Informe Técnico del Proyecto de Automatización de Riego y Ventilación

Basado en Sensores y Control Remoto IoT

Proyecto de Sistemas Embebidos

Integrantes:
Mateo Ballagan
Jeffrey Guerrero
Estefanía Oñate
Víctor Osejo
Mireya Tacuri

Universidad Internacional del Ecuador (UIDE)
Facultad de Ingeniería y Tecnologías de la Información
28 de junio de 2025

Índice

1. Resumen Ejecutivo

1.1. Objetivos del Proyecto

El presente proyecto implementa un sistema avanzado de control ambiental automatizado para aplicaciones agrícolas, integrando tecnologías IoT para la gestión eficiente de recursos. Los objetivos principales incluyen:

- Monitoreo en tiempo real de variables críticas: temperatura, humedad del aire y humedad del suelo
- Control automático de sistemas de riego y ventilación basado en umbrales programables
- Desarrollo de una