Proyecto Arduino: Estación Ambiental Inteligente

GRUPO 25: Ignacio Alzabé, Donato Doebbeling, Ulises Quiñones, Juan Manuel Servadei

Prof. Martín Salamero

Universidad Blas Pascal

14 de noviembre de 2025

ÍNDICE

INTRODUCCIÓN

Nuestro objetivo de todo el semestre fue crear una estación ambiental propia, con código propio y aprendiendo paso a paso en el camino. Al igual que los demás grupos, los materiales con los que iniciamos fueron:

- Arduino UNO
- Protoboard
- Cables macho-macho, hembra-hembra, macho-hembra
- Fotorresistor LDR
- Sensor DHT11 (Humedad y Temperatura)
- Sensor opcional de humedad de la tierra
- Clock RTC DS1302
- Resistencias de 1k ohm
- Pantalla LCD I2C 16x2
- Lector de tarjetas SD
- Módulo Bluetooth HC-05

Nuestra plataforma de organización fue Microsoft Teams, donde fuimos escribiendo nuestras bitácoras día a día y en los Mensajes Directos están todas nuestras iteraciones de código, hasta las más mínimas.

En el tiempo dado, pudimos armar, codificar, probar y finalizar un prototipo completo de una Estacion Ambiental Inteligente, donde se registran datos como: Temperatura, Humedad del suelo y de la tierra (donde sea que se lo plante al sensor), luminosidad, y grabar todos los datos en una tarjeta SD con una duración estimada de 5 años de almacenamiento.

DESARROLLO

Información de sensores

Fotorresistor LDR

Su nombre proviene del inglés y significa resistor dependiente de la luz. Esto significa que el valor de la resistencia depende de la luz que esté recibiendo. Gracias a esto, se puede transformar con Arduino IDE el valor que recibe a un porcentaje para medir la luz en un ambiente, por ejemplo. También puede usarse para detectar cuando pasan personas por una puerta. Tienen solo dos patas, por lo que deben conectarse en serie con otras resistencias.

Sensor DHT11

Es uno de los sensores más usados y conocidos para trabajar con Arduino. Tiene una alta estabilidad y fiabilidad, pero una de sus desventajas es que sus patas son muy finas. Esto puede llevar a problemas en la señal si no se conectan adecuadamente. Mide temperatura y humedad del aire, y transmite ambos datos por pin analógico. Tiene una versión con PCB, que integra una resistencia pull up.
Necesita una librería para configurarlo.

Sensor de humedad del suelo

Comprado en MercadoLibre, tiene dos electrodos y un módulo con led de conexión y de alta humedad integrado. Funciona midiendo la capacidad de la tierra para transmitir electricidad, y gracias a eso, calcular el agua presente en el suelo y convertirlo a voltaje. Está conectado a una salida analógica.

Reloj RTC DS1302

Tiene la capacidad de llevar el tiempo con el poder desconectado gracias a una pila CR2025 o CR2032. Necesita una librería para configurarlo.

Pantalla I2C 16x2

Muestra todos los datos de los sensores y la fecha y hora en el Arduino. Tiene un módulo i2c conectado detrás para reducir la cantidad de pines necesarios. En nuestro caso particular, la dirección donde se encuentra es 0x26, pero la más común es 0x27.

Lector de tarjetas SD

Gracias a la librería SD de arduino, puede cargar los datos de los sensores en un archivo csv cada cierta cantidad de tiempo. En nuestro caso, cada minuto.

Módulo Bluetooth HC-05

Módulo con SLAVE y MASTER, usado para transmitir datos desde el Arduino hasta nuestra app para celulares creada en Mit App Inventor.

Primeras versiones del código

Lo primero fue la investigación; aplicando lo que nos habían enseñado hasta ahora, conectamos los sensores a los cables y estos a los distintos puertos del Arduino. Revisamos en internet, y en páginas de los sensores, distintos códigos de prueba para verificar el funcionamiento de cada uno de los mismos.
La primera dificultad que nos encontramos fue cuando detectamos que el DHT11 no funcionaba, creímos que fue porque estaba defectuoso o dañado, por lo que fue reemplazado por otro comprado fuera del laboratorio. Este nuevo sensor, a diferencia del anterior, requería dos resistencias externas para funcionar, que agregamos en la protoboard para realizar la señal.

