Universidad de Murcia

Máster Universitario en Tecnologías de Análisis de Datos Masivos: Big Data

Memoria de prácticas

Internet de las Cosas en el contexto de Big Data

Autores:
Juan Luis Serradilla Tormos
Alejandro Pérez Belando

Curso 2024-2025

Índice

1. Práctica 1: introducción a Arduino

A lo largo de esta práctica, realizada el miércoles 13/03/2025, se exponen a grandes rasgos los conceptos básicos sobre Arduino, se hace un primer programa introductorio de ejemplo: Blink, que muestra un parpadeo rítmico del led integrado de la placa de Arduino y dos ejercicios que se dejan al alumno.

1.1. Hola Mundo con Arduino: Blink

Para realizar este primer ejemplo, hay que seguir los siguientes pasos:

1. Para conectar el arduino y configurar los puestos:
   a) Conectar Arduino por USB al PC y descargar (y abrir) el IDE de Arduino.
   b) Seleccionar la placa: `Tools` → `Board` → `Arduino AVR Boards` → `Arduino Ehernet`
   c) Seleccionar el puerto: `Tools` → `Port` → `COM5` (en este casotuvimos que concederle permisos de lectura y escritura a todos los usuarios, que bastó con ejecutar `sudo chmod a+rw /dev/ttyUSB0` desde la terminal)
2. Abrir el programa `Blink`: `File` → `Examples` → `01.Basics` → `Blink`

Blink en Arduino

```
// Se ejecuta una vez cuando se resetea o encinede la placa
void setup() {
  // inicializamos el LED integrado como el outpt
  pinMode(LED_BUILTIN, OUTPUT);
}

// Esta función se ejecuta una y otra vez sin detenerse
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // Encender el LED
  delay(1000);                      // Esperar (1000 milisegundos)
  digitalWrite(LED_BUILTIN, LOW);   // Apagar el LED
  delay(1000);
}
```

3. Verificar el código seleccionando `Verify`
4. Cargarlo en la placa con la selección `Upload`. una vez cargado, comenzará el parpadeo. Como se ha explicado en el código, este parpadeo se repetirá indefinidamente hasta que se desconecte la placa o se suba otro programa.

1.2. Señal SOS

Se pide, básicamente, realizar la señal de S.O.S. con el LED. Esta señal se compone, en código Morse, de tres puntos (tres destellos cortos), tres rayas (destellos largos) y de nuevo tres puntos.

S.O.S. en Arduino

```
// Vamos a crear un blink que indique SOS:
// - 3 destellos cortos (puntos)
// - 3 destellos largos (rayas)
// - 3 destellos cortos (puntos)

// Aquí se inicializan las variables
void setup() {
  // Inicializamos la variable LED_BUILTIN como un OUTPUT
  pinMode(LED_BUILTIN, OUTPUT);
}

// Creamos una funcion para el SOS
void SOS() {
  // Genera 3 puntos (cada "punto" dura 250 ms encendida)
  for (int i = 0; i < 3; i++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
  }

  // Genera 3 rayas (cada "raya" dura 1000 ms encendida)
  for (int i = 0; i < 3; i++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
  }

  // Genera 3 puntos nuevamente
  for (int i = 0; i < 3; i++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
  }

  // Tiempo adicional antes de repetir la secuencia
  delay(1000);
}

// Aquí definimos la función "loop", que como dice su nombre será un loop en el
// que el código se iniciará infinitamente
void loop() {
  SOS();
}
```

1.3. Controlar la activación y desactivación del S.O.S.

Hacer código que escribiendo una letra por el puerto serial y active/desactive la señal S.O.S. y muestre por pantalla el estado de este.

