Registro Diario de Avances – Sprint 1

**Fecha:01/10/2025**

**Autores: Pool K. Apaza, Erick K. Aguila, Victor E. Condori, Crystian L. García**

**Versión:** 1.0

# 1. Introducción

Este documento detalla el seguimiento diario del Sprint 1 del proyecto Invernadero Automatizado, centrado en el desarrollo de funcionalidades básicas. Se incluyen actividades diarias, tareas completadas, evidencias de código, un burndown chart y un análisis del rendimiento del equipo.

# 2. Lineamientos de Diseño

Desarrollar una primera versión funcional de la aplicación TaskManager que permita a los usuarios:

* Registrarse e iniciar sesión.
* Crear nuevas tareas.
* Marcar tareas como completadas.
* Almacenar los datos de manera persistente con localStorage.

# 3. Historias de Usuario y Tareas

| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- |
| HU-1.1 | Visualización en tiempo real | 3 pts | 17/09/2025 | 26/09/2025 |

| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| --- | --- | --- | --- | --- |
| 1.1.1 | Investigar sensores de humedad compatibles con Arduino | Pool | 4 | Concluido |
| 1.1.2 | Conectar y programar sensor de humedad | Victor | 6 | Concluido |
| 1.1.3 | Diseñar la interfaz de usuario para mostrar la lectura de humedad | Cristian | 5 | Concluido |
| 1.1.4 | Integrar la lectura del sensor a la interfaz | Erick | 6 | Concluido |
| 1.1.5 | Probar la visualización de datos en tiempo real | Victor | 4 | Concluido |
| 1.1.6 | Implementar alerta en caso de fallo del sensor | Erick | 3 | Pendiente |

## 

# 4. Evidencias de Investigación de Sensores

**Investigación: sensores de humedad compatibles con Arduino**

## 1. Objetivo y alcance

Objetivo: proporcionar una guía técnica y práctica (investigación) para elegir, conectar, calibrar y usar sensores de humedad de suelo y humedad ambiental (aire) con placas Arduino (UNO, Nano, Mega, ESP32, etc.).

Alcance: revisión de tipos comerciales comunes (resistivos, capacitivos, DHTxx, BME280, SHT31/SHT3x, SI7021), comparativa de especificaciones, instrucciones de conexión, código de ejemplo, métodos de calibración/validación y recomendaciones para proyectos hobby y aplicaciones más exigentes (agricultura, estación meteorológica, IoT).

## 2. Resumen ejecutivo / recomendación rápida

● Para proyectos baratos y temporales (hobby): YL-69 / FC-28 (resistivo) + DHT11 (ambiente).

● Para proyectos permanentes/fiables de riego: sensor capacitivo de humedad de suelo (v1.2 o similar) + DHT22 o BME280.

● Para estaciones meteorológicas o aplicaciones profesionales: BME280 o SHT31 (humedad con ±2–3 %RH) y sensores de suelo capacitivo bien calibrados. (Evidencia: corrosión en sondas resistivas; ventajas de capacitivo; especificaciones DHT/BME).

## 3. Tipos de sensores: principios y características

### 3.1. Sensores de humedad de suelo

#### **A) Resistivos(YL-69 / FC-28)**

● Principio: miden la conductividad eléctrica entre dos electrodos insertados en el suelo (mayor humedad entonces más conductividad).

● Ventajas: muy baratos, fáciles de leer (analógico), disponibles en módulos con salida A0 y salida digital.

● Desventajas: corrosión de los electrodos cuando están permanentemente alimentados/insertados, vida útil corta; sensibles a la salinidad del suelo y variaciones químicas. Para alargar la vida se sugiere alimentar el sensor solo al tomar lectura (pulsado) o invertir polaridad periódicamente.

#### **B) Capacitivos (Capacitive Soil Moisture Sensor v1.2)**

● Principio: miden la capacitancia entre placas o trazos cuando se alteran por la humedad del sustrato; la presencia de agua cambia la permitividad dieléctrica.

● Ventajas: mayor durabilidad (sin electrodos expuestos), menos corrosión, lecturas más estables y menos afectadas por salinidad; adecuados para instalaciones semi-permanentes. Estudios recientes muestran buena relación costo/precisión frente a equipos TDR profesionales.

● Desventajas: requieren calibración para convertir lectura analógica (V) a contenido volumétrico de agua (VWC). Algunos clones baratos pueden tener problemas de calidad o envejecimiento; comprar de proveedores confiables.

### 3.2. Sensores de humedad ambiental (aire)

#### **A) DHT11 / DHT22 (AM2302)**

● DHT11: rango ~20–80 %RH, precisión ≈ ±5 %RH, económico; muestreo lento.

● DHT22 / AM2302: rango 0–100 %RH (o amplio), mejor precisión ≈ ±2–5 %RH según fuente, mayor resolución; más adecuado que DHT11 si necesitas mejor lectura. Ambos usan protocolo digital de un solo pin y requieren lectura no más de 0.5–1 Hz.

#### **B) Sensores de precisión I²C/SPI (BME280, SHT31/SHT3x, SI7021)**

● BME280 (Bosch): mide humedad, temperatura y presión; precisión humedad ~±3 %RH típ. Soporta I²C y SPI, bajo consumo y buena estabilidad a largo plazo. Ideal si quieres también presión (altitud/meteorología).

● SHT31 / SHT3x (Sensirion): sensores de alta precisión para RH y temperatura, linealidad y estabilidad superiores a DHT; comunicación I²C y mejores prestaciones térmicas. (Usados en aplicaciones profesionales).

## 4. Especificaciones útiles y comparativa resumida

*Nota:* los valores exactos dependen del fabricante / módulo. Aquí se muestran rangos típicos basados en datasheets y guías.

### ● YL-69 / FC-28 (resistivo)

○ Salida: analógica (0–Vcc).