Con el nuevo sensor, escribimos un código inicial que hacía esto:

- Iniciaba el LCD y el sensor DHT11
- Leía los valores de temperatura y humedad cada 5 segundos.
- Mostraba alternativamente en el LCD dichos valores, con mensajes simples
- Implementa una verificación de errores para evitar mostrar valores inválidos (isnan)

Este prototipo nos permitió validar que el Arduino pudiese interactuar correctamente con el sensor y el display.

Código inicial

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

#define DHTPIN 5
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal_I2C lcd(0x26, 16, 2);

void setup() {
  lcd.init();
  lcd.backlight();
  dht.begin();
}

void loop() {
```

```
  delay(5000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    lcd.print("Error obteniendo los datos del sensor DHT11");
    return;
  }

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Humedad:");
  lcd.setCursor(10, 0);
  lcd.print(h);
  lcd.print("%");
  delay(5000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp:");
  lcd.setCursor(10, 1);
  lcd.print(t);
  lcd.print(" C");
  delay(5000);
}
```

Explicacion del codigo

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

- Wire.h : permite la comunicación 12C entre el Arduino y otros dispositivos
- LiquidCrystal_I2C.h : Controla la pantalla LCD utilizando solo dos pines mediante I2C.
- DHT.h : Permite leer datos del sensor DHT11

Definición del sensor:

```
#define DHTPIN 5
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

- Acá se define el pin digital 5 como el pin de datos del sensor.

- Se indica que se está utilizando un DHT11
- Se crea un objeto “dht” para manejar las lecturas del sensor

Inicializacion del display y el sensor:

```
LiquidCrystal_I2C lcd(0x26, 16, 2);

void setup() {
  lcd.init();          // Inicializa la pantalla
  lcd.backlight();     // Enciende la luz de fondo del LCD
  dht.begin();         // Inicia el sensor DHT11
}
```

- El LCD tiene dirección “0x26” y es de 2 filas por 16 columnas.
- En setup() se inicializan ambos periféricos para que estén listos para usarse

Lógica principal (Loop):

```
void loop() {
  delay(5000);   // Espera 5 segundos entre cada ciclo
```

- Se utiliza una espera inicial para evitar lecturas demasiado seguidas

```
  float h = dht.readHumidity();        // Lee la humedad (%)
  float t = dht.readTemperature();     // Lee la temperatura (°C)
```

- Las lecturas se almacenan como variables Float para poder mostrar los decimales

```
  if (isnan(h) || isnan(t)) {
    lcd.print("Error obteniendo los datos del sensor DHT11");
    return;
  }
```

- Si alguna lectura falla, se muestra un mensaje de error y se interrumpe el ciclo actual.

```
  float hic = dht.computeHeatIndex(t, h);
```

- A pesar de que no se muestre en pantalla, se calcula el índice de calor usando la temperatura y la humedad

Mostrar humedad:

```
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Humedad:");
  lcd.setCursor(10, 0);
  lcd.print(h);
  lcd.print("%");
  delay(5000);
```

- Limpia la pantalla, escribe “Humedad:” y luego muestra el valor leído seguido de “%”
- Espera 5 segundos antes de cambiar la pantalla

Mostrar temperatura:

```
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp:");
  lcd.setCursor(10, 1);
  lcd.print(t);
  lcd.print(" C");
  delay(5000);
}
```

- Limpia la pantalla, muestra “Temp:” y el valor de temperatura seguido de “°C”
- Espera 5 segundos antes de reiniciar el bucle

Etapa 2

Ese mismo día, avanzamos hubo una mejora del código. Incorporaba un sensor de luz analogico (LDR) conectado al pin A0

En esta versión del código se mantuvo el DHT11 para leer la temperatura y humedad, se agregó una lectura analógica desde el LDR, usando “analogRead(A0)”. Se implementó la función map() para convertir el valor leído (entre 0 y 1023) Se programó que el LCD muestre tres variables secuencialmente (Humedad, Temperatura y Nivel de luz)

Esta versión fue importante porque empezamos a implementar múltiples sensores y a experimentar con la secuencia de visualización en pantalla.