S.O.S. en Arduino

```
// 'input' es booleana e inicialmente es falso
bool input = false;

// Se inicializan las variables, igual que antes
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600); // Comenzar la comunicación serial
}


// Aquí definimos la función "loop", que como dice su nombre será un loop
// en el que el código se iniciará infinitamente
void loop() {
  if (Serial.available() > 0) {  // Verificar si hay datos en el puerto serial
    char entrada = Serial.read();  // Leer el carácter ingresado

    if (entrada == 'S' || entrada == 's') {  // Si la tecla es 'S' o 's'
      input = !input; // Cambia el valor de input
      if (input) { // Si input es verdadero,
        Serial.print("SOS activado\n"); // Muestra "SOS activado"
      }
      else { // Si input es falso,
        Serial.print("SOS desactivado\n"); // Muestra "SOS desactivado"
      }
    }
  }

  if (input) { //Si en algún momento 'input' es true
    SOS(); // Se ejecuta la misma función SOS
  }
}

// Misma función que para el ejercicio antertior
void SOS() {
  // Genera 3 puntos (cada "punto" dura 250 ms encendida)
  for (int i = 0; i < 3; i++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
  }
```

```
  // Genera 3 rayas (cada "raya" dura 1000 ms encendida)
  for (int i = 0; i < 3; i++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
  }

  // Genera 3 puntos nuevamente
  for (int i = 0; i < 3; i++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
  }

  // Tiempo adicional antes de repetir la secuencia
  delay(1000);
}
```

2. Practica 2: despliegue de plataforma IoT

A lo largo de esta práctica, realizada el miércoles 26/03/2025, se pide realizar un sencillo montaje a partir de una placa de Arduino y un sensor para realizar lecturas de la humedad y la temperatura.

2.1. Enunciado

Para la realización de esta práctica se pide lo siguiente:

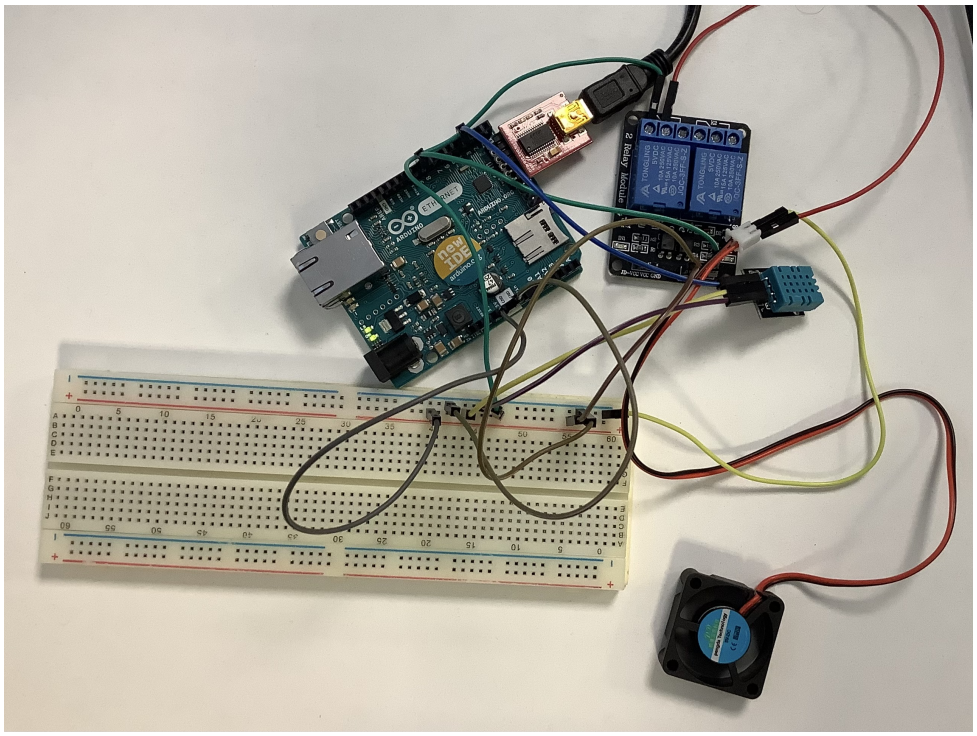
1. Copiar las librerías proporcionadas (carpetas DHT11 y thinger.io) a la ubicación *Documentos/Arduino/libraries/*.
2. Medir en intervalos regulares de 5 segundos la temperatura y la humedad desde el sensor DHT11. Almacenar esta información en dos variables que deben mostrarse por la pantalla a través de la consola serie.
3. Fijar un umbral de dos grados por encima de la temperatura actual de la sala, de modo que si la temperatura supera ese valor, el relé debe dar paso a la corriente y hacer funcionar el ventilador. Cuando la temperatura vuelva a bajar, el relé debe apagarse y el ventilador detener su funcionamiento.
4. Replicar este comportamiento para la humedad, controlando el diodo integrado en el Arduino (pin digital 9)

