Materia: Arq. de Com. De Sist. Emb.
Prof. Leonardo Castillo
Estudiante: Colque Chura, Daniela

Trabajo Final

MINI ESTACIÓN METEREOLÓGICA CON ESP32

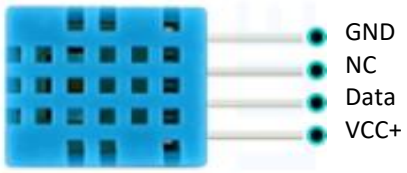
Propuesta: Armar una estación metereológica mini en Arduino IDE usando el microcontrolador ESP32 para medir humedad y temperatura, mostrando los datos en 2 dispositivos: una pantalla oled y mediante una aplicación en el celular.

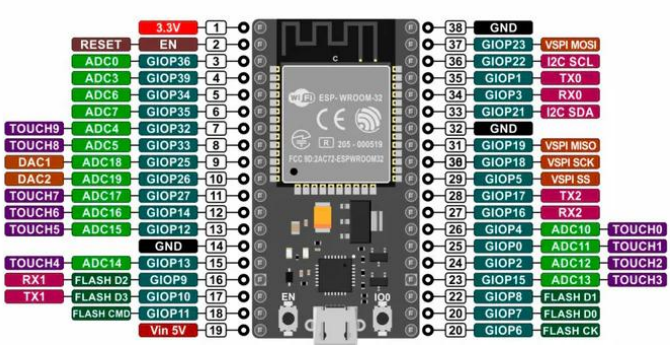
Materiales Usado:

- ✓ Micro ESP32S
- ✓ Pantalla Oled
- ✓ Sensor de humedad/temperatura DHT11
  Rango de Temperatura: 0~ 50°C (+/-2°C)
  Rango de Humedad: 20~ 90% (+/-5%)
- ✓ Protoboard 830 puntos
- ✓ LCD 16x2 por I2C
  Recordar:

LCD 16X2: 16 columnas (caracteres por línea) y 2 filas.
I2C: este módulo se usa para controlar un LCD Alfanumérico, con solo 4 pines (GND, VCC, SDA, SCL).

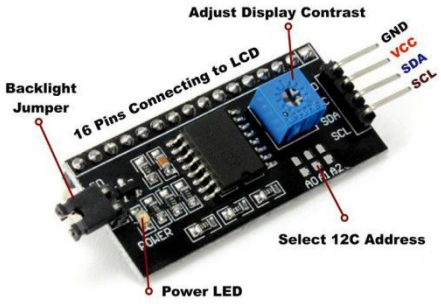
- ✓ Cables dupont m-h, m-m (varios)
- ✓ Cable USB A→Micro-USB para alimentación y programación.





Librerías Necesarias:

Función	Gestor de Librerías
DHT11	DHT sensor library <DHT.h>
Bus I²C (para DHT interno)	<Adafruit Unified Sensor>
LCD 16×2 I²C	<LiquidCrystal_I2C.h>
WI-FI	Incluida con el core ESP32
ThingSpeak (HTTP)	<ThingSpeak>
HTTPClient (para TS)	Incluída con el core ESP32



Conexiones y voltajes:

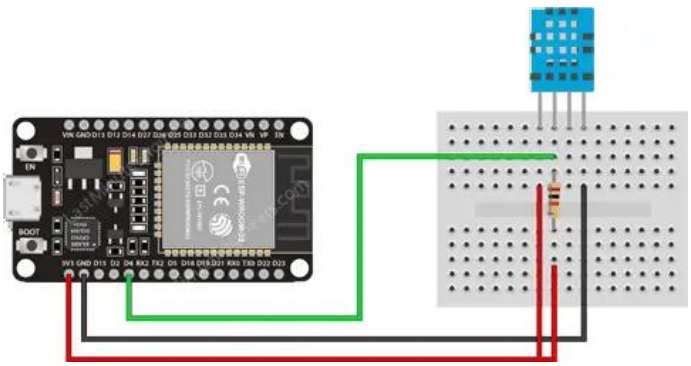
Dispositivo	Pin ESP32	Pin DHT11	VCC
ESP32			USB (5V)
DHT11	GPIO 4	DATA	3.3 V
LCD 16×2 I²C	GPIO 21 (SDA)	SDA	3.3V o 5V
	GPIO 22 (SCL)	SCL	

(Opcional Elegido) Para que el ESP32 esté en **Modo bootloader** y evitar hacer el 'juego' entre botones EN/RESET y BOOT. Se conectó un jumper entre GPIO0 a GND.

