Sistema de autorriego utilizando microcontrolador ESP32

Juan Camilo Parra Sanchez

Ingeniería de software, Universidad Cooperativa de Colombia

Contexto de la ingeniería de software

Luis Revelo Tovar

Diciembre 2024

Introducción

En este proyecto se desarrolla un sistema de riego automatizado para plantas utilizando un microcontrolador ESP32. El objetivo principal es monitorear los niveles de humedad del suelo y controlarlos de manera. Esto se logra integrando sensores de humedad, módulos de relé y conectividad Wi-Fi para proporcionar una solución autónoma y accesible desde cualquier dispositivo conectado a la misma red.

El sistema se compone de dos sensores de humedad, los cuales miden el nivel de humedad en dos plantas de manera independiente. A partir de estos datos, el ESP32 decide si activar o desactivar las bombas de riego, garantizando así que las plantas se mantengan dentro de un rango óptimo de humedad. Este control automático asegura un riego eficiente, reduce el desperdicio de agua y evita el exceso de riego que podría dañar las plantas.

Una característica clave de este sistema es la implementación de un servidor web en el ESP32, accesible desde cualquier navegador. Esta interfaz web permite al usuario visualizar en tiempo real los niveles de humedad, el porcentaje relativo de humedad del suelo y el estado de las bombas de agua (encendidas o apagadas). Además, la página web se actualiza dinámicamente mediante solicitudes asíncronas, lo que brinda al usuario una experiencia interactiva y fluida.

El diseño del sistema es altamente escalable y puede adaptarse para gestionar múltiples plantas o incluir funcionalidades adicionales, como el control remoto desde una aplicación móvil. Este proyecto no solo facilita el cuidado de las plantas, sino que también representa una solución tecnológica para la agricultura urbana y doméstica, promoviendo la eficiencia en el uso de recursos y el acceso a tecnologías modernas en el hogar.

Metodología

La metodología para el desarrollo del sistema de autoriego basado en ESP32 se dividió en cuatro etapas principales: selección de materiales, diseño del circuito, desarrollo del código, y montaje físico y pruebas. A continuación, se describe detalladamente cada etapa:

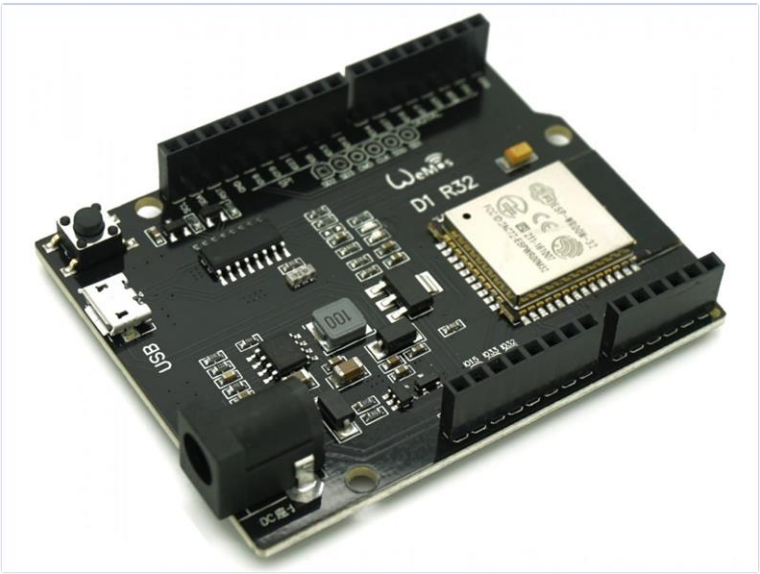
Materiales usados

A continuación, se muestra los materiales que fueron usados y el funcionamiento de estos

Microcontrolador ESP32

Figura 1

Modulo ESP32 (wemos D1 R32)



El ESP32 es el componente central del sistema, encargado de procesar los datos provenientes de los sensores de humedad y controlar el encendido y apagado de las bombas de agua a través del módulo de relé.