2.2. Materiales y montaje experimental

Los materiales empleados en esta práctica son:

- Una placa Arduino Ethernet.
- Un cable de alimentación.
- Un sensor de temperatura y humedad.
- Un relé.
- Un pequeño ventilador.
- Una protoboard.

Y el montaje experimental se ha realizado tal y como se muestra en la siguiente figura:



2.3. Código empleado

Detección de temperatura y humedad

```
#include <DHT.h>

#define DHTPIN 3 // Pin de datos del DHT11
#define DHTTYPE DHT11 // Definimos el tipo de sensor DHT11
DHT dht(DHTPIN, DHTTYPE);

#define RELAY_PIN 8 // Pin para el relé del ventilador
#define LED_PIN 9 // Pin para el LED (diodo integrado)

// Variables para almacenar umbrales
float tempThreshold;
float humThreshold;

void setup() {
Serial.begin(9600);
dht.begin();

pinMode(RELAY_PIN, OUTPUT);
pinMode(LED_PIN, OUTPUT);
```

```
// Lectura inicial para fijar los umbrales
float baselineTemp = dht.readTemperature();
float baselineHum = dht.readHumidity();

// Se suma 2 unidades a la lectura base para definir el umbral
tempThreshold = baselineTemp + 2;
humThreshold = baselineHum + 2;

Serial.print("Temperatura base: ");
Serial.println(baselineTemp);
Serial.print("Umbral de temperatura: ");
Serial.println(tempThreshold);
Serial.print("Humedad base: ");
Serial.println(baselineHum);
Serial.print("Umbral de humedad: ");
Serial.println(humThreshold);
}

void loop() {
// Lectura de temperatura y humedad
float t = dht.readTemperature();
float h = dht.readHumidity();

// Mostrar los valores por el monitor serie
Serial.print("Temperatura: ");
Serial.print(t);
Serial.print(" °C, Humedad: ");
Serial.print(h);
Serial.println(" %");

// Control del relé para el ventilador según la temperatura
if (t > tempThreshold) {
digitalWrite(RELAY_PIN, HIGH); // Activa relé y ventilador
Serial.println("Ventilador ACTIVADO");
} else {
digitalWrite(RELAY_PIN, LOW); // Desactiva relé y ventilador
Serial.println("Ventilador DESACTIVADO");
}

// Control del LED para la humedad
if (h > humThreshold) {
digitalWrite(LED_PIN, HIGH); // Enciende LED
Serial.println("LED HUMEDAD ACTIVADO");
} else {
digitalWrite(LED_PIN, LOW); // Apaga LED
Serial.println("LED HUMEDAD DESACTIVADO");
}

// Espera 5 segundos antes de la siguiente lectura
delay(5000);
}
```

2.4. Resultados

Para visualizar el correcto funcionamiento de este sistema, hemos realizado una serie de experimentos y tomado capturas de pantalla de la salida del monitor serial.

- **Situación 1**: detección de la temperatura y la humedad del espacio de trabajo, sin alteraciones. Como hemos establecido $2°C$ de umbral para la temperatura y un $2\,\%$ para la humedad y por el momento no se ha producido ninguno de estos incrementos, tanto el ventilador como el led indicador de aumento de humedad están desactivados. Esto se puede ver en la siguiente figura:

```
Temperatura: 28.00 °C, Humedad: 38.00 %
Ventilador DESACTIVADO
LED HUMEDAD DESACTIVADO
```