PRUEBA 1: **Lectura de DHT11**

<u>Conexiones</u>

- ✓ DHT11 VCC → ESP32 3.3 V
- ✓ DHT11 GND → ESP32 GND
- ✓ DHT11 DATA → ESP32 GPIO 4



Código de prueba

```
#include "DHT.h"
#define DHTPIN 4     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  Serial.println(F("DHT11 test!"));
  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);
```

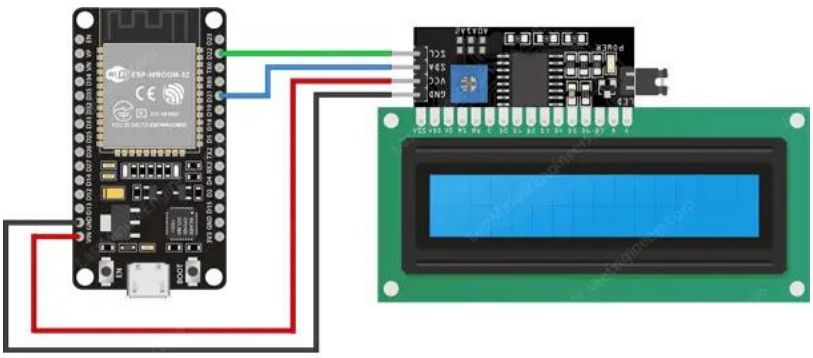
```
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
```

```
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  } else {
    // Línea 1: temperatura
    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.println(" °C ");
    // Línea 2: humedad
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.println(" % ");
    // Línea 3: separador
    Serial.println("-----------------------");// línea en blanco opcional entre lecturas
  }
}
```

PRUEBA 2: **Lectura de LCD por I2C**

Conexiones

- ✓ I2C SDA → ESP32 GPIO21 (SDA)
- ✓ I2C SCL → ESP32 GPIO22 (SCL)
- ✓ I2C GND → ESP32 GND
- ✓ I2C VCC → ESP32 Vin 5 V



Código de prueba

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Set the LCD address to 0x27 for a 16 chars and 2 line display
//Ajusta la dirección I2C (0x27 o 0x3F) según tu módulo.
LiquidCrystal_I2C lcd(0x3F, 16, 2);
void setup()
{
  // initialize the LCD
  lcd.begin();
  // Turn on the blacklight and print a message.
  lcd.backlight();
  lcd.print(""Hola ESP32!");
}
```

```
void loop()
{
  // Do nothing here...
}
```

Código para conectar DHT11 y I2C

```
#include "DHT.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

```
#define DHTPIN 4
#define DHTTYPE DHT11   // DHT 11
// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x3F, 16, 2);
DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {
  //LCD por I2C
  lcd.begin();
  lcd.backlight();
  //DHT11
  Serial.begin(115200);
  dht.begin();
}
```