Además, se utilizó la capacidad de conectividad Wi-Fi del ESP32 para realizar un monitore remoto de los niveles de humedad de la planta

El ESP32 fue alimentado mediante un adaptador conectado a una toma de corriente de pared. Esto eliminó la necesidad de utilizar baterías para su funcionamiento, garantizando un

suministro continuo de energía, especialmente útil para mantener activa su capacidad de conexión Wi-Fi.

Sensores de humedad del suelo (capacitivos)

Figura 2

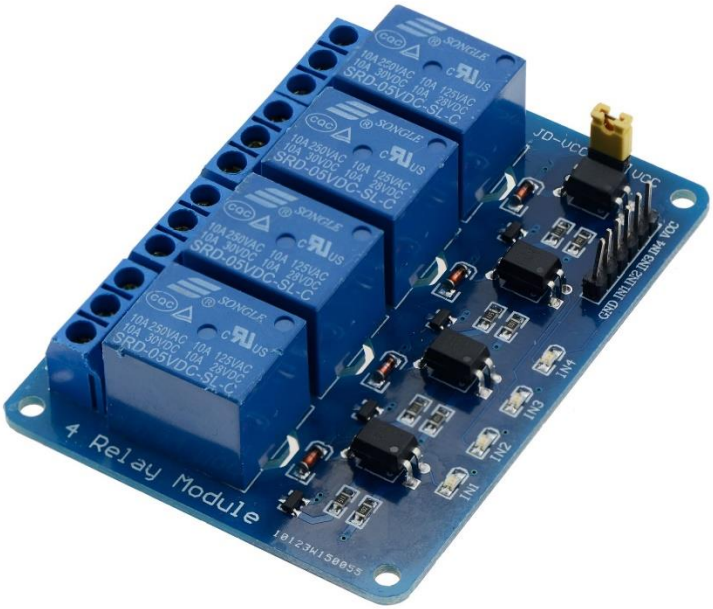
Sensor de humedad del suelo, acompañado con jumpers para su conexión

Los sensores de humedad miden la cantidad de agua en el suelo mediante variaciones en la resistencia eléctrica. Se emplearon sensores capacitivos para medir los niveles de humedad en el suelo. Estos sensores fueron conectados a los pines analógicos del ESP32 mediante cables jumper, lo que permitió establecer una comunicación confiable y precisa entre los sensores y el microcontrolador.

Modulo relé de 5V

Figura 3

Módulo de 4 relés



Los módulos de relé funcionan como interruptores electrónicos. El módulo relé fue usado para controlar las bombas de agua, este módulo se conecta al ESP32 y permite activar o desactivar las bombas de agua según los valores recibidos de los sensores de humedad. El módulo relé fue alimentado atreves de las salidas de energía del microcontrolador. Cada canal del módulo fue configurado oara controlar de manera independiente cada una de las bombas.

Bombas de Agua sumergibles

Figura 4

Bomba de Agua sumergible de 5v

Se emplearon dos bombas de agua sumergibles pequeñas, ideales para aplicaciones de riego automatizado. Cada bomba fue alimentada por un par de baterías AA, conectadas en serie

para alcanzar el voltaje necesario. Esta configuración garantiza una operación estable y elimina la dependencia de una fuente de alimentación compartida, reduciendo el riesgo de fallos en caso de agotamiento de energía.

Tuberías plásticas

Figura 3

Tubería plástica, compatible con la bomba de agua

La tubería plástica permite el transporte de agua desde el recipiente de agua atravez de las bombas de agua hasta las planats

Cables Jumper

Figura 4

Cables Jumper

Se utilizaron cables jumper hembra-macho y macho-macho para establecer conexiones entre los sensores de humedad, el módulo de relé y el ESP32.

Recipientes Plásticos

Figura 5

Recipientes Plásticos

Recipiente Principal.

Se uso un recipiente principal en el que albergó el ESP32, el módulo de relé y el cableado. Este contenedor ofreció protección frente a la humedad y polvo, preservando la integridad de los componentes electrónicos.

Recipiente de Agua.

Otro recipiente, que sirvió como depósito de agua, donde se instalaron con las bombas sumergibles.