- **Situación 2**: detección de la temperatura y la humedad del espacio de trabajo, con alteraciones. Ahora hemos colocado las manos alrededor del sensor de temperatura (sin llegar a tocarlo) para aumentar la temperatura y la humedad del ambiente. Al haber establecido esos umbrales que son fácilmente superables, se ve rápidamente cómo primero la humedad aumenta, traspasa este umbral y se activa el led indicador de humedad, pero el ventilador queda desactivado porque la temperatura ambiente no ha subido lo suficiente en este punto.

```
Temperatura: 27.00 °C, Humedad: 41.00 %
Ventilador DESACTIVADO
LED HUMEDAD ACTIVADO
```

- **Situación 3**: Tras unos segundos con las manos cerca del sensor, tanto la humedad como la temperatura han aumentado lo suficiente como para que ahora se superen ambos umbrales y tanto el led identificador de humedad como el ventilador se activen.

```
Temperatura: 30.00 °C, Humedad: 83.00 %
Ventilador ACTIVADO
LED HUMEDAD ACTIVADO
```

3. Practica 3: creación de webserver

A lo largo de esta práctica, realizada el martes 01/04/2025, se pide lanzar un sencillo servidor web local para, a partir del montaje y el código de la práctica anterior, mostrar las últimas 10 lecturas del sensor de temperatura y humedad con una tasa de refresco de 1 segundo. Además, debe incorporar un botón digital para activar y desactivar el ventilador.

3.1. Enunciado

Para la realización de esta práctica se pide lo siguiente:

1. El Arduino debe hospedar una sencilla web en html.
2. Se deben medir a intervalos iguales de 1 segundo la temperatura y humedad desde el sensor DHT11 y almacenarlas en dos arrays de 10 posiciones (en ciclos posteriores, machacar los datos).
3. Cuando se haga un acceso a la página web, se debe mostrar los últimos 10 valores de ambas magnitudes.
4. Además, se debe diseñar un botón dentro de la página web, que permita encender/apagar el ventilador a través del relé. También mostrar el estado del ventilador (ON/OFF)

3.2. Materiales y montaje experimental

Los materiales empleados en esta práctica así como su montaje experimental son idénticos a los empleados en la práctica anterior.

3.3. Código empleado

webserver que muestra la temperatura y humedad

```
#include <DHT.h>
#include <Ethernet.h>

// Configuración del sensor DHT11
#define DHTPIN 3         // Pin de datos del DHT11
#define DHTTYPE DHT11    // Definimos el tipo de sensor DHT11
DHT dht(DHTPIN, DHTTYPE);

// Configuración de pines para el relé y LED (opcional)
#define RELAY_PIN 8      // Pin para el relé del ventilador
#define LEDPIN 9         // Pin para un LED indicador (si se usa)

// Configuración de la MAC (está detrás de la placa, 6 parejas de dígitos)
byte mac[] = {
  0x90, 0xA2, 0xDA, 0x10, 0x80, 0x8B
};
EthernetServer server(80);
```

```
// Variables para almacenar las últimas 10 lecturas
float temperaturas[10];
float humedades[10];
int indiceArray = 0;  // Índice actual en los arrays

// Variables para el muestreo periódico del sensor (cada 1 segundo)
unsigned long sensorPrevTime = 0;
const unsigned long sensorInterval = 1000;

// Variables para el estado del ventilador (relé)
bool estadoVentilador = false;

// Variables para procesar la petición HTTP
String method = String(4);
bool method_readed = false;

void setup() {
  Serial.begin(9600);
  dht.begin();

  // Configurar pines
  pinMode(DHTPIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(LEDPIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);

  // Inicialización de Ethernet
  Ethernet.begin(mac);
  server.begin();
  Serial.print("Servidor disponible en: ");
  Serial.println(Ethernet.localIP());

  // Inicializar arrays con la primera lectura del sensor
  float tempInicial = dht.readTemperature();
  float humInicial = dht.readHumidity();
  for (int i = 0; i < 10; i++) {
    temperaturas[i] = tempInicial;
    humedades[i] = humInicial;
  }
}

void loop() {
  // Actualización de la lectura del sensor cada 1 segundo
  unsigned long tiempoActual = millis();
  if (tiempoActual - sensorPrevTime >= sensorInterval) {
    sensorPrevTime = tiempoActual;
    float t = dht.readTemperature();
    float h = dht.readHumidity();

    temperaturas[indiceArray] = t;
    humedades[indiceArray] = h;
    indiceArray = (indiceArray + 1) % 10;
```

```
  Serial.print("Temperatura: ");
  Serial.print(t);
  Serial.print(" °C, Humedad: ");
  Serial.print(h);
  Serial.println(" %");
}

// Creamos la web
EthernetClient client = server.available();
if (client) {
  Serial.println("new client");
  boolean currentLineIsBlank = true;

  while (client.connected()) {
    if (client.available()) {
      char c = client.read();
      Serial.write(c);

      if (method.length() < 4)
        method += c;

      if (method.length() == 4 && method_readed == false) {
        method_readed = true;

        if (method == "POST") {
          estadoVentilador = !estadoVentilador;
          digitalWrite(RELAY_PIN, estadoVentilador ? HIGH : LOW);
          Serial.print("Ventilador: ");
          Serial.println(estadoVentilador ? "ON" : "OFF");
        }
      }

      if (c == '\n' && currentLineIsBlank) {
        method = "";
        method_readed = false;

        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println("Connnection: close");
        client.println();
        client.println("<!DOCTYPE HTML>");
        client.println("<html>");
        client.println("<meta http-equiv=\"refresh\" content=\"1\">");
        client.println("<h1>Lecturas DHT11</h1>");
        client.println("<br />");

        client.println("<table border='1'>");
        client.print("<tr><th>#</th>");
        client.print("<th>Temperatura (°C)</th>");
        client.println("<th>Humedad (%)</th></tr>");
```