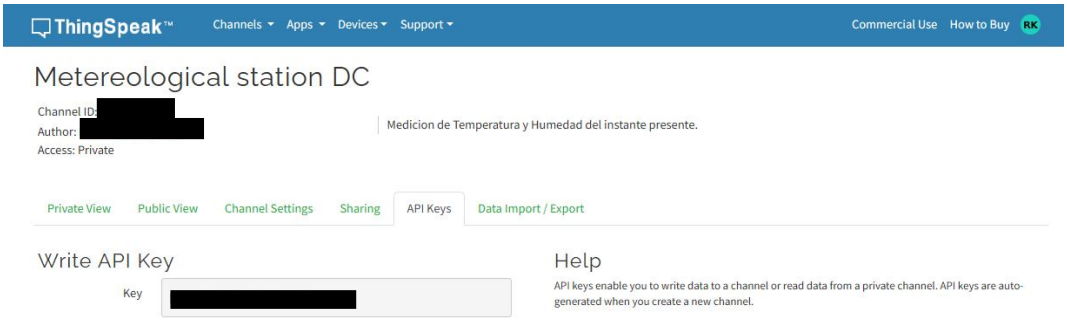
```
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  } else {
    // Línea 1: temperatura
```

```
    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.println(" °C ");
    //-----LCD por I2C
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Temp: ");
    lcd.print(t);
    lcd.print(" C");
    // Línea 2: humedad
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.println(" % ");
    //-----LCD por I2C
    lcd.setCursor(0,1);
    lcd.print("Hum:  ");
    lcd.print(h);
    lcd.print(" %");
    // Línea 3: separador
    Serial.println("-----------------------");// línea en blanco opcional entre lecturas
  }
}
```

PRUEBA 4: **Envío de datos a ThingSpeak**

Preparación en ThingSpeak

- ✓ Regístrarse en https://thingspeak.com
- ✓ Creo dos campos: “Temperatura” y “Humedad”.
- ✓ Copiar el Channel ID y WRITE API KEY, que utilizaremos en nuestro código en el IDE de Arduino.



Código para conectar DHT11 , I2C y ThinkSpeak

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHT.h"
#include <WiFi.h>
#include <WiFiClient.h>
#include <ThingSpeak.h>

// --- Configuración DHT11 ---
#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

```
// --- Configuración LCD I2C ---
LiquidCrystal_I2C lcd(0x3F, 16, 2);
// --- Configuración Wi-Fi y ThingSpeak ---
/*Definimos como constantes Char las credenciales de acceso a la red WiFi*/
const char* ssid     = "AVC650451";
const char* password = "95717466";
unsigned long channelID = 2995140;       // tu Channel ID
const char* writeAPIKey = "FWDSJKWX4N14WWBC";  // tu Write API Key
//Definimos el cliente WiFi que usaremos
WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(1000);
  //Inicia DHT11
  /*Imprimimos una frase, e iniciamos nuestro sensor DHT*/
  Serial.println("Sensores Instalados y listos");
  dht.begin();
  //Inicia LCD
  Wire.begin(21, 22);  //Inicializa el bus I2C en los pines por defecto del ESP32
  lcd.begin();
  lcd.backlight();
  // Conecta Wi-Fi
  WiFi.begin(ssid, password);
  /*Iniciamos la conexión a la red WiFi, y se imprimirán caracteres indicando el tiempo que tarda la
conexión*/
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  /*Una vez conextado, se imprimirá una frase y se iniciará la conexión a la Plataforma usando el cliente
definido anteriormente*/
  Serial.println("WiFi conectado");
  //Inicia ThinkSpeak
  ThingSpeak.begin(client);
  delay(500);
}
```

```
void loop(){
  /*Usamos un retardo de 5 segundos, y utilizamos la función Medición para la lectura de los sensores*/
  delay(5000);
  medicion();
  /*Hacemos la conexión y envío de datos a la plataforma, utilizando las credenciales definidas
anteriormente*/
  ThingSpeak.writeFields(channelID,writeAPIKey);
  /*Imprimimos una frase indicando el envío, y agregamos un retardo de 10 segundos*/
  Serial.println("Datos enviados a ThingSpeak!");
  delay(10000);
}
```

```
/*Definimos la función Medición*/
void medicion() {
```

```
  lcd.clear();
  //Lee sensor
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  /*Imprimimos los valores obtenidos en el terminal Serial*/
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" °C  ");
  //-----LCD por I2C
  //lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temp: ");
  lcd.print(t);
  lcd.print(" C  ");
  Serial.print("Humedad: ");
  Serial.print(h);
  Serial.println(" %");
  //-----LCD por I2C
  lcd.setCursor(0,1);
  lcd.print("Hum:  ");
  lcd.print(h);
  lcd.print(" %");
  Serial.println("----------------------------------------");
  /*Indicamos el orden de envío por campos o Field, en el orden definido de la plataforma, junto a los
valores del sensor*/
  ThingSpeak.setField(1,t);
  ThingSpeak.setField(2,h);
}
```