Elementos Adicionales

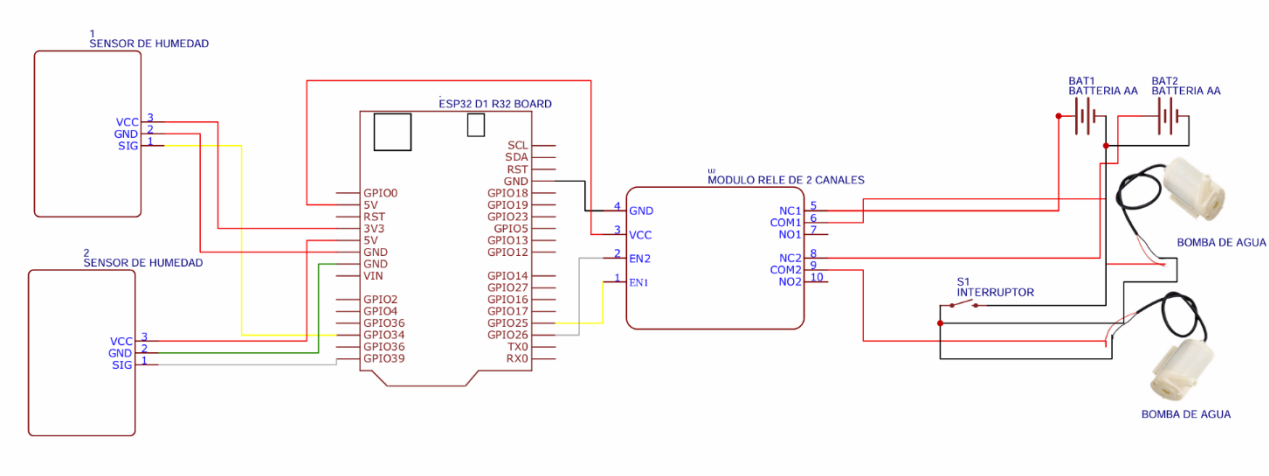
Se utilizo, 2 tipos de adhesivos, cinta aislante y cinta acrílica, se utilizó un interruptor para el apagado y encendido manual de las bombas de agua, y se usaron herramientas para soldar los cables que van conectados a las baterías y a la fuente de poder de estas mismas

Diseño del circuito

A continuación, se desglosan las conexiones y especificaciones del circuito para su implementación práctica:

Figura 6.

Diagrama del circuito de autorriego con ESP32. El diseño incluye sensores de humedad, un módulo de relés y bombas de agua para automatizar el riego. Elaborado en EasyEDA.



Sensores de humedad del suelo

Sensor Numero 1

- VCC: Conectado al pin 3v3 del ESP32
- GND: Conectado al pin GND del ESP32
- SIG: Conectado al pin GPIO36 del ESP32 (entrada analógica).

Sensor Numero 2

- VCC: Conectado al pin 3.3V del ESP32.
- GND: Conectado al GND del ESP32.
- SIG: Conectado al pin GPIO39 del ESP32 (entrada analógica).

Modulo relé de 2 Canales

El módulo relé permite controlar las bombas de agua mediante las señales digitales enviadas por el ESP32.

- Pin GND: Conectado al GND del ESP32.
- Pin VCC: Conectado al pin 5V del ESP32.
- Pin IN1: Conectado al GPIO25 del ESP32 (control del relé 1).
- Pin IN2: Conectado al GPIO26 del ESP32 (control del relé 2).

Bombas de agua

Las bombas se alimentan con baterías independientes para garantizar suficiente potencia. Cada una de las bombas de agua está conectada a su paquete de baterías AA, en donde los polos negativos de los paquetes de batería y el de los bombas de agua se encuentran concetados a un interruptor, y ste interruptor actúa para cerrar or abrir el paso de energía de las baterías a las bombas de agua, también una forma con la que se puede apagar las bombas de agua de manera manual si en algún momento se necesita hacer esto o el ESP32 se encuentra sin energía, y los polos positivos de los paquetes de batería van de la siguiente manera:

Bomba Numero 1

- Alimentada por un pack de baterías AA.
- Controlada por el relé 1 del módulo.
- COM1 (relé 1): Conectado al terminal positivo del pack de baterías.
- NO1 (relé 1): Conectado al polo positivo de la bomba.