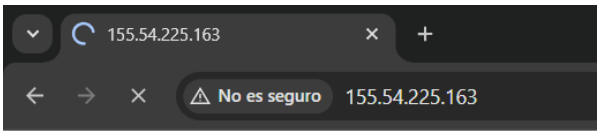
```
          for (int i = 0; i < 10; i++) {
            int pos = (indiceArray + i) % 10;
            client.print("<tr><td>");
            client.print(i + 1);
            client.print("</td><td>");
            client.print(temperaturas[pos]);
            client.print("</td><td>");
            client.print(humedades[pos]);
            client.println("</td></tr>");
          }
          client.println("</table>");
          client.println("<br />");

          client.print("<p>Ventilador: ");
          client.print(estadoVentilador ? "ON" : "OFF");
          client.println("</p>");

          client.print("<form action='/' method='post'>");
          client.print("<input type='submit' value='ON/OFF'>");
          client.println("</form>");

          client.println("</html>");
          client.println("</html>");
          break;
        }

        if (c == '\n') {
          currentLineIsBlank = true;
        } else if (c != '\r') {
          currentLineIsBlank = false;
        }
      }
    }

    delay(1000);

    // Cerramos la paginaa
    client.stop();
    Serial.println("client disconnected");
  }
}
```

3.4. Resultados

Una vez validado y lanzado el programa en la placa *Arduino Ethernet*, devolverá una dirección IP de un servidor local donde se encuentra la tabla con los elementos requeridos. En nuestro caso la dirección IP generada ha sido **155.54.225.63** y basta con pegarla en la barra de un buscador web para acceder a ella. **Importante**: para poder ver el servidor web debemos conectar tanto la placa como el ordenador a (en este caso) la web de la universidad empleando los cables ethernet que hay en la clase.

Para visualizar el correcto funcionamiento de este servidor, hemos tomado capturas de pantalla de la salida de la página web.

- **Situación 1 (izquierda)**: con el ventilador apagado
- **Situación 2 (derecha)**: con el ventilador encendido una vez pulsado el botón **ON/OFF**

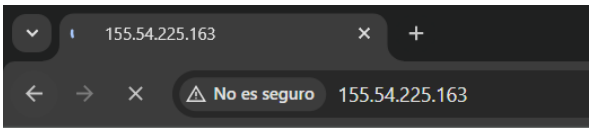


Lecturas DHT11

#	Temperatura (°C)	Humedad (%)
1	28.00	35.00
2	28.00	35.00
3	28.00	35.00
4	28.00	36.00
5	28.00	36.00
6	28.00	36.00
7	28.00	36.00
8	28.00	36.00
9	28.00	35.00
10	28.00	35.00