○ Vida útil: baja si se deja alimentado y en contacto permanente (corrosión)

### ● Capacitive Soil Moisture Sensor v1.2

○ Salida: analógica (AOUT) — generalmente 0–3.3/0–5V según módulo.

○ Durabilidad: alta relativa; menos corrosión.

### ● DHT11

○ Humedad: 20–80 %RH, ±5 %RH.

○ Tiempo lectura: ≤1 Hz.

### ● DHT22 / AM2302

○ Humedad: 0–100 %RH, precisión ±2–5 %RH (mejor que DHT11), -40–80°C.

### ● BME280

○ Humedad: precisión típica ±3 %RH; I²C (0x76/0x77) o SPI; bajo consumo; alta estabilidad.

### ● SHT31 / SHT3x / SI7021

○ Humedad: típicamente ±2 %RH (SHT31), buena estabilidad y respuesta.

# 5. Evidencias de Código

## 4.1 Historia de Usuario 1 — Conectar y programar sensor de humedad

Archivo: sensor\_humedad/sensor\_humedad.ino

const int PinTemp = A0; // Defino Pin Temperatura

const int PinHumAmb = A1; // Defino Pin Humedad Ambiente

const int PinHumSuelo = A2; // Defino Pin Humedad Suelo

const int PinLuz = A3; // Defino Pin iluminación.

## 

## 4.2 ID-1.1.3— Diseñar la interfaz de usuario para mostrar la lectura de humedad

Archivo: components/login.js

#include <LiquidCrystal.h> // Libreria de pantalla LCD

// INICIALIZACIÓN DE VARIABLES Y CONSTANTES

//defino constantes

const int PinPower = 13; // Defino Pin switch.

const int PinExtBot = 2; // Defino Pin boton abrir puerta

const int PinIntPir = 8; // Defino Pin PIR interior

const int PinTemp = A0; // Defino Pin Temperatura

const int PinHumAmb = A1; // Defino Pin Humedad Ambiente

const int PinHumSuelo = A2; // Defino Pin Humedad Suelo

const int PinLuz = A3; // Defino Pin iluminación.

const int PinExtBot1 = 3; // Defino Pin Boton cerrar puerta

//defino variables

int VarPower = 0; // variable que recibe el inicio del sistema

int VarBot1 = 0; // variable de lectura abrir puerta ext

int VarBot2 = 0; // variable de lectura cerrar puerta ext

int VarPir2 = 0; // variable de lectura estado pir interior.

float Temp = 0; // variable lectura temperatura

float HumAmb = 0; // variable humedad ambiente

float HumSuelo = 0; // variable humedad suelo

int VarLuz = 0; // variable lectura iluminación

String mensaje; // variable que contiene texto a mostrar

int Contador = 0; // variable de usos varios

//defino variables de estado

int VarEstVent = 0; //para verificar estado ventilacion

int VarEstCale = 0; //para verificar estado calefaccion x tmp

int VarEstAspe = 0; //para verificar estado aspersor

int VarEstCaHu = 0; //para verificar estado calefaccion x humedad

int VarEstHuSu = 0; //para verificar estado aspersor x humedad del suelo

//defino objetos

LiquidCrystal lcd(12, 11, 7, 6, 5, 4); //definimos conexion pantalla LCD

// RUTINA PRINCIPAL - SETUP

void setup() {

Serial.begin(9600);

pinMode(PinPower, INPUT); //defino pin de inicio como entrada

pinMode(PinExtBot, INPUT); //defino como entrada pin boton abrir

pinMode(PinExtBot1, INPUT); // defino como entrada pin boton cerrar

pinMode(PinIntPir, INPUT); //defino como entrada pin pir interior

while (VarPower == 0) {

VarPower = digitalRead(PinPower);// genero un loop en setup a la espera de

MjeDefinido(5); // muestro mensaje para que inicie

delay(80);

} // que se inicie el sistema a través de switch

StartLcd(); //inicio el lcd

MjeDefinido(1); // muestro mensajes

delay(2000);

attachInterrupt(digitalPinToInterrupt(PinExtBot), AbrirPta, RISING);

attachInterrupt(digitalPinToInterrupt(PinExtBot1), CerrarPta, RISING);

}

// RUTINA LOOP - REPETITIVA - PROGRAMA PINCIPAL

void loop() {

if (digitalRead(PinPower) == 1){

MjeDefinido(3); // leyendo sensores

delay(1000);

}

if (digitalRead(PinPower) == 1){

leyendosensores(1); // temp

}

if (digitalRead(PinPower) == 1){

leyendosensores(2); // Humedad Ambiente

}

if (digitalRead(PinPower) == 1){

leyendosensores(3); // Humedad Suelo

}

if (digitalRead(PinPower) == 1){

leyendosensores(5); // Pir interior

}

if (digitalRead(PinPower) == 1){

leyendosensores(6); // sensor iluminación

}

if (digitalRead(PinPower) == 0){

MjeDefinido(2); // Indico que el sistema esta apagado

PararTodo(); // aca llamo a funcion q apaga todos los actuadores

}

delay(1000);

}

// PROCEDIMIENTOS ESPECIALES

// --------------------------------------------------

//Rutina de inicio de pantalla

void StartLcd() {

lcd.begin(16, 2);

lcd.noDisplay();

lcd.clear();

lcd.home();

}

//---------------------------------------------------

// Rutina para mostrar mensajes en el lcd

void MostrarMje(int c, int f,String mje,float nro1){

lcd.display();

lcd.setCursor(c,f);

if (mje != "XX"){

lcd.print(mje);

}

if (nro1 != 0) {

lcd.print((int) nro1);

}

}

//---------------------------------------------------

// Rutina que indica que mensaje mostrar.

void MjeDefinido(int nro){

switch (nro){

case 1:

lcd.clear();

mensaje = String("SISTEMA DE");