Segundo código

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

#define DHTPIN 5
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
const int sensorluz = A0;

LiquidCrystal_I2C lcd(0x26, 16, 2);

void setup() {
  lcd.init();
  lcd.backlight();
  dht.begin();
}

void loop() {
  delay(1000);

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  int valorluz = analogRead(sensorluz);
  int porcentajeluz = map(valorluz, 0, 1023, 0,100);

  if (isnan(h) || isnan(t)) {
    lcd.print("Error obteniendo los datos del sensor DHT11");
    return;
  }

  float hic = dht.computeHeatIndex(t, h);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Humedad:");
  lcd.setCursor(10, 0);
  lcd.print(h);
  lcd.print("%");
  delay(5000);

  lcd.clear();
  lcd.setCursor(0, 0);
```

```
  lcd.print("Temp:");
  lcd.setCursor(10, 0);
  lcd.print(t);
  lcd.print(" C");
  delay(5000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Luz:");
  lcd.setCursor(10, 0);
  lcd.print(porcentajeluz);
  lcd.print("%");
  delay(1000);
}
```

Explicacion del codigo (Solo lo añadido)

```
int valorluz = analogRead(sensorluz);
```

- Se realiza una lectura analogica desde el pin A0, que entregas valores entre 0 y 1023

```
int porcentajeluz = map(valorluz, 0, 1023, 0, 100);
```

- Se usa la función map() para transformar la lectura a un porcentaje inverso
  - 0%: máxima oscuridad
  - 100%: máxima luz

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Luz:");
lcd.setCursor(10, 0);
lcd.print(porcentajeluz);
lcd.print("%");
delay(1000);
```

- Se agregó una tercera pantalla al ciclo de visualización
- Muestra el porcentaje de luz y luego se espera 1 segundo antes de reiniciar el bucle

Etapa 3

En esta nueva versión del código, se incorporó el modulo RTC DS1302, que nos permitió disponer de una referencia horaria confiable y permanente, independientemente del estado de alimentación del Arduino.

Además, reorganizamos completamente la forma en el que se muestran los datos en la pantalla LCD, dividiéndose en dos grupos de variables, por un lado las variables de los sensores y por el otro la fecha y hora , que estos se muestran en la pantalla LCD y alternan sus valores cada 5 segundos.

Código tres, RTC y organización

```
// BETA ESTACION AMBIENTAL

#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#include <DHT.h>

#include <RtcDS1302.h>


#define DHTPIN 5

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);


const int sensorluz = A0;

const int sensorhumt = A2;


LiquidCrystal_I2C lcd(0x26, 16, 2);

ThreeWire myWire(4, 5, 2); // IO, SCLK, CE

RtcDS1302<ThreeWire> Rtc(myWire);


// ---- control de tiempos ----
```

```
unsigned long previousMillisRTC = 0;
unsigned long previousMillisSensors = 0;
const unsigned long intervalRTC = 1000;      // refrescar hora cada 1s
const unsigned long intervalSensors = 5000;  // alternar sensores cada 5s

bool mostrarGrupo1 = true;

void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
  dht.begin();
  delay(2000);

  // Inicializar el RTC
  Rtc.Begin();
  delay(1000);

  // Setear fecha/hora de compilación (solo como ejemplo)
  RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
  Rtc.SetDateTime(compiled);

  if (Rtc.GetIsWriteProtected()) {
    Rtc.SetIsWriteProtected(false);
  }
```

```
}

void loop() {
  unsigned long currentMillis = millis();

  // --- actualizar hora cada 1s ---
  if (currentMillis - previousMillisRTC >= intervalRTC) {
    previousMillisRTC = currentMillis;

    RtcDateTime now = Rtc.GetDateTime();
    printTime_LCD(now);
  }

  // --- alternar sensores cada 5s ---
  if (currentMillis - previousMillisSensors >= intervalSensors) {
    previousMillisSensors = currentMillis;

    // limpiar primera línea
    lcd.setCursor(0, 0);
    lcd.print("                ");

    if (mostrarGrupo1) {
      // Grupo 1: humedad suelo + luz
      int valorluz = analogRead(sensorluz);
      int porcentajeluz = map(valorluz, 0, 1023, 0, 100);
```

```
  int valorhumedadt = analogRead(sensorhumt);
  int porcentaje hunt = map(valorhumedadt, 365, 1023, 100, 0);

  lcd.setCursor(0, 0);
  lcd.print("HS:");
  lcd.print(porcentajehumt);
  lcd.print("%");

  lcd.setCursor(7, 0);
  lcd.print("Luz:");
  lcd.print(porcentajeluz);
  lcd.print("%");

} else {
  // Grupo 2: temperatura + humedad aire
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print((int)t); // entero para que entre
  lcd.print((char)223); // símbolo de grados °
  lcd.print("C");
```

```
      lcd.setCursor(8, 0);
      lcd.print("HA:");
      lcd.print((int)h);
      lcd.print("%");
    }

    mostrarGrupo1 = !mostrarGrupo1; // alternar
  }
}

void printTime_LCD(const RtcDateTime& dt) {
  char datestring[9]; // HH:MM:SS
  snprintf_P(datestring, sizeof(datestring),
             PSTR("%02u:%02u:%02u"),
             dt.Hour(), dt.Minute(), dt.Second());

  lcd.setCursor(7, 1);
  lcd.print(datestring);
}
```