Ventilador: OFF

ON/OFF



Lecturas DHT11

#	Temperatura (°C)	Humedad (%)
1	28.00	33.00
2	28.00	33.00
3	28.00	33.00
4	28.00	33.00
5	28.00	33.00
6	28.00	33.00
7	28.00	33.00
8	28.00	33.00
9	28.00	33.00
10	28.00	33.00

Ventilador: ON

ON/OFF

4. Practica 4: Thinger.io

A lo largo de esta práctica, realizada el lunes 07/04/2025, se pide crear un dashboard en la platsforma thinger.io donde consultar la temperatura y humedad del sensor DHT11 y un botón para activar y desactivar el ventilador de forma manual, generar un cubo de datos para almacenar los valores recogidos y crear una alerta para que una vez por minuto se envíe un correo de notificación cuando la temperatura supere un umbral dado.

4.1. Enunciado

Para la realización de esta práctica se pide lo siguiente:

1. Registrarse en la web de thinger.io y declarar un nuevo dispositivo: asignar un nombre y descripción al nuevo dispositivo, generar credenciales y guardar estos datos.

2. Programar el Arduino:

   a) Instalar las librerías para Arduino. Copiar el contenido de "librerias.arduino.zip" a: C:

   b) Código: registrar dispositivo con las credenciales obtenidas en la plataforma, configurar la red (DHCP) utilizando la dirección MAC que aparece en la pegatina del propio Arduino e implementar funcionalidad. Declarar los 3 elementos con los que trabajamos (temp., humedad y relé) como "cosas" de cara a la plataforma: ojo! Dos elementos (temp y humedad) son variables de salida y el relé una variable de entrada (su declaración cambia).

3. Configurar dashboard: una vez que el dispositivo aparezca como conectado en el gestor de dispositivos de thinger, crear un dashboard para monitorizar la temperatura y humedad y un interruptor para el relé. Al declarar cada widget, emplear un intervalo de muestreo ("sampling interval") de 10 segundos.

4. Data bucket: añadir un "cubo de datos" para almacenar los valores de temperatura y humedad.

5. End point: crear un "punto final" de tipo e-mail e incluir en el código de Arduino un aviso cuando la temperatura supere cierto umbral. Las notificaciones no se gestionan desde la plataforma sino desde el dispositivo, es decir, el control de temperatura y disparo del evento se hace en Arduino.

6. AppAndroid: descargar la app para Android y mostrar un dashboard (más básico) en el móvil.

4.2. Materiales y montaje experimental

Los materiales empleados en esta práctica así como su montaje experimental son idénticos a los empleados en la práctica anterior.