MostrarMje(0,0,mensaje,0);

mensaje = String("INVERNADERO");

MostrarMje(5,1,mensaje,0);

break;

case 2:

lcd.clear();

mensaje = String("INVERNADERO");

MostrarMje(0,0,mensaje,0);

mensaje = String("DETENIDO");

MostrarMje(5,1,mensaje,0);

break;

case 3:

lcd.clear();

mensaje = String("LEYENDO");

MostrarMje(0,0,mensaje,0);

mensaje = String("SENSORES");

MostrarMje(5,1,mensaje,0);

break;

case 4:

lcd.clear();

mensaje = String("Temp: ");

MostrarMje(0,0,mensaje,0);

break;

case 5:

lcd.clear();

mensaje = String("ENCIENDA SWITCH");

MostrarMje(0,0,mensaje,0);

break;

case 6:

lcd.clear();

mensaje = String("Hum Amb: ");

MostrarMje(0,0,mensaje,0);

break;

case 7:

lcd.clear();

mensaje = String("Hum Suelo: ");

MostrarMje(0,0,mensaje,0);

break;

case 8:

lcd.clear();

mensaje = String("Puerta Exterior");

MostrarMje(0,0,mensaje,0);

break;

case 9:

lcd.clear();

mensaje = String("Movimiento");

MostrarMje(0,0,mensaje,0);

mensaje = String("Interior: ");

MostrarMje(0,1,mensaje,0);

break;

case 10:

mensaje = String("ESTADO:");

MostrarMje(0,1,mensaje,0);

break;

case 11:

lcd.clear();

mensaje = String("Iluminacion");

MostrarMje(0,0,mensaje,0);

mensaje = String("Interior: ");

MostrarMje(0,1,mensaje,0);

break;

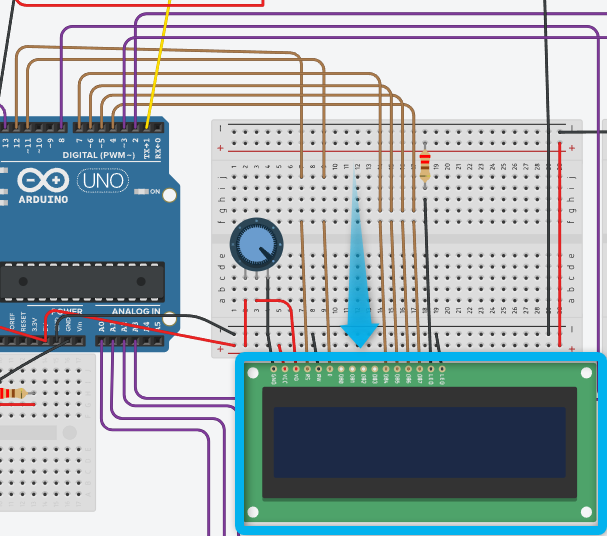
default:

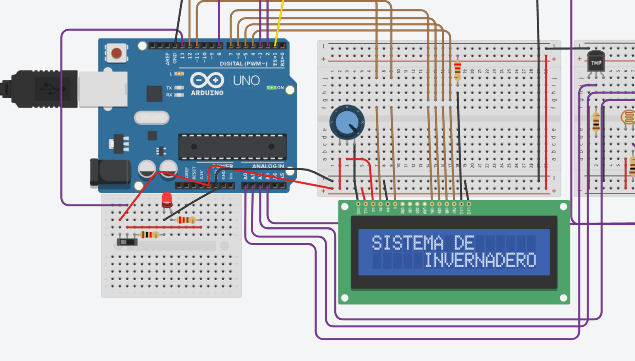
lcd.clear();

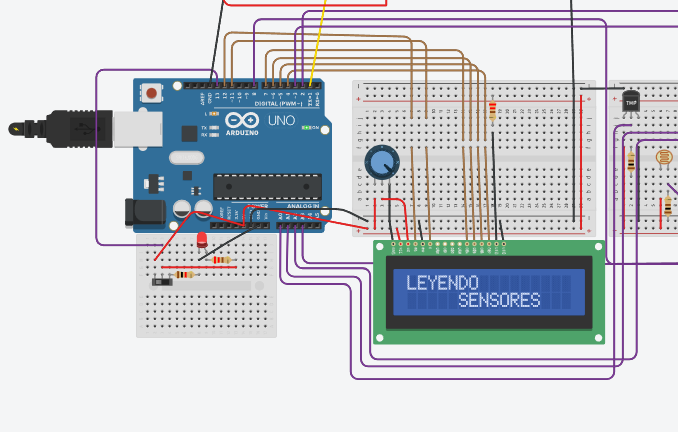
break;

}

}







Descripción del código: El presente fragmento de código en Python utiliza la librería Tkinter para diseñar una interfaz gráfica que muestre la lectura de humedad.

Primero se importa tkinter con el alias tk. Luego, dentro de la función interfaz\_humedad(), se crea la ventana principal con tk.Tk(), a la que se le asigna un título, dimensiones y un color de fondo.

Se añade un título principal mediante un Label con el texto Monitor de Humedad, configurado con fuente Arial en negrita y color azul oscuro, que se coloca con un margen superior para darle espacio visual. Después, se crea otro Label que muestra la humedad inicial con el texto Humedad: -- %, usando un tamaño de fuente más grande y color verde oscuro, pensado para representar la lectura del sensor de humedad.

A continuación, se incorpora un botón con la etiqueta Actualizar, diseñado con fuente mediana, fondo azul claro y un relieve para dar apariencia de botón real, aunque en esta versión aún no ejecuta ninguna acción, simplemente aparece en la ventana.