Explicacion del codigo, solo lo nuevo:

#include <RtcDS1302.h>

- Se utiliza la librería RtcDs1302.h

```
ThreeWire myWire(4, 5, 2); // IO, SCLK, CE
RtcDS1302<ThreeWire> Rtc(myWire);
```

- Se utiliza la interfaz “TrheeWire” para manejar el reloj de tiempo real.

RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
Rtc.SetDateTime(compiled);

- En el setup(), se carga la fecha/hora de compilación como valor inicial
- Esto se hace una sola vez para sincronizar el reloj interno, luego queda funcionando de forma autónoma

printTime_LCD(now);

- Cada segundo, se actualiza el reloj usando esa línea
- El tiempo se muestra en la segunda fila del LCD con el formato HH:MM:SS

Ciclo de visualización alternado

bool mostrarGrupo1 = true;

- Se alternan dos grupos de datos en la primera fila de LCD, cada 5 segundos:
    - Grupo 1:
        - HS: Humedad del suelo
        - Luz: Nivel de luz
    - Grupo 2:
        - T: Temperatura ambiente
        - HA: Humedad ambiente
- Se utilizó una bandera booleana “mostrarGrupo1” para alternar entre ambos grupos

Este diseño simplifica la interfaz y permite mostrar mas informacion con un solo display de 16x2 sin saturar

Control por temporizadores:

if (currentMillis - previousMillisRTC >= intervalRTC)

- Se reemplazó el uso de delau() por un control manual del tiempo con millis(), lo que permite ejecutar tareas en paralelo sin bloquear el programa
- Esto fue clave para poder actualizar sensores y reloj a diferentes intervalos sin conflictos.

Con esta versión logramos una visualización organizada. Integramos el RTC DS1302 que nos permitió que el proyecto de un salto de calidad: ahora puede registrar eventos o variables con una referencia temporal real, esencial para la futura integración con la tarjeta SD y el registro de datos históricos.

Etapa 4

En esta etapa del desarrollo, implementamos el sensor de humedad y avanzamos hacia uno de los objetivos fundamentales del proyecto: registrar los datos medidos en archivos CSV, almacenados en una tarjeta microSD, con su respectiva marca temporal obtenida del módulo RTC.

Este paso permitió que la estación ambiental ya no solo mostrará datos en tiempo real, sino que pudiera registrarlos de forma histórica para su posterior análisis.

Agregados nuevos

Inicialización del módulo SD

```
#include <SD.h>
#include <SPI.h>
...
if (!SD.begin(10)) {
  Serial.println("Fallo al inicializar SD.");
  return;
}
```

- Se utilizó la librería SD.h para manejar el almacenamiento
- El módulo se inicializa en el pin 10
- Si la tarjeta no es detectada correctamente, el programa muestra un mensaje aborta la operación

Creación del nombre de archivo

```
sprintf(nombreArchivo, "LOG_%04u%02u%02u.CSV", ahora.Year(), ahora.Month(), ahora.Day());
```

- Se crea un archivo diferente por cada día de funcionamiento
- El nombre tiene el formato “LOG_AAAAMMDDD.CSV”

Formateo de timestamp con RTC

sprintf(tiempoBuffer, "%04u/%02u/%02u %02u:%02u:%02u", ...);

- Se construye una cadena con la fecha y hora actual.
- Se almacena en “tiempoBuffer” que luego se utiliza tanto para mostrar en LCD como para registrar en el archivo

Lectura de sensores y escrituras en SD

sprintf(cargardatos, "%s,%d,%d,%d,%d", tiempoBuffer, t, h, porcentajeluz, porcentajehumedadt);
myFile = SD.open(nombreArchivo, FILE_WRITE);

- Cada 60 segundos se actualizan las lecturas de los sensores
- Los valores se combinan con el timestamp en una línea CSV.
- La línea se escribe en el archivo correspondiente al día actual

Visualización dual mejorada

- Ahora cada 3 segundos la pantalla LCD alterna entre:
    - Pantalla 1: Muestra las cuatro variables
    - Pantalla 2: Muestra la fecha y la hora

Con esta versión alcanzamos un prototipo casi completo y autónomo, capaz de medir, mostrar y registrar datos ambientales de forma continua y confiable.