4.3. Código empleado para programar el Arduino:

webserver que muestra la temperatura y humedad

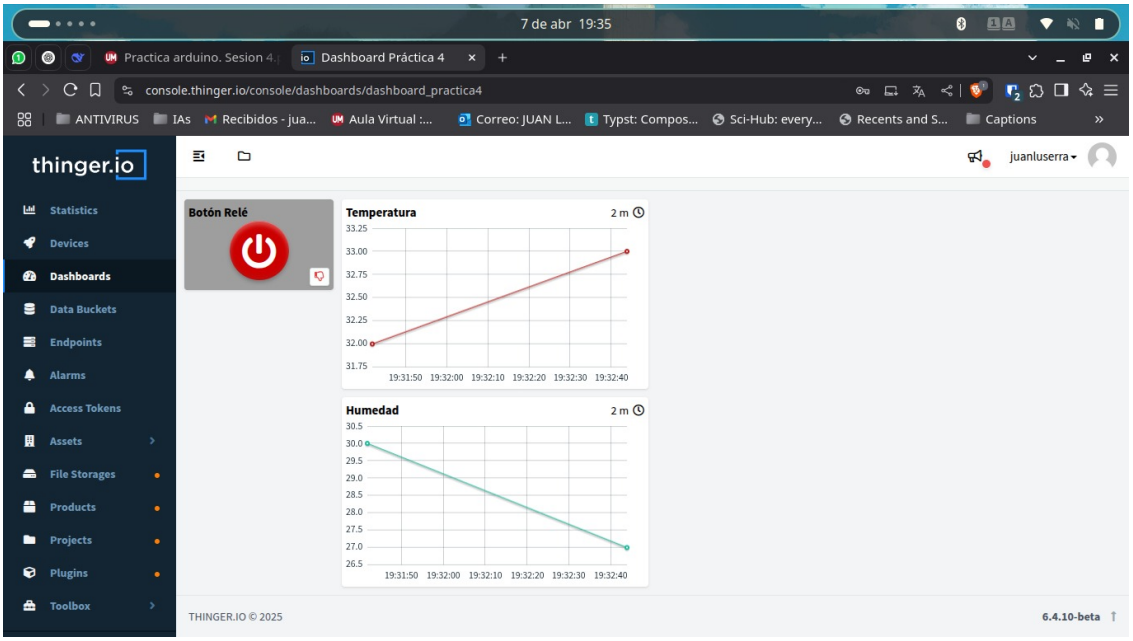
```
#include <Ethernet.h>
#include "DHT.h"
#include <ThingerEthernet.h>

#define DHTPIN 3
#define RELAYPIN 8  // Corregido de DELAYPIN a RELAYPIN
#define DHTTYPE DHT11  // Definir el tipo de sensor DHT

// Rellenamos los datos necesarios de Thinger.io
#define USERNAME "juanluserra"
#define DEVICE_SENSOR_ID "arduino-practica4"
#define DEVICE_SENSOR_CREDENTIAL "IWuPP!8yYdW7E2£l"

// Inicializamos los objeto de Thinger.io y DHT
ThingerEthernet thing(USERNAME, DEVICE_SENSOR_ID, DEVICE_SENSOR_CREDENTIAL);
DHT dht(DHTPIN, DHTTYPE);

// Guardamos la MAC del Arduino e iniciamos la conexión con Ehternet
byte mac[] = {0x90, 0xA2, 0xDA, 0x10, 0x81, 0x16};
EthernetServer server(80);

float temp_umbral = 28.5;

void setup() {
  Serial.begin(9600);

  dht.begin();
  pinMode(DHTPIN, INPUT);
  pinMode(RELAYPIN, OUTPUT);
  digitalWrite(RELAYPIN, LOW);

  Ethernet.begin(mac);

  Serial.print("IP asignada: ");
  Serial.println(Ethernet.localIP());

  // Recurso para enviar datos a Thinger.io
  thing["dht11"] >> [](pson& out){
    out["Temperatura"] = dht.readTemperature();
    out["Humedad"] = dht.readHumidity();
  };
```

```
  // Recurso para controlar el relay
  thing["relay"] << [](pson& in){
    if(in.is_empty()){
      in = (bool)digitalRead(RELAYPIN);
    }
    else{
      digitalWrite(RELAYPIN, in ? HIGH : LOW);
    }
  };
}

void loop() {
  // Mantenemos abierta la conexión con Thinger.io
  thing.handle();

  // Control de temperatura
  static unsigned long ultimaRevision = 0;
  if(millis() - ultimaRevision >= 10000){
    ultimaRevision = millis();

    float t = dht.readTemperature();
    if(t > temp_umbral){
    pson data;
      data["temp_umbral"] = temp_umbral;
      data["temperatura"] = t;
      thing.call_endpoint("endpoint_practica4", data);
      Serial.println(F("Correo enviado"));
    }
  }
   // Hacemos una pausa entre lecturas
  delay(1000);
}
```

Importante: de nuevo, para poder ver los valores y dashboards actualizados debemos conectar tanto la placa como el ordenador a (en este caso) la web de la universidad empleando los cables ethernet que hay en la clase.