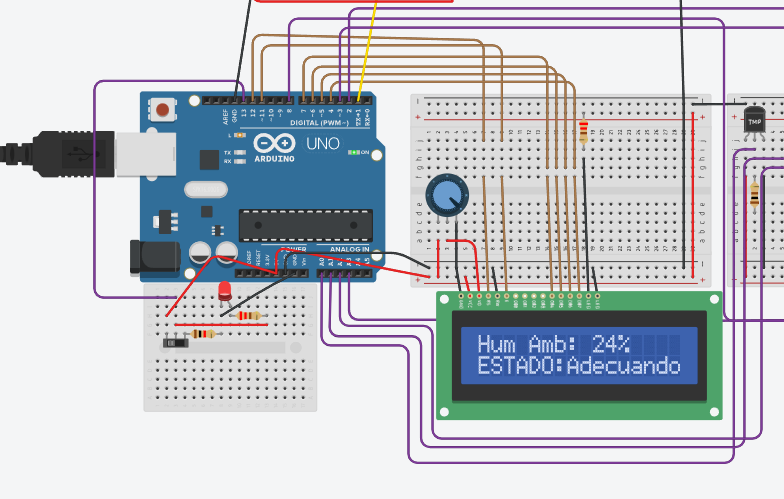
Todos los elementos (Label y Button) se organizan en la interfaz usando el gestor de geometría .pack(), que permite posicionarlos uno debajo del otro con márgenes verticales (pady). Finalmente, la instrucción root.mainloop() mantiene la ventana activa esperando interacciones del usuario, evitando que se cierre automáticamente.

Para ejecutar la aplicación, se utiliza la estructura if \_\_name\_\_ == "\_\_main\_\_": interfaz\_humedad(), lo que asegura que la interfaz se muestre solo si el archivo se corre directamente y no cuando se importe como módulo en otro programa.

En resumen, este código construye una ventana simple con un título, un indicador de humedad y un botón de actualización, sirviendo como base para simular o mostrar valores de humedad en un sistema más completo.

## 4.3 ID-1.1.4— Integrar la lectura del sensor a la interfaz

| float leyendosensores(int valor){  switch(valor){    case 1: //sensar temperatura  Temp = (5.0 /1024 \* analogRead(PinTemp))\*100-50;  if (VarEstCale == 1) {  MjeDefinido(4);  MostrarMje(6,0,"XX",Temp);  MostrarMje(9,0,"C",0);  MjeDefinido(10);  MostrarMje(7,1,"Adecuando",0);  Serial.write('k'); //mantengo cale  if (Temp >= 25) { //hasta llegar a 25º  VarEstCale = 0;  Serial.write('l');  }  delay(1000);  }  if (VarEstVent == 1) {  MjeDefinido(4);  MostrarMje(6,0,"XX",Temp);  MostrarMje(9,0,"C",0);  MjeDefinido(10);  MostrarMje(7,1,"Adecuando",0);  Serial.write('g'); // mantengo ventilador  if (Temp <= 20) { // hasta que llegue a los 20º  VarEstVent = 0;  Serial.write('h');  }  delay(1000);  }  if (VarEstVent != 1 && VarEstCale != 1 && Temp >= 10 && Temp <= 30) {  MjeDefinido(4);  MostrarMje(6,0,"XX",Temp);  MostrarMje(9,0,"C",0);  MjeDefinido(10);  MostrarMje(7,1,"Normal",0); //si todo esta normal entra aca  delay(1000);  }  if (Temp < 10 && VarEstCale != 1) {  MjeDefinido(4);  MostrarMje(6,0,"XX",Temp);  MostrarMje(9,0,"C",0);  MjeDefinido(10);  MostrarMje(7,1,"Baja",0);  Serial.write('k'); // enciende calefacción  Serial.write('h');  VarEstCale = 1;  VarEstVent = 0;  delay(1000);  }  if (Temp > 30 && VarEstVent != 1) {  MjeDefinido(4);  MostrarMje(6,0,"XX",Temp);  MostrarMje(9,0,"C",0);  MjeDefinido(10);  MostrarMje(7,1,"Alta",0);  Serial.write('g'); //enciende ventilacion  Serial.write('l');  VarEstCale = 0;  VarEstVent = 1;  delay(1000);  }  break;    case 2: //sensar Humedad ambiente  HumAmb = (5.0 /1024 \* analogRead(PinHumAmb))\*100-50;  if (VarEstAspe == 1){  MjeDefinido(6);  MostrarMje(9,0,"XX", HumAmb);  MostrarMje(11,0,"%",0);  MjeDefinido(10);  MostrarMje(7,1,"Adecuando",0);  Serial.write('i');  if (HumAmb >=80) {  VarEstAspe = 0;  Serial.write('j');  }  delay(1000);  }  if (VarEstCaHu == 1){  MjeDefinido(6);  MostrarMje(9,0,"XX", HumAmb);  MostrarMje(11,0,"%",0);  MjeDefinido(10);  MostrarMje(7,1,"Adecuando",0);  Serial.write('k');  if (HumAmb <=70) {  VarEstCaHu = 0;  Serial.write('l');  }  delay(1000);  }  if (VarEstAspe != 1 && VarEstCaHu != 1 && HumAmb >= 60 && HumAmb <= 80) {  MjeDefinido(6);  MostrarMje(9,0,"XX", HumAmb);  MostrarMje(11,0,"%",0);  MjeDefinido(10);  MostrarMje(8,1,"Normal",0);  if (VarEstHuSu == 0) { //lo cruzo con humedad de suelo  Serial.write('j'); // para no apagar aspesor cuando se encesite  }  Serial.write('l');  delay(1000);  }  if (HumAmb < 60 && VarEstAspe != 1) {  MjeDefinido(6);  MostrarMje(9,0,"XX", HumAmb);  MostrarMje(11,0,"%",0);  MjeDefinido(10);  MostrarMje(7,1,"Baja",0);  Serial.write('i');  VarEstAspe = 1;  delay(1000);  }  if (HumAmb > 80 && VarEstCaHu != 1){  MjeDefinido(6);  MostrarMje(9,0,"XX", HumAmb);  MostrarMje(11,0,"%",0);  MjeDefinido(10);  MostrarMje(7,1,"Alta",0);  Serial.write('k');  VarEstCaHu = 1;  delay(1000);  }  break;  case 3: //sensar humedad suelo  HumSuelo = (5.0 /1024 \* analogRead(PinHumSuelo))\*100-50;  if (VarEstHuSu == 1) {  MjeDefinido(7);  MostrarMje(11,0,"XX", HumSuelo);  MostrarMje(13,0,"%",0);  MjeDefinido(10);  MostrarMje(7,1,"Adecuando",0);  Serial.write('i');  if (HumSuelo >=70) {  VarEstHuSu = 0;  Serial.write('j');  }  delay(1000);  }  if (VarEstHuSu == 0 && HumSuelo >=50){  MjeDefinido(7);  MostrarMje(11,0,"XX", HumSuelo);  MostrarMje(13,0,"%",0);  MjeDefinido(10);  MostrarMje(7,1,"Normal",0);  if (VarEstAspe == 0) { //aca cruzo con var de hum amb para no  Serial.write('j'); // apagar aspersor cuando es necesario  }  delay(1000);  }  if (HumSuelo < 50 && VarEstHuSu != 1) {  MjeDefinido(7);  MostrarMje(11,0,"XX", HumSuelo);  MostrarMje(13,0,"%",0);  MjeDefinido(10);  MostrarMje(7,1,"Baja",0);  Serial.write('i');  VarEstHuSu = 1;  delay(1000);  }  break;    case 4: //detectar pulsación boton motor puerta externa  // ELIMINADO (MODIFICADO A TRAVES DE INTERRUPCIONES)  break;    case 5: //sensar movimiento interno  VarPir2 = digitalRead(PinIntPir);  if (VarPir2 == HIGH){  MjeDefinido(9);  MostrarMje(10,1,"SI", 0);  Serial.write('c');  Serial.write('j');  delay(1000);  }  else {  MjeDefinido(9);  MostrarMje(10,1,"NO", 0);  Serial.write('d');  delay(1000);  }  break;    case 6: //Sensar iluminación  VarLuz = analogRead(PinLuz);  if (VarLuz < 100){  MjeDefinido(11);  MostrarMje(10,1,"BAJA", 0);  Serial.write('a');  delay(1000);  }  else {  MjeDefinido(11);  MostrarMje(10,1,"ALTA", 0);  Serial.write('b');  delay(1000);  }  break;    default:  break;  }  } |
| --- |