La explicación del código se muestra a continuación:

Como primer paso (y muy importante, por cierto), debemos llamar a las librerías WiFi.h, ThingSpeak.h y DHT.h, instaladas en los incisos anteriores.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <ThingSpeak.h>
```

Ahora definimos que el pin de Datos del sensor estará conectado al pin 4 del ESP32, y que además, estamos usando el sensor DHT11.

```
// --- Configuración DHT11 ---
#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

Definimos como constantes char nuestras credenciales de la red WiFi a la cual se conectará la tarjeta ESP32 DevKit V1.

```
const char* ssid     = "          ";
const char* password = "          ";
```

Definimos las variables channelID y WriteAPIKey, para la conexión a la plataforma. Ambos los obtuvimos en el inciso anterior a la programación.

```
unsigned long channelID = [redacted];       // tu Channel ID
const char* writeAPIKey = "[redacted]";  // tu Write API Key
```

Se define además que el cliente WiFi el cual usaremos, se denominará «cliente»

/*Definimos el cliente WiFi que usaremos*/

```
WiFiClient client;
```

Iniciamos la **función Setup**; iniciamos el terminal Serial a una velocidad de 115200 junto a un retardo de 1 segundo.

/*Iniciamos la función Setup()*/

```
void setup() {
  Serial.begin(115200);
  delay(1000);
```

Pasamos a imprimir una frase e iniciamos nuestro sensor DHT11

```
  //Inicia DHT11
  /*Imprimimos una frase, e iniciamos nuestro sensor DHT*/
  Serial.println("Sensores Instalados y listos");
  dht.begin();
```

Inicializamos la pantalla LCD

```
 //Inicia LCD
  Wire.begin(21, 22);  //Inicializa el bus I2C en los pines por defecto del ESP32
  lcd.begin();
  lcd.backlight();
```

Iniciamos la conexión a la red WiFi, y mientras esta se realiza, se mostrarán caracteres indicando la conexión.

/*Iniciamos la conexión a la red WiFi, y se imprimirán caracteres indicando el tiempo que tarda la conexión*/

```
// Conecta Wi-Fi
  WiFi.begin(ssid, password);
  /*Iniciamos la conexión a la red WiFi, y se imprimirán caracteres indicando el tiempo que tarda la
conexión*/
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

Realizado la conexión, se mostrará la frase «Conectado a WiFi», y se iniciará la conexión a la plataforma usando el cliente WiFi definido líneas arriba, todo seguido de un retardo de 5 segundos.

```
Serial.println("WiFi conectado");
```

/*Una vez conectado y se iniciará la conexión a la Plataforma usando el cliente definido anteriormente*/

```
 //Inicia ThinkSpeak
  ThingSpeak.begin(client);
  delay(500);
}
```

Iniciamos la **función Loop()**, seguido de un retardo de 5 segundos y utilizamos la función Medición para la lectura de los valores del sensor.

/*Iniciamos la función Loop*/

```
void loop(){
  /*Usamos un retardo de 5 segundos, y utilizamos la función Medición para la lectura de los sensores*/
```

```
  delay(5000);
  medicion();
```

Realizamos la conexión y envío de datos a la plataforma usando las credenciales definidas líneas arriba.

```
/*Hacemos la conexión y envío de datos a la plataforma, utilizando las credenciales definidas anteriormente*/
  ThingSpeak.writeFields(channelID,writeAPIKey);
```

Imprimimos una frase indicando el envío, y agregamos un retardo de 10 segundos.