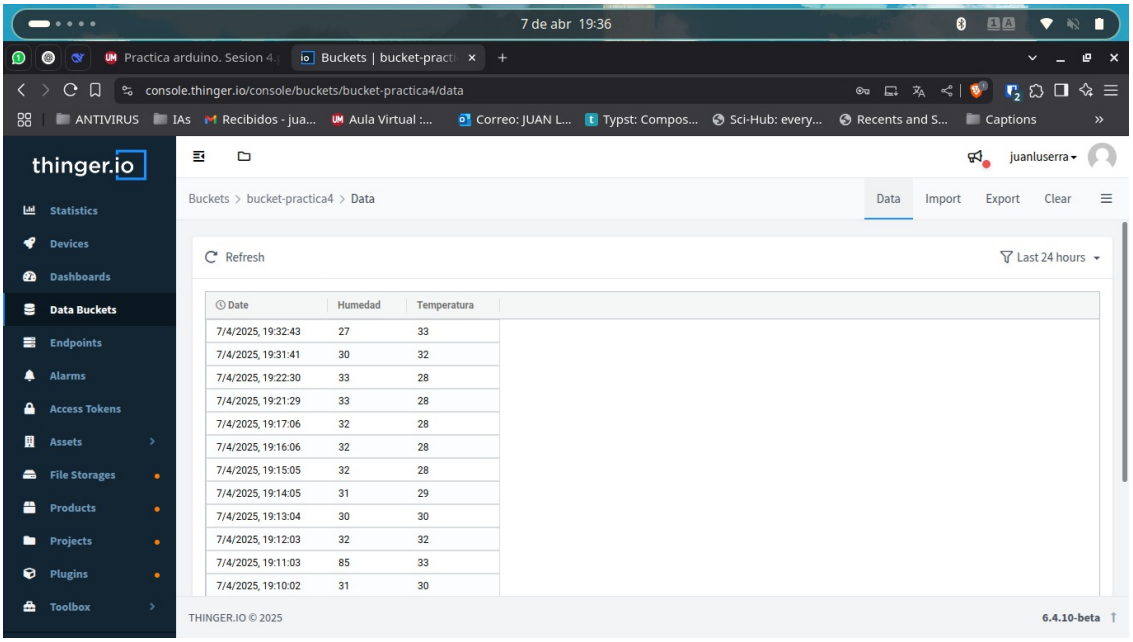
4.4. Dashboard configurado

Este dashboard se compone, como se pide en el enunciado, de tres widgets: una gráfica que muestra la evolución de la temperatura, otra que muestra de la misma forma la evolución de la humedad y un selector para encender y apagar el relé (es decir, el ventilador). El dashboard se muestra en la siguiente imagen:



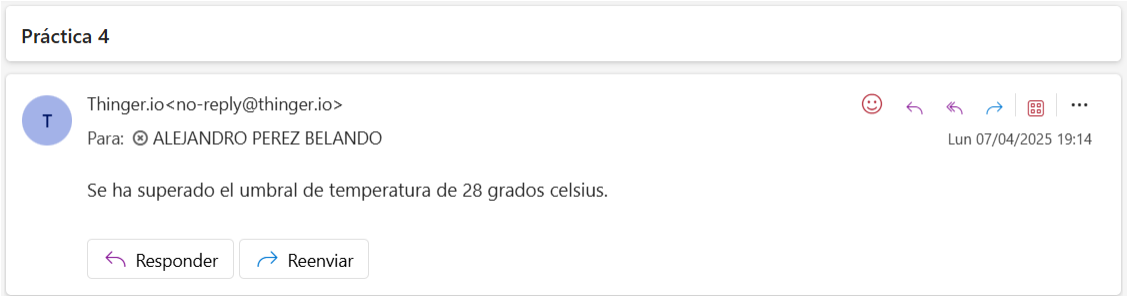
4.5. Cubo de datos

Se ha configurado correctamente el cubo de datos que debe almacenar los valores de temperatura y humedad. Estos valores almacenados se ven reflejados en la siguiente imagen:



4.6. End point

Se ha creado y configurado el punto final desde la plataforma de thinger.io en formato de e-mail. El correo al que se envía el mensaje cuando la temperatura supere un umbral determinado se envía en este caso a alejandro.p.b1@um.es. Una muestra del mensaje configurable es:



4.7. App Android

En la app para Android se muestra un dashboard similar al visto anteriormente pero en formato de aplicación. La siguiente imagen muestra cómo ha quedado la app: