necesario que estén visible en todo momento, o algún seguimiento de alarmas, etc.
La línea de programa que hace esto es la siguiente:

```
ventana.wm_attributes("-topmost", 1)
```

Donde ventana es el nombre que tiene la ventana principal.
La idea es que Arduino envíe datos por la red cuando alguna de las dos variables (*temperatura o humedad*) sea distinta al valor anterior medido, esto es importante para no tener datos redundantes en la red lo que cargaría el tráfico inútilmente.
El trozo de código que maneja el archivo CSV es el siguiente:

```
format = "%Y/%a/%d/%b %H:%M"                        # Define el formato de los datos.
        today = datetime.datetime.today()           # Obtiene los datos del sistema.
        s = today.strftime(format)                  # Aplica el formato a los datos leídos.
        temperatura = data.decode().split(",")[0]# Obtiene la temperatura.
        if(temperatura != "-"):# Si es distinto a "-" procesar de lo contrario descartar.
                label_datoT.config(text = temperatura)# Muestra la temperatura en label.
                humedad = data.decode().split(",")[1]  # Obtiene la humedad.
                label_datoH.config(text = humedad)     # Muestra la humedad en label.
                datos = [[s,temperatura, humedad]]     # Prepara los datos para ser escritos.
                with open('datos.csv', 'a',newline='') as csvfile: # Crea o el archivo con
                    writer = csv.writer(csvfile, delimiter=',')    # separador ´,´
                    writer.writerows(datos)  # Crea o agrega datos al archivo datos.csv
```

Si el archivo *datos.csv* existe, se agregan datos luego del último dato escrito, si no existe lo crea.
El código completo del programa Python es el siguiente.

```
import socket
import csv
from tkinter.messagebox import showinfo
from tkinter import Frame
from tkinter import Text
from tkinter import Label
import sys
import select
import errno
import time
import datetime
```

Bibliotecas necesarias

```
from tkinter import *

def conseguir_ip():
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        try:
            s.connect(('10.255.255.255', 0))
            IP = s.getsockname()[0]
        except:
            IP = '127.0.0.1'
        finally:
            s.close()
        return IP

class Aplicacion():
    def __init__(self):
        global sock
        global ventana
        global label_datoT
        global label_IP_remoto
        global label_datoH
        ventana = Tk()                  # Ventana principal
        ventana.wm_attributes("-topmost", 1) # Ventana encima de todas!!!
        ventana.iconbitmap('11.ICO')         # Icono de la ventana.
        ventana.config(bg="Steel Blue")      # Color de fondo de la ventana.
        ventana.geometry("380x100")          # Tamaño de la ventana.
        ventana.resizable(0,0)               # Ventana de tamaño fijo.
        UDP_PORT = 30000                     # Puerto del socket
        Dir_IP = conseguir_ip()              # Obtiene la IP de la red.
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # Cre ale socket UDP
        sock.bind((Dir_IP, UDP_PORT))        # Socket servidor.
        sock.setblocking(0)                  # Socket del tipo no bloqueante.
        ventana.title(Dir_IP)                # Nombre de la ventana será el IP.
        label_temperatura = Label(ventana, text="Temperatura: ",bg="Steel Blue",fg="black",
                                                                font=("Helvetica", 14))
        label_temperatura.place(x=30, y=50)
```

Datos globales para acceder las variables desde afuera de la clase.

```
        label_datoT = Label(ventana, text="---",bg="Steel Blue",fg="black",font=("Helvetica",
                                                                                    14))
        label_datoT.place(x=150, y=50)

        label_humedad = Label(ventana, text="Humedad: ",bg="Steel Blue",fg="black",
                                                                font=("Helvetica", 14))
        label_humedad.place(x=210, y=50)
        label_datoH = Label(ventana, text="---", bg="Steel Blue", fg="black",
                                                                font=("Helvetica", 14))
        label_datoH.place(x=300, y=50)
        label_remoto = Label(ventana, text= "Conectado con: ", bg="Steel Blue", fg="gold",
                                                                font=("Helvetica", 14))
        label_remoto.place(x=20, y=20)
        label_IP_remoto = Label(ventana, text=" SIN CONEXIÓN ", bg="Steel Blue", fg="gold",
                                                                font=("Helvetica", 14))
        label_IP_remoto.place(x=160, y=20)
        leer_socket()
        ventana.mainloop()

def leer_socket():                          # Se leen datos desde la electrónica.
    global sock
    try:
        data,addr = sock.recvfrom(1500)
    except socket.error as e:
        err = e.args[0]
        if err == errno.EAGAIN or err == errno.EWOULDBLOCK:
            time.sleep(0.01)
            ventana.after(2, leer_socket)
    else:                                   # Se define y crea el archivo con los datos.
        format = "%Y/%a/%d/%b %H:%M"
        today = datetime.datetime.today()
        s = today.strftime(format)
        temperatura = data.decode().split(",")[0]
        if(temperatura != "-"):
                label_datoT.config(text = temperatura)
                humedad = data.decode().split(",")[1]
```

```
                label_datoH.config(text = humedad)
                datos = [[s,temperatura, humedad]]
                with open('datos.csv', 'a',newline='') as csvfile:
                    writer = csv.writer(csvfile, delimiter=',')
                    writer.writerows(datos)

        label_IP_remoto.config(text = addr)
        ventana.after(2, leer_socket)

def main():
    mi_app = Aplicacion()
    return 0

if __name__ == '__main__':
    main()
```

En el pin 8 de Arduino hay un botón que si se presiona envía al servidor -,- ocupando el espacio de la temperatura y la humedad , esto solo a los efectos de comprobar que el sistema está en línea con el esclavo de lo contrario habría que esperar a que Arduino envíe los primeros datos para saber si están conectados.

Recuerde que deberá esperar que Arduino haga una medición para recibir los primeros datos en la interfaz Python.

Las bibliotecas usadas para el sensor son las mismas ya vistas en ejemplos anteriores, el código completo para Arduino es el siguiente.

```
/*************************************************************************************
 * Placa Arduino: UNO-R3
 * Placa Ethernet: W5100
 * Toolchain: Arduino IDE: 1.8.12
 *
 * Autor: Daniel Schmidt
*************************************************************************************/
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#define DHTTYPE DHT22    // Tipo de sensor
#define ok 9             // LED para indicar sistema OK
#define test 8 // Botón de prueba para verificar si el sistema encuentra el servidor
const int DHTPin = 2;                   // Pin del sensor
DHT dht(DHTPin, DHTTYPE);               // Instancia del sensor
volatile unsigned int segundos = 0;     // Variable para la cuenta de los segundos
```

```
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 1, 200);    // Para IP fija

static char Temperatura[5]="";
static char Humedad[5]="";
static char cadena[10] = "";
static char cadena_old[10];
bool bandera = false;

const char * udpAddress = "192.168.0.102";   // IP donde se encuentra el servidor Python
const int udpPort = 30000;                   // Puerto para los datos
EthernetUDP udp;              // Instancia para enviar y recibir paquetes sobre UDP
```


```
void setup(){
pinMode(test,INPUT_PULLUP);
pinMode(ok, OUTPUT);
digitalWrite(ok, LOW);
if (Ethernet.begin(mac) == 0) {
    Serial.println(F("No puedo configurar Ethernet usando DHCP"));
  }
  //Ethernet.begin(mac, ip); // Configura Ethernet con IP fija, no es este el caso

  Serial.begin(9600);
  while (!Serial) {
    ;
  }

  if (Ethernet.hardwareStatus() == EthernetNoHardware) {    // Hardware de red OK?
    Serial.println(F("ERROR en la placa Ethernet("));       // NO, ERROR!!!
    while (true) {
      delay(1);
    }
  }
  if (Ethernet.linkStatus() == LinkOFF) {
    Serial.println(F("El cable de rer no esta conectado."));
```

```
  }
  Serial.print(F("IP asignada: "));
  Serial.println(Ethernet.localIP());
  udp.begin(udpPort);
  dht.begin();

  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1  = 0;
  OCR1A = 15625;
  TCCR1B |= (1 << WGM12)|(1<<CS10)|(1 << CS12);
  TIMSK1 |= (1 << OCIE1A);
  digitalWrite(ok, HIGH); // Sistema listo y el linea !!!
}

void loop(){
  if (digitalRead(test) == LOW && bandera == false) {
      udp.beginPacket(udpAddress, udpPort);       // Conecta con el servidor
      char cadena2[6] = {"-,-"};                  // Datos de control
      udp.print(cadena2);     // Pone los datos de prueba en el socket
      udp.endPacket();        // Fin de transacción
      bandera = true;         // Bandera para verificar cuando se suelta el botón
  }
  if (digitalRead(test) == HIGH) { // Espera que suelte el botón
    bandera = false;
}
   if (strcmp(cadena_old, cadena)) {    // Compara el dato previo con el actual
        strcpy(cadena_old, cadena);     // Entonces actualiza datos
        //Serial.println(cadena);
        udp.beginPacket(udpAddress, udpPort);     // Conecta con el servidor
        udp.print(cadena);                        // Pone los datos en el socket
        udp.endPacket();                          // Fin de transacción
      }
  }
```

Con un cristal de 16Mhz como tiene Arduino, este módulo ajusta el Timer para genera una interrupción cada segundo.

Configura el timer para llevar la cuenta de los segundos.

Al apretar el botón se envía al servidor una trama de control, en ese momento el servidor deberá mostrar la IP y el puerto del cliente. Esto permite saber que ambas puntas están conectadas.

```
/*****************************************************************************************
  • Rutina de interrupción del Timer 1
  • Cada un segundo el Timer interrumpe y el flujo del programa viene a esta función,
    contando los segundos se pueden contar minutos, horas, etc.
  • Este será el tiempo entre cada muestra, se puede ajustar
  • según la necesidad.
*****************************************************************************************/
 ISR(TIMER1_COMPA_vect){
segundos++;
if(segundos > 15){                      // En este caso se esperan 15 segundos entre muestras.
  float t = dht.readTemperature();      // Lee sensor temperatura.
 float h = dht.readHumidity();          // Lee sensor humedad.

 if (isnan(h) || isnan(t)) {            // Verifica si error de lectura.
             Serial.println("ERROR!");
             }
 else{                                  // No hay error, procesa y envía los datos.
  dtostrf(t, 4, 1, Temperatura);        // Procesa la temperatura.
  dtostrf(h, 4, 1, Humedad);            // Procesa la humedad.
  sprintf(cadena,"%s,%s",Temperatura,Humedad); // Ensambla la trama.
  segundos = 0;
   }
  }
 }
```

NOTA I.

Como el pin 8 tiene activas las resistencias Pull-Up su conexión es entre el pin 8 y GND siempre con un condensador de .01uF en paralelo con el condensador para evitar los rebotes eléctricos.

NOTA II.

El ejemplo anterior tiene un grado de complejidad importante sobre todo por el código Python usado. El objetivo es solo mostrar el potencial que tienen los sockets vinculados a Arduino y como se pueden conectar datos con programas informáticos como una hoja de cálculo o incluso una base de datos.