Bomba Numero 2

- Alimentada por otro pack de baterías AA.
- Controlada por el relé 2 del módulo.
- COM2 (relé 2): Conectado al terminal positivo del pack de baterías.

- NO2 (relé 2): Conectado al polo positivo de la bomba.

Ensamblaje físico

Para realizar el ensamblaje del circuito, se siguieron los siguientes pasos de manera estructurada, asegurando la funcionalidad y la protección de los componentes:

Preparación del contenedor

Se colocaron cuidadosamente el ESP32 el módulo de relés y 2 pares de baterías AA dentro de un contenedor plástico diseñado para proteger los componentes electrónicos. Posteriormente, se realizaron agujeros estratégicos en el contenedor para permitir la salida de los cables necesarios, para el funcionamiento de los sensores, fuente de alimentación del ESP32 y la salida de los 4 cables necesarios para las bombas de agua

Conexión de las bombas de agua

Las bombas de agua se conectaron de la siguiente manera:

- Los polos positivos de las bombas se conectaron al puerto central (COM) del relé correspondiente.
- Los polos positivos de las baterías de las bombas se conectaron al puerto normalmente cerrado (NC) del relé respectivo.
- Los polos negativos de las bombas y las baterías se conectaron a través de un interruptor común, para permitir un control adicional sobre el suministro eléctrico.
- A las salidas de las bombas, se conectaron mangueras flexibles que se extendieron desde el contenedor plástico hacia las plantas

Integración del Módulo de Relés al ESP32

El módulo de relés se conectó al ESP32 mediante los pines 25 y 26, correspondientes a las salidas de control de las bombas de agua. Las conexiones de alimentación del módulo se realizaron a los pines GND y 5V del ESP32.

Conexión de los sensores al ESP32

Los sensores de humedad se conectaron al ESP32 a través de los pines analógicos 34 y 39. Además, los terminales de alimentación de los sensores se unieron a los pines 3.3V y GND del ESP32.

Instalación de las bombas y sensores

Las bombas de agua se introdujeron en el contenedor de almacenamiento de agua y se fijaron firmemente con cinta acrílica para evitar movimientos durante el funcionamiento. Los sensores de humedad se insertaron en las plantas junto a las mangeras, asegurando una posición estable para obtener mediciones confiables.

Programación del ESP32

La programación del ESP32 en este sistema permite la integración de hardware y software para automatizar el riego de dos plantas. A través de la conexión Wi-Fi y un servidor web, se ofrece una interfaz visual para monitorear en tiempo real los datos de humedad del suelo y el estado de las bombas de agua.

Características Principales del código

Conexión Wifi:

El ESP32 se conecta a una red inalámbrica utilizando las credenciales proporcionadas. Esto permite que el dispositivo hospede un servidor web accesible desde cualquier navegador en la misma red.

Lectura de Sensores:

Los valores de los sensores de humedad son leídos de forma continua. Estos valores son procesados para determinar el porcentaje de humedad, que se compara con un umbral previamente configurado.

Control de Bombas de Agua:

Basado en los niveles de humedad detectados, las bombas se activan o desactivan automáticamente para mantener las plantas en condiciones óptimas.

Interfaz Web Dinámica:

El servidor web proporciona una página que muestra en tiempo real los valores de humedad de los sensores, el porcentaje de humedad y el estado de las bombas de agua. Los datos se actualizan automáticamente mediante solicitudes asíncronas.

Estructura del código

El código está dividido en las siguientes secciones:

Definición de Pines y Variables Globales

Se definen los pines utilizados para los sensores y los relés, además de las constantes y variables necesarias para el cálculo y monitoreo.

Conexión a la Red Wi-Fi

En el setup(), el ESP32 se conecta a la red Wi-Fi utilizando las credenciales proporcionadas.