El código comienza incluyendo las librerías necesarias para el sensor DHT22 y la pantalla LCD, definiendo los pines de conexión y creando los objetos correspondientes. En la función setup(), se inicializa la comunicación serial a 9600 baudios, se configura la pantalla LCD como display de 16x2 caracteres y se inicia el sensor DHT22. Posteriormente, se muestra una pantalla de bienvenida en el LCD con el mensaje "SISTEMA DHT22" y "INICIANDO..." durante 3 segundos, mientras que por el monitor serial se imprime un mensaje indicando que el sistema está listo y esperando lecturas. Todo este proceso de inicialización asegura que todos los componentes hardware estén correctamente configurados antes de comenzar el ciclo principal de medición.

El núcleo del programa se ejecuta en la función loop(), que utiliza temporizadores no bloqueantes para gestionar las diferentes tareas. Cada 2 segundos (definido por intervaloLectura), el sistema realiza una lectura del sensor DHT22, obteniendo los valores de temperatura en grados Celsius y humedad relativa en porcentaje. Si la lectura es exitosa, calcula automáticamente el índice de calor (sensación térmica) utilizando la función computeHeatIndex(). En caso de error en la lectura, el sistema detecta valores NaN (Not a Number) y muestra mensajes de error tanto en el LCD como en el monitor serial, indicando que se debe verificar la conexión del sensor, lo que garantiza robustez ante fallos hardware.

La presentación de datos se realiza de forma simultánea en la pantalla LCD y el monitor serial. En el LCD, la primera línea muestra la temperatura con un decimal seguida del símbolo de grados Celsius, mientras que la segunda línea presenta la humedad como valor porcentual entero. Adicionalmente, se incluyen iconos visuales que cambian dinámicamente según los valores medidos: para temperatura usa (frío), (normal) o (caliente), y para humedad utiliza (seco), (normal) o (húmedo). Paralelamente, el monitor serial recibe toda la información en formato texto, incluyendo temperatura, humedad, índice de calor y un descriptor del estado ambiental, facilitando el registro y análisis histórico de los datos mediante herramientas externas.

## 4.4 ID-1.1.5— Probar la visualización de datos en tiempo real

| int pinHumedad = 34; // pin analógico en ESP32  int valorHumedad = 0;  void setup() {  Serial.begin(115200);  delay(1000);  Serial.println("INICIO\_SENSOR");  }  void loop() {  valorHumedad = analogRead(pinHumedad); // 0 - 4095 en ESP32  int humedadPorc = map(valorHumedad, 4095, 0, 0, 100);  unsigned long ts = millis();  // Formato: TS,RAW,HUM  Serial.print(ts);  Serial.print(",");  Serial.print(valorHumedad);  Serial.print(",");  Serial.println(humedadPorc);  delay(1000); // 1 segund  } |
| --- |