```
Serial.println("Datos enviados a ThingSpeak!");
  delay(10000);
}
```

Procedemos a indicar lo que se realiza en la función medicion(), donde se procede a hacer la lectura de Temperatura y Humedad del sensor.

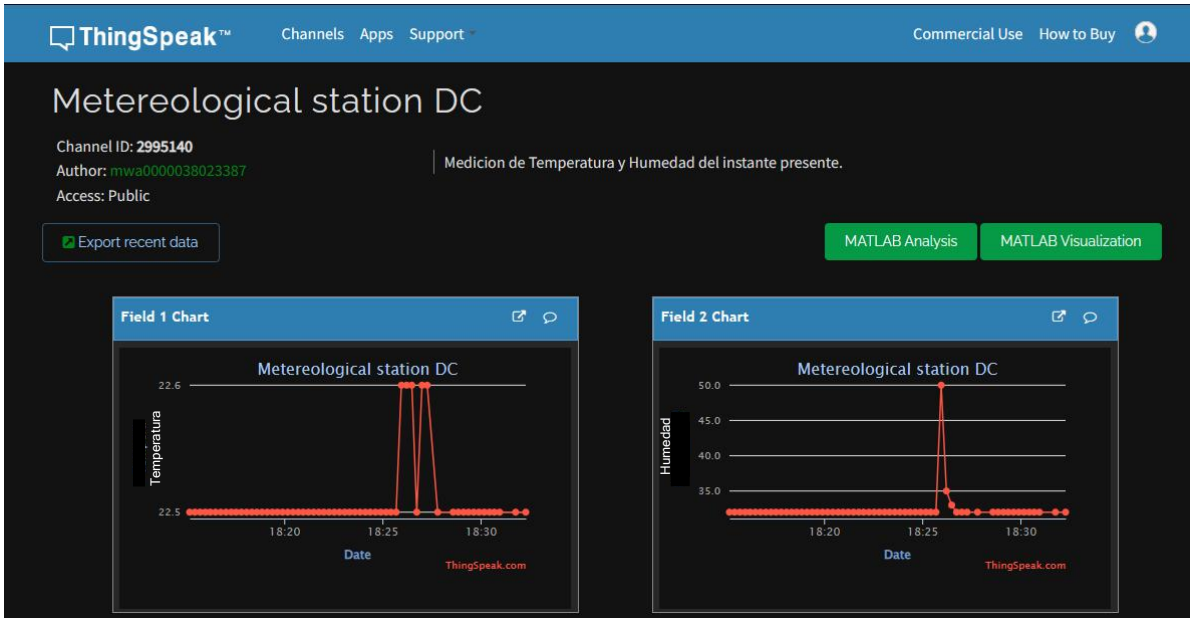
```
/*Definimos la función Medición*/
void medicion() {
  lcd.clear();
  //Lee sensor
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  /*Imprimimos los valores obtenidos en el terminal Serial*/
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" °C  ");
  //-----LCD por I2C
  //lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temp: ");
  lcd.print(t);
  lcd.print(" C  ");

  Serial.print("Humedad: ");
  Serial.print(h);
  Serial.println(" %");
  //-----LCD por I2C
  lcd.setCursor(0,1);
  lcd.print("Hum:  ");
  lcd.print(h);
  lcd.print(" %");
  Serial.println("----------------------------------------");
```

Indicamos el orden de los campos / Field, como lo realizamos al momento de crear el canal en la plataforma, junto a las variables del sensor.

```
  /*Indicamos el orden de envío por campos o Field, en el orden definido de la plataforma, junto a los valores del sensor*/
  ThingSpeak.setField(1,t);
  ThingSpeak.setField(2,h);
}
```

Para revisar los datos de temperatura y humedad, se puede revisar la siguiente página web:
https://thingspeak.mathworks.com/channels/2995140



En este proyecto se logró:

- ✓ Tener una estación IoT completa
- ✓ Medir temperatura y humedad.
- ✓ Mostrar localmente en el LCD.
- ✓ Publicar en tiempo real en ThingSpeak para monitorización remota.

Bibliografía:

- https://www.taloselectronics.com/blogs/tutoriales/programar-esp32-con-ide-arduino
- https://naylampmechatronics.com/blog/35_tutorial-lcd-con-i2c-controla-un-lcd-con-solo-dos-pines.html
- https://todomaker.com/blog/envio-de-datos-a-thingspeak-usando-esp32/