Servidor Web

El servidor web embebido escucha en el puerto 80 y define rutas específicas para manejar las solicitudes de los clientes.

Procesamiento de Sensores y Control de Relés

En el loop(), se realiza la lectura de los sensores, el cálculo del porcentaje de humedad y el control de las bombas según los valores leídos.

Página Web

La función paginaWeb() genera una página HTML que muestra los datos del sistema y utiliza JavaScript para actualizaciones automáticas.

Manejo de Solicitudes Asíncronas

La función actualizacionInformacion() devuelve los datos actuales en formato JSON para actualizar dinámicamente la interfaz.

Código Completo

```
#include <WiFi.h>   // Permite Conectar el ESP32 a una red WiFi
#include <WebServer.h>  // Permite la creacion de un servidor en el ESP32

// Credenciales WiFi - Deben ser remplasadas por las credenciales de la red WiFi a la que se quiere conectar el ESP32
const char* ssid = "ElNombredeTuRedWiFi";
const char* password = "LaContraseniadeTuRedWiFi";

// configuracion de servidor web
WebServer server(80); // servidor web en el puerto 80

// Definir pines
#define SENSOR_PIN_1 34    // Sensor de Humedad 1
#define RELAY_PIN_1 25     // Relé 1

#define SENSOR_PIN_2 39    // Sensor de Humedad 2
#define RELAY_PIN_2 26     // Relé 2

// umbrales de humedad - Entre mas bajo el valor, mas alta la humedad
int umbralHumedadSensor1 = 2500;  // el valor es 2500 debido a que este representa el nivel optimo de humedad en las plantas, el cual ronda entre un 50 y 60 porciento
int umbralHumedadSensor2 = 2500;  // se utiliza un umbral diferente debido a si se puede requerir un nivel de humedad diferente para la planta numero 2

// Variables para guardar el estado de los componentes
int sensorValue1 = 0;
int sensorValue2 = 0;
int porcentajeHumedad1 = 0;
int porcentajeHumedad2 = 0;
bool bomba1Estado = false;
bool bomba2Estado = false;

// constantes para los valores maximos y minimos de humedad
const int Mojado = 1500; // Valor de los sensores en agua
const int Seco = 4000; // Valor de los sensores en seco y en un dia soleado
```

```
// Comienza la comunicacion serial y configura los pines
void setup() {
  Serial.begin(115200);

  // Configura pines de el rele
  pinMode(RELAY_PIN_1, OUTPUT);
  digitalWrite(RELAY_PIN_1, HIGH);  // Rele empieza apagado

  pinMode(RELAY_PIN_2, OUTPUT);
  digitalWrite(RELAY_PIN_2, HIGH);  // Rele empieza apagad

  // Conectarse al WiFi con las credenciales proporcionadas
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Direccion IP: ");
  Serial.println(WiFi.localIP());

  // Definir la ruta de el servidor
  server.on("/", paginaWeb); // Define como manejar la ruta raiz
  server.on("/Actualizar", actualizacionInformacion); // Define como gestionar la ruta status

  // Arranca el servidor
  server.begin();
}

void loop() {
  // Gestionar solicitudes de el cliente - Solicitudes HTTP entrantes
  server.handleClient();

  // Lee y procesa el primer sensor
  sensorValue1 = analogRead(SENSOR_PIN_1);  // Lee el valor de la humedad de el primer sensor
  porcentajeHumedad1 = map(sensorValue1, Mojado, Seco, 100, 0); // Calcula el porcentaje de
que tan Humeda esta la planta

  if (sensorValue1 > umbralHumedadSensor1) {  // Si elvalor de la humedad es mayor que el
umbral
    digitalWrite(RELAY_PIN_1, HIGH);  // Activa la bomba de agua
    bomba1Estado = true; // Actualiza la variable de estado de la bomba
  } else {
    digitalWrite(RELAY_PIN_1, LOW);
    bomba1Estado = false;
  }

  // Lee y procesa el segundo sensor
  sensorValue2 = analogRead(SENSOR_PIN_2);  // Ocurre el primer proceso de elprimer sensor
  porcentajeHumedad2 = map(sensorValue2, Mojado, Seco, 100, 0);

  if (sensorValue2 > umbralHumedadSensor2) {
    digitalWrite(RELAY_PIN_2, HIGH);
    bomba2Estado = true;
  } else {
    digitalWrite(RELAY_PIN_2, LOW);
    bomba2Estado = false;
  }

  delay(3000);
```

```
}

// Disenio pagina web
void paginaWeb() { // Genera una pagina web usando HTML donde se muestra los valores actuales
de los sensores y el estado de las bombas de agua
  String html = "<!DOCTYPE html><html><head>"
                "<meta charset='UTF-8'>"
                "<title>Sistema de Autorriego con ESP32</title>"
                "<meta name='viewport' content='width=device-width, initial-scale=1'>"
                "<style>"
                "body{font-family:Arial;text-align:center;max-width:600px;margin:0
auto;padding:20px;}"
                "h1{color:#2c3e50;}"
                ".datosSensor{background:#f1f1f1;padding:15px;margin:10px 0;border-
radius:5px;}"
                ".status-on{color:green;}"
                ".status-off{color:red;}"
                "</style>"
                "</head><body>"
                "<h1>Sistema de Autorriego con ESP32</h1>"
                "<div class='datosSensor'>"
                "<h2>Planta 1</h2>"
                "<p>Valor de Humedad: <span id='humedad1'>" + String(sensorValue1) +
"</span></p>"
                "<p>Porcentaje de Humedad: <span id='porcentaje1'>" +
String(porcentajeHumedad1) + "%</span></p>"
                "<p>Estado de Bomba: <span id='bomba1' class='status-" + (bomba1Estado ?
"on'>Encendida" : "off'>Apagada") + "</span></p>"
                "</div>"
                "<div class='datosSensor'>"
                "<h2>Planta 2</h2>"
                "<p>Valor de Humedad: <span id='humedad2'>" + String(sensorValue2) +
"</span></p>"
                "<p>Porcentaje de Humedad: <span id='porcentaje2'>" +
String(porcentajeHumedad2) + "%</span></p>"
                "<p>Estado de Bomba: <span id='bomba2' class='status-" + (bomba2Estado ?
"on'>Encendida" : "off'>Apagada") + "</span></p>"
                "</div>"
                "<script>"
                "function actualizarDatos() {"
                "  fetch('/Actualizar')"
                "    .then(response => response.json())"
                "    .then(data => {"
                "      document.getElementById('humedad1').textContent = data.humedad1;"
                "      document.getElementById('porcentaje1').textContent = data.porcentaje1 +
'%';"
                "      document.getElementById('humedad2').textContent = data.humedad2;"
                "      document.getElementById('porcentaje2').textContent = data.porcentaje2 +
'%';"
                "      document.getElementById('bomba1').textContent = data.bomba1Estado ?
'Encendida' : 'Apagada';"
                "      document.getElementById('bomba1').className = 'status-' +
(data.bomba1Estado ? 'on' : 'off');"
                "      document.getElementById('bomba2').textContent = data.bomba2Estado ?
'Encendida' : 'Apagada';"
                "      document.getElementById('bomba2').className = 'status-' +
(data.bomba2Estado ? 'on' : 'off');"
                "    });"
                "}"
                "setInterval(actualizarDatos, 3000);"
                "</script>"
```

```
                "</body></html>";

  server.send(200, "text/html", html);
}

// Manejo de la ruta /Actualizar
void actualizacionInformacion() {
  String jsonResponse = "{\"humedad1\":" + String(sensorValue1) +
                        ",\"porcentaje1\":" + String(porcentajeHumedad1) +
                        ",\"humedad2\":" + String(sensorValue2) +
                        ",\"porcentaje2\":" + String(porcentajeHumedad2) +
                        ",\"bomba1Estado\":" + (bomba1Estado ? "true" : "false") +
                        ",\"bomba2Estado\":" + (bomba2Estado ? "true" : "false") + "}";

  server.send(200, "application/json", jsonResponse);
}
```

Este código aprovecha las capacidades del ESP32 para combinar monitoreo local (lecturas de sensores) y accesibilidad remota (interfaz web). Se pueden realizar ajustes en los umbrales de humedad y personalizar la página web para mejorar la interacción con el usuario. Es importante verificar los valores de los sensores en campo para calibrar correctamente el sistema.

Pruebas y Resultados

El proceso de pruebas se desarrolló de manera estructurada para garantizar el correcto funcionamiento de cada componente del sistema. Estas pruebas incluyeron desde la compilación inicial del código hasta la evaluación completa del desempeño del sistema integrado.

Compilación del Código y Conexión WiFi

Se inició compilando el código en el entorno de desarrollo del ESP32, verificando que no se generaran errores. Posteriormente, se configuró la conectividad Wifi del dispositivo utilizando las credenciales de red correspondientes. Se validó la conexión y se comprobó la obtención de una dirección IP, confirmando el enlace exitoso con la red.

Verificación de la Interfaz Web

Con la conectividad establecida, se accedió a la interfaz web generada por el ESP32. Esta prueba permitió verificar que las rutas definidas (e.g., "/", "/Actualizar") mostraran los datos esperados.

Conexión de Sensores y Relés

Se procedió a conectar los sensores de humedad al ESP32 y los módulos relé a las bombas de agua. Para validar el funcionamiento, se diseñó un entorno controlado donde se simulaban diferentes niveles de humedad en el suelo.

Evaluación del Funcionamiento de las Bombas de Agua

Tras confirmar la respuesta de los relés, se conectaron las bombas de agua y las baterías al módulo relé. En esta etapa se verificó que las bombas se activaran o desactivaran

Calibración de los Sensores

Figura 7

Pruebas con los sensores

Para determinar los valores máximos y mínimos de los sensores, se realizaron las siguientes pruebas:

- **Humedad máxima:** Se introdujo un sensor en un vaso con agua, obteniendo un valor aproximado de 1600.
- **Humedad mínima:** Se colocó un sensor en un entorno seco y soleado, registrando un valor aproximado de 4000.

Figura 8

Resultados

Estos valores se usaron para ajustar las constantes del código, permitiendo un cálculo preciso del porcentaje de humedad.

Ajuste del Umbral de Activación

Con los datos obtenidos, se determinó un valor de 2500 como umbral para la activación de las bombas de agua. Este valor, correspondiente al 50 % de humedad, se seleccionó teniendo en cuenta que el rango óptimo de humedad para plantas comunes oscila entre 50 % y 60 %

Pruebas de Integración

Finalmente, se llevó a cabo una prueba del sistema completo. Esta incluyó monitorear los sensores en tiempo real, verificar la activación de las bombas de agua de acuerdo con el umbral establecido, y observar la actualización de los datos en la interfaz web.

Resultados

- Los sensores mostraron precisión al detectar los niveles de humedad.
- Las bombas de agua se activaron correctamente al superar el umbral configurado.
- La interfaz web funcionó sin interrupciones y con actualizaciones en tiempo real.
- No se detectaron problemas en la conectividad WiFi ni en la comunicación entre los componentes.

Conclusión

El desarrollo del sistema de autorriego basado en el ESP32 cumplió con los objetivos planteados al inicio del proyecto. La integración de sensores de humedad, módulos de relé, bombas de agua y la conectividad Wi-Fi permitió crear un sistema automatizado capaz de medir los niveles de humedad del suelo y accionar el riego de manera eficiente.

A través de pruebas, se demostró que el sistema es funcional y estable, con una respuesta precisa a las condiciones de humedad detectadas. La calibración de los sensores y el ajuste de los umbrales garantizaron que el sistema proporcionara riego en los niveles óptimos para plantas comunes, promoviendo el uso eficiente del agua. Además, la interfaz web embebida ofreció una experiencia interactiva y accesible, permitiendo al usuario monitorear y supervisar el sistema en tiempo real.