visualizacion\_tk\_serial.py

| import tkinter as tk  from tkinter import ttk  import serial  import threading  import time  from collections import deque  import matplotlib  matplotlib.use("TkAgg")  from matplotlib.figure import Figure  from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  # Configuración  SERIAL\_PORT = "COM3" # Windows ej. "COM3"; Linux ej. "/dev/ttyUSB0"  BAUDRATE = 115200  # Buffer para plot  MAX\_POINTS = 60  datos = deque(maxlen=MAX\_POINTS)  timestamps = deque(maxlen=MAX\_POINTS)  # Conectar serial en hilo  ser = None  stop\_thread = False  def lector\_serial():  global ser, stop\_thread  try:  ser = serial.Serial(SERIAL\_PORT, BAUDRATE, timeout=1)  except Exception as e:  print("Error abriendo puerto serial:", e)  return  while not stop\_thread:  try:  linea = ser.readline().decode('utf-8').strip()  if not linea:  continue  # Esperamos formato: TS,RAW,HUM  partes = linea.split(',')  if len(partes) >= 3:  try:  ts = int(partes[0])  raw = int(partes[1])  hum = int(partes[2])  except:  continue  timestamps.append(ts/1000.0) # segundos  datos.append(hum)  # si quieres, imprimir linea completa  # print("Serial:", linea)  except Exception as e:  print("Error lectura serial:", e)  time.sleep(0.5)  if ser and ser.is\_open:  ser.close()  # GUI  class App:  def \_\_init\_\_(self, root):  self.root = root  root.title("Monitor de Humedad - Tiempo Real")  root.geometry("500x360")  self.label = ttk.Label(root, text="Humedad: -- %", font=("Arial", 16))  self.label.pack(pady=10)  # Boton de reconectar  self.btn\_frame = ttk.Frame(root)  self.btn\_frame.pack()  self.btn\_start = ttk.Button(self.btn\_frame, text="Iniciar lectura", command=self.start)  self.btn\_start.grid(row=0, column=0, padx=5)  self.btn\_stop = ttk.Button(self.btn\_frame, text="Detener lectura", command=self.stop, state="disabled")  self.btn\_stop.grid(row=0, column=1, padx=5)  # Plot matplotlib embebido  self.fig = Figure(figsize=(5,2.5))  self.ax = self.fig.add\_subplot(111)  self.ax.set\_ylim(0, 100)  self.ax.set\_ylabel("% Humedad")  self.ax.set\_xlabel("Tiempo (s)")  self.line, = self.ax.plot([], [], lw=1)  self.canvas = FigureCanvasTkAgg(self.fig, master=root)  self.canvas.get\_tk\_widget().pack(fill="both", expand=True, padx=10, pady=10)  # Actualizador GUI  self.update\_gui()  def start(self):  global lector\_thread, stop\_thread  stop\_thread = False  lector\_thread = threading.Thread(target=lector\_serial, daemon=True)  lector\_thread.start()  self.btn\_start.config(state="disabled")  self.btn\_stop.config(state="normal")  def stop(self):  global stop\_thread  stop\_thread = True  self.btn\_start.config(state="normal")  self.btn\_stop.config(state="disabled")  def update\_gui(self):  # Actualiza etiqueta con último valor  if datos:  self.label.config(text=f"Humedad: {datos[-1]} %")  # actualizar plot  x = list(range(-len(datos)+1, 1)) # eje relativo (últimos N)  y = list(datos)  self.line.set\_data(x, y)  self.ax.set\_xlim(min(x)-1, 0+1)  self.canvas.draw\_idle()  self.root.after(500, self.update\_gui) # actualizar cada 500 ms  if \_\_name\_\_ == "\_\_main\_\_":  root = tk.Tk()  app = App(root)  root.mainloop() |
| --- |

## 4.5 ID-1.1.6— Implementar alerta en caso de fallo del sensor

| #include <DHT.h>  #include <LiquidCrystal.h>  // Configuración del sensor DHT  #define DHT\_PIN 2  #define DHT\_TYPE DHT22  // Configuración LCD  LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  // Pin para buzzer de alerta  #define BUZZER\_PIN 8  DHT dht(DHT\_PIN, DHT\_TYPE);  float temperatura, humedad;  int fallosConsecutivos = 0;  bool alertaActiva = false;  void setup() {  Serial.begin(9600);  lcd.begin(16, 2);  pinMode(BUZZER\_PIN, OUTPUT);  dht.begin();    lcd.print("Iniciando...");  delay(2000);  }  void loop() {  // Leer sensor cada 2 segundos  if (leerSensor()) {  // Lectura exitosa  mostrarDatosNormal();  alertaActiva = false;  noTone(BUZZER\_PIN); // Apagar alerta  } else {  // Error en lectura  fallosConsecutivos++;  mostrarAlertaError();  activarAlertaSonora();  alertaActiva = true;  }    delay(2000);  }  bool leerSensor() {  humedad = dht.readHumidity();  temperatura = dht.readTemperature();    // Si la lectura es exitosa, resetear contador  if (!isnan(humedad) && !isnan(temperatura)) {  fallosConsecutivos = 0;  return true;  }  return false;  }  void mostrarDatosNormal() {  lcd.clear();  lcd.setCursor(0, 0);  lcd.print("T:");  lcd.print(temperatura, 1);  lcd.print("C H:");  lcd.print(humedad, 0);  lcd.print("%");    lcd.setCursor(0, 1);  lcd.print("Sensor: OK");  }  void mostrarAlertaError() {  lcd.clear();  lcd.setCursor(0, 0);  lcd.print("ALERTA!");  lcd.setCursor(0, 1);  lcd.print("Error sensor DHT");    Serial.print("Alerta! Fallo #");  Serial.println(fallosConsecutivos);  }  void activarAlertaSonora() {  // Tono de alerta intermitente  tone(BUZZER\_PIN, 1000, 500);  } |
| --- |

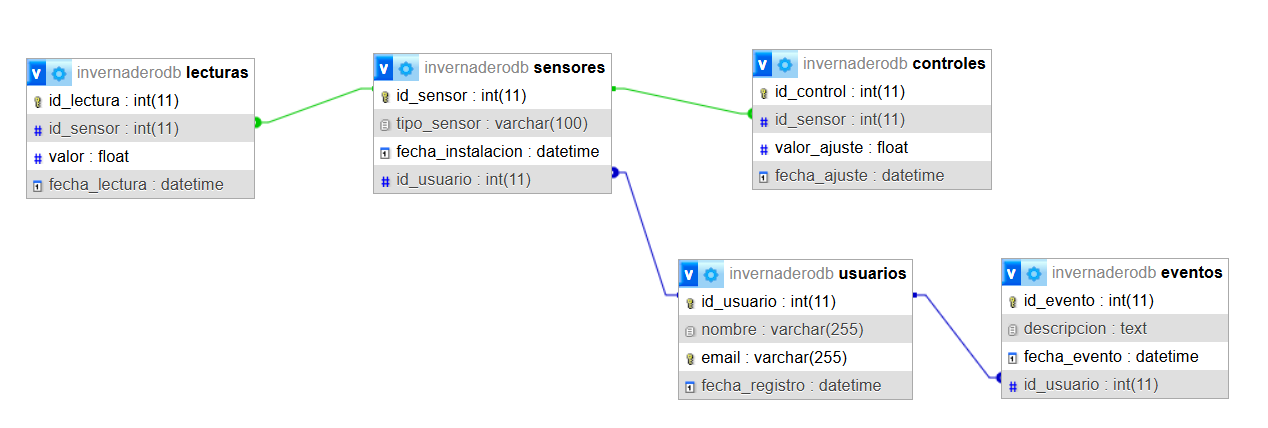
El código comienza incluyendo las librerías necesarias para el sensor DHT22 y la pantalla LCD, definiendo los pines de conexión específicos para cada componente. En la función setup(), se inicializa la comunicación serial a 9600 baudios para el monitoreo, se configura la pantalla LCD con 16 columnas y 2 filas, se establece el pin del buzzer como salida y se inicia el sensor DHT22. El sistema muestra un mensaje de "Iniciando..." en la pantalla durante 2 segundos, indicando que todos los componentes se están preparando para comenzar las lecturas. Esta fase asegura que tanto el hardware como el software estén correctamente configurados antes de iniciar el ciclo principal de medición.

En el bucle principal loop(), el sistema realiza lecturas del sensor cada 2 segundos. La función leerSensor() obtiene los valores de humedad y temperatura, verificando si son números válidos mediante isnan(). Si ambas lecturas son correctas, el contador de fallos consecutivos se reinicia a cero y el sistema continúa en modo normal. Sin embargo, si alguna lectura falla (devuelve NaN), el contador de fallos se incrementa y se activa inmediatamente el protocolo de alerta. Esta detección automática permite identificar problemas de conexión, fallos del sensor o condiciones ambientales que impidan lecturas precisas.

Cuando se detecta un fallo, el sistema activa múltiples formas de alerta simultáneamente: ¡en la pantalla LCD muestra “ALERTA!" y "Error sensor DHT", por el monitor serial envía un mensaje indicando el número de fallo consecutivo, y activa el buzzer con un tono de 1000 Hz durante 500 milisegundos. En contraste, cuando el sensor funciona correctamente, la pantalla muestra temperatura y humedad en formato claro ("T: XX.XC H: XX%") con el mensaje "Sensor: OK", y se silencia el buzzer. Este enfoque dual proporciona una respuesta inmediata ante fallos mientras mantiene una interfaz limpia durante el funcionamiento normal.

## Historia de Usuario 1.2: Registro histórico

**1.2.1 Configurar base de datos local para almacenar lecturas diarias 4**

**Base de Datos:**

**Revisión de tu modelo**

1. **Usuarios**
   * Correcto: identificación única, email único y fecha de registro.
   * Puedes agregar campo rol o estado (activo/inactivo) si quieres diferenciar administradores de usuarios comunes.
2. **Sensores**
   * Bien vinculado a un usuario (propietario/instalador).
   * Podrías agregar campos como:
   * ubicacion VARCHAR(255),
   * unidad\_medida VARCHAR(50)

**para saber dónde está instalado el sensor y qué mide (°C, %, ppm, etc.).**

1. **Lecturas**
   * Correcto para lecturas diarias.
   * Si planeas guardar lecturas muy frecuentes (cada segundo o minuto), conviene usar BIGINT para id\_lectura y considerar particionar la tabla por fecha.
2. **Controles**
   * Está bien, sirve para registrar ajustes (ej: apertura de válvulas, ventiladores, etc.).
   * Tal vez convenga relacionar también con Eventos si quieres trazar qué ajuste fue producto de un evento (alarma).
3. **Eventos**
   * Correcto, ya ligado a usuario.
   * Puedes agregar severidad o tipo de evento:
   * tipo\_evento ENUM('Alerta','Error','Mantenimiento','Info') NOT NULL

**Ejemplo de consultas útiles**

1. **Obtener lecturas de un sensor en la última semana:**

SELECT fecha\_lectura, valor

FROM Lecturas

WHERE id\_sensor = 1

AND fecha\_lectura >= NOW() - INTERVAL 7 DAY

ORDER BY fecha\_lectura;

1. **Promedio diario de temperatura por sensor:**

SELECT DATE(fecha\_lectura) AS dia, AVG(valor) AS promedio

FROM Lecturas

WHERE id\_sensor = 1

GROUP BY DATE(fecha\_lectura)

ORDER BY dia;

1. **Últimos eventos de un usuario:**

SELECT descripcion, fecha\_evento

FROM Eventos

WHERE id\_usuario = 2

ORDER BY fecha\_evento DESC

LIMIT 10;

## Historia de Usuario 2.1: Riego automático por humedad

**2.1.1 Investigar y adquirir la bomba de riego**

1. Tipos de bombas recomendadas

* Mini bomba sumergible DC (3V–12V)
  + Ideal para prototipos o pequeños invernaderos.
  + Bajo consumo de energía.
  + Caudal aproximado: 80–120 L/h.
  + Precio referencial: S/ 25 – 40.
* Bomba peristáltica DC (5V–12V)
  + Diseñada para riego por goteo o dosificación de nutrientes.
  + Permite un control preciso del flujo de agua.
  + Precio referencial: S/ 45 – 70.
* Bomba sumergible DC de mayor capacidad (12V–24V)
  + Adecuada para invernaderos medianos o grandes.
  + Caudal aproximado: 240–800 L/h.
  + Requiere fuente de alimentación independiente y módulo relé.
  + Precio referencial: S/ 60 – 120.

2. Componentes complementarios necesarios

* Módulo relé 5V o MOSFET: para controlar la bomba desde el Arduino.
  + Precio: S/ 10 – 20.
* Fuente de alimentación externa (según el voltaje de la bomba).
  + Precio: S/ 20 – 40.
* Sensor de humedad de suelo: para medir el nivel de humedad y automatizar el riego.
  + Precio: S/ 10 – 25.
* Mangueras y válvulas de riego: costo variable (S/ 15 – 30).

3. Ejemplo de funcionamiento del sistema

1. El sensor de humedad detecta sequedad en el suelo.
2. Arduino procesa la señal recibida.
3. Se activa el módulo relé conectado al Arduino.
4. El relé enciende la bomba de agua.
5. El riego se detiene cuando la humedad del suelo alcanza el nivel deseado.

4. Opciones de adquisición en Perú

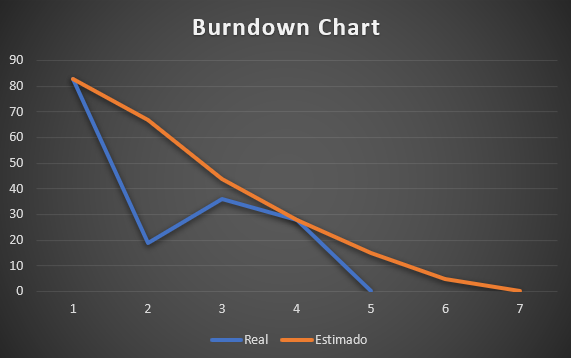
* Mercado Libre Perú
  + “Mini bomba sumergible DC 5V–12V para Arduino”: S/ 25 – 40.
  + “Bomba de agua peristáltica DC 12V”: S/ 45 – 70.
  + “Kit de riego automático (bomba + sensor + relé)”: S/ 60 – 90.
* Tiendas de electrónica (Promelsa, Infotronica, Steren, distribuidores locales).
  + Bombas DC sumergibles y accesorios.
  + Precios similares a Mercado Libre.
* AliExpress (importación)
  + Bombas de 5V, 6V y 12V desde S/ 12 – 25.
  + Kits completos desde S/ 40 – 70 (con tiempos de entrega más largos).

**Recomendación para el proyecto**

* Prototipo o pruebas en laboratorio: Mini bomba sumergible DC 12V + módulo relé + sensor de humedad (costo total aproximado S/ 60 – 80).
* Invernadero en operación real: Bomba DC 12V de mayor caudal + sistema de riego por goteo + relé + fuente de 2A (costo total aproximado S/ 100 – 160).

# 5. Burndown Chart

| Backlog ID | Historia de Usuario | Tiempo Estimado | 21/04/2025 | 22/04/2025 | 23/04/2025 | 24/04/2025 | 25/04/2025 | 26/04/2025 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** |
| HU-1.1 | Visualización en tiempo real | 28 hrs. | 8 | 8 | 5 | 3 | 3 | 1 |
| HU-1.2 | Registro histórico | 36 hrs. | 8 | 8 | 8 | 6 | 4 | 2 |
| HU-1.3 | Umbrales personalizables | 19 hrs. |  | 7 | 3 | 4 | 3 | 2 |
| Tiempo de trabajo | | 83 hrs. | 67 | 44 | 28 | 15 | 5 | 0 |



# 6. Análisis del Sprint

* Se completaron todas las tareas planificadas:
  + Investigación de sensores de humedad compatibles con Arduino.
  + Conexión y programación del sensor de humedad.
  + Diseño de la interfaz de usuario para mostrar lecturas.
  + Integración de la lectura del sensor en la interfaz.
  + Pruebas de visualización de datos en tiempo real.
  + Implementación de una alerta en caso de fallo del sensor.
* El progreso fue constante y sin bloqueos significativos.
* Las decisiones técnicas sobre integración sensor–interfaz fueron efectivas para este primer Sprint.
* El código se modularizó y se documentó en comentarios para facilitar mantenimiento y futuras mejoras.
* Las estimaciones fueron precisas y se mantuvo un ritmo de trabajo estable.
* Los objetivos del Sprint se cumplieron satisfactoriamente, dejando la base preparada para nuevas funcionalidades.

# 7. Conclusiones

El Sprint 1 concluyó exitosamente con todas las funcionalidades básicas implementadas en torno al sensor de humedad. Se investigaron sensores compatibles con Arduino, se logró la conexión y programación del módulo, y se diseñó la interfaz de usuario para mostrar en tiempo real la lectura de humedad. Asimismo, se integró la lectura del sensor en la interfaz, se validó la visualización de datos en tiempo real y se implementó una alerta ante posibles fallos del sensor.

El equipo demostró capacidad de entrega puntual y una adecuada gestión de riesgos menores. Las métricas indican que la carga de trabajo fue bien distribuida y los objetivos del sprint se cumplieron de manera satisfactoria, quedando la base lista para futuras mejoras y ampliaciones del sistema.