Etapa final, finalizacion del código

En esta versión final, se completó el ciclo completo del proyecto: Medir,mostrar,registrar y transmitir datos ambientales en tiempo real, de forma autónoma y continua. También, se hizo una aplicación en “MIT App Inventor” para que pueda recibir los datos que da el Arduino.

La principal novedad es la integración del módulo Bluetooth HC-05, que permite enviar los datos a un celular que tenga la aplicación que desarrollamos. Esto transforma la estación en un sistema de monitoreo en vivo.

Código final

Visible en repositorio de GIT

Lo nuevo

```
#include <SoftwareSerial.h>
SoftwareSerial BT(7, 8); // RX, TX

BT.begin(9600);
BT.println("HC-05 listo");
```

- Se utiliza la librería “SoftwareSerial” para crear un puerto serie adicional en los pines 7 y 8
- Al encender la estación, se envía un mensaje de bienvenida a la app
- Este mensaje inicial también sirve para comprobar que la conexión está activa

```
BT.println(cargardatos);
```

- Esto permite visualizar en tiempo real las variables ambientales sin tener que abrir el archivo o conectar el Arduino a una PC.
- Si ocurre un error al guardar la tarjeta SD, también se avisa por bluetooth

```
BT.println("ERROR_SD");
```

- Esto mejora la detección de fallas

Mejoras finales

Se implementó una función llamada “abrirArchivoNuevo()” que verifica si el archivo diario está abierto y lo reabre automáticamente si es necesario. También, se añadieron mecanismos de recuperación ante fallos en la tarjeta SD, como un reinicio del bus SPI, intentar abrir el archivo de nuevo y por último un Fallback con mensaje de error.

A lo largo de las distintas etapas, el código evolucionó desde una simple lectura de sensores hasta un sistema integrado de monitoreo ambiental inteligente, con funcionalidades:

- Sensor de temperatura y humedad del aire
- Sensor de luz
- Sensor de humedad del suelo

- Registro de datos en tarjeta SD
- Visualización cíclica en pantalla LCD
- Control de fecha/hora con RTC DS1302
- Comunicación Bluetooth en tiempo real

Se logró una estación autónoma capaz de operar más de 48 horas sin intervención.

CONCLUSIÓN

A lo largo del proyecto, nuestro grupo logró desarrollar una estación de monitoreo ambiental inteligente completamente funcional, capaz de operar de forma autónoma durante más de 48 horas. El proceso de desarrollo no fue lineal. Tuvimos que investigar sobre todo los sensores, como funcionan y cómo se conectan. Fue una prueba, error y corrección constante, y un fuerte trabajo colaborativo que fue moldeando el producto final.

Comenzamos con un código muy simple que solo leía temperatura y humedad del aire (Con el DHT11), y lo mostraba por la pantalla. A medida que fuimos investigando y aprendiendo de los componentes, fuimos incorporando nuevas funcionalidades claves, como la lectura de luz, humedad del suelo, gestión horaria con un reloj RTC, guardado automático de datos en archivos CSV y, finalmente, comunicación en tiempo real por Bluetooth hacia una aplicación móvil diseñada por nosotros mismos.

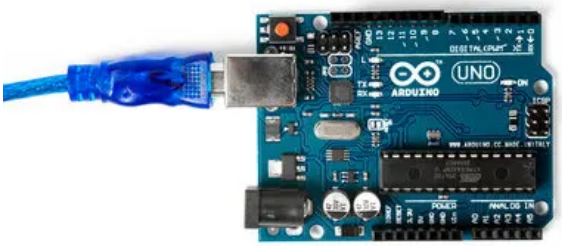
Además de integrar varios sensores y módulos distintos, aplicamos principios de programación eficientes, tales como el uso de millis() para evitar bloqueos, estructuras condicionales para la rotación de pantallas, verificación de errores en sensores y tarjetas SD, y funciones especificas como “abrirArcvhioNuevo()” para lograr un sistema robusto y tolerante a fallos.

El resultado es un dispositivo modular, portátil y confiable, capaz de medir en simultáneo cinco variables distintas, como la temperatura, humedad del aire, humedad del suelo, nivel de luz y fecha/hora. Todos estos datos se visualizan, registran y transmiten sin necesidad de una intervención humana.

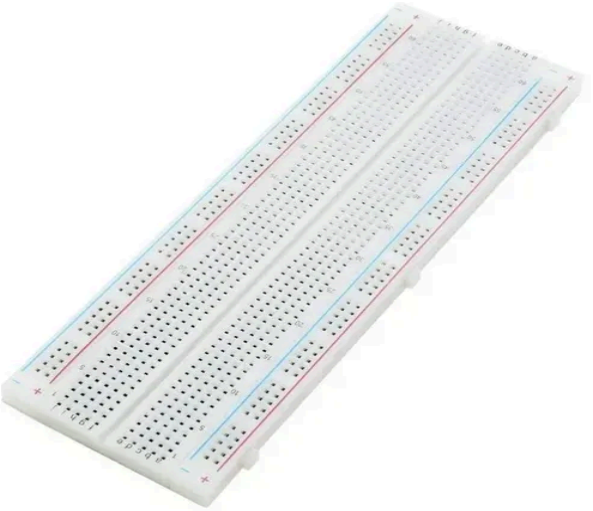
Este proyecto nos permitió aplicar lo aprendido en clase y enfrentarnos a problemas reales que nos obligaron a buscar soluciones, debatir ideas y trabajar como equipo. Más allá de lo técnico, fue una experiencia formativa que refuerza nuestras habilidades en programación, electrónica, diseño de sistemas y documentación.

ANEXO

- **Arduino UNO**



- **Protoboard**



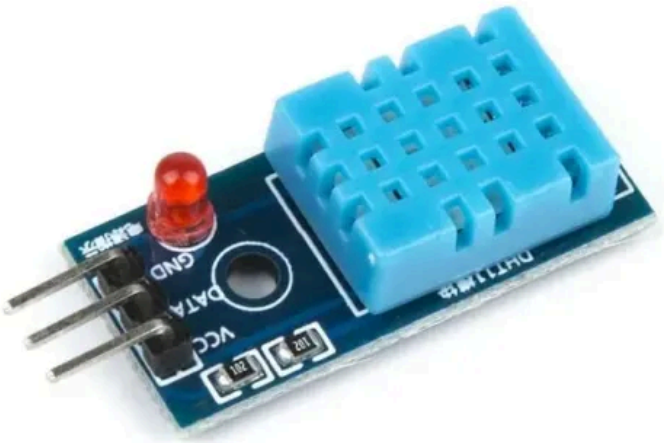
- **Cables Macho-Macho**

- **Cables Macho-Hembra**

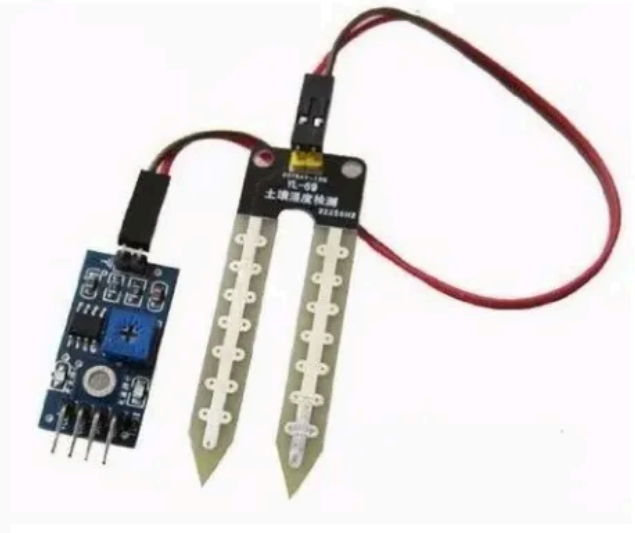
- **Cables Hembra-Hembra**

- Fotorresistor LDR

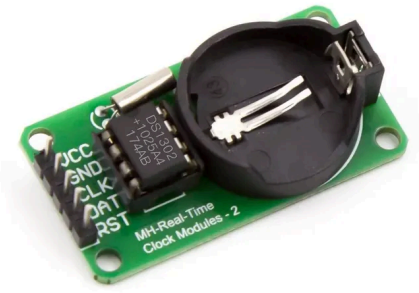
- Sensor DHT11 (Humedad y Temperatura)



- Sensor opcional de humedad de la tierra



- **Clock RTC DS1302**



- Resistencias de 1k ohm

- Pantalla LCD
  I2C 16x2

- Lector de tarjetas SD