El diseño del sistema se caracteriza por su escalabilidad y adaptabilidad, lo que lo convierte en una solución versátil para diferentes contextos, desde el cuidado de plantas domésticas hasta aplicaciones en agricultura urbana. Sin embargo, se identificaron oportunidades

de mejora, como la inclusión de notificaciones remotas o el uso de energías renovables para alimentar el sistema, que podrían explorarse en futuros desarrollos.

En conclusión, este proyecto demuestra cómo la tecnología puede facilitar la gestión de recursos en el hogar y contribuir a la sostenibilidad, brindando una herramienta práctica para el cuidado eficiente de las plantas.

Referencias

Arduino - HTTP Request | Arduino getting started. (n.d.). Arduino Getting Started. https://arduinogetstarted.com/tutorials/arduino-http-request

Bas on Tech. (2020, March 28). *SOIL MOISTURE SENSOR FOR PLANTS (V1.2 / V2.0) - Arduino tutorial #31* [Video]. YouTube. https://www.youtube.com/watch?v=pFQaFnqpOtQ

EasyEDA(Standard) - a simple and powerful electronic circuit design tool. (n.d.). https://easyeda.com/editor

Emilostuff. (2020, December 10). *Preventing plant death with technology* [Video]. YouTube. https://www.youtube.com/watch?v=9Fx9zQJe3H4

ESP32 - Web Server | ESP32 Tutorial. (n.d.). ESP32 Tutorial. https://esp32io.com/tutorials/esp32-web-server

espressif. (n.d.-a). *arduino-esp32/libraries/WebServer/src/WebServer.h at master · espressif/arduino-esp32*. GitHub. https://github.com/espressif/arduino-esp32/blob/master/libraries/WebServer/src/WebServer.h

espressif. (n.d.-b). *arduino-esp32/libraries/WiFi/src/WiFi.h at master · espressif/arduino-esp32*. GitHub. https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFi.h

Flaura - Smart Plant Pot. (2021a, September 3). *Parts list for Flaura – The Smart, self watering plant Pot (DIY project, 3D printed, Arduino)* [Video]. YouTube. https://www.youtube.com/watch?v=GNNzLwCidCE

Flaura - Smart Plant Pot. (2021b, October 3). *Capacitive Soil Moisture Sensors don't work correctly + Fix for v2.0 v1.2 Arduino ESP32 Raspberry Pi* [Video]. YouTube. https://www.youtube.com/watch?v=IGP38bz-K48

FreeCodeCamp.org. (n.d.-a). https://www.freecodecamp.org/learn/2022/responsive-web-design/

FreeCodeCamp.org. (n.d.-b). https://www.freecodecamp.org/certification/oclayos/responsive-web-design

M, J. (2022, December 24). The Best Humidity Level for Plants (Plus How to Achieve It!). *GreenUpSide*. https://greenupside.com/what-is-the-best-humidity-level-for-plants/

map() - Arduino Reference. (n.d.). https://reference.arduino.cc/reference/en/language/functions/math/map/

Mazen Salah. (2023, December 18). *How to read data from a JSON file into HTML | json to html js 2024* [Video]. YouTube. https://www.youtube.com/watch?v=w1Oz0Sj1QyQ

Practical Engineering. (2015, April 2). *Arduino Garden Controller - automatic watering and data logging* [Video]. YouTube. https://www.youtube.com/watch?v=O_Q1WKCtWiA

Santos, R., & Santos, R. (2019, April 2). *ESP32 Web Server - Arduino IDE | Random Nerd Tutorials*. Random Nerd Tutorials. https://randomnerdtutorials.com/esp32-web-server-arduino-ide/

Science & Fun Club. (2019, July 20). *How to connect switch with LED, 9v battery - DIY switch light tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=LCeGB2ejQQQ

Thingiverse.com. (n.d.). *PLANT KEEPER – Automated Plant Watering System by Emilostuff*. Thingiverse. https://www.thingiverse.com/thing:4681068

W3Schools.com. (n.d.-a). https://www.w3schools.com/whatis/whatis_json.asp

W3Schools.com. (n.d.-b). https://www.w3schools.com/html/html_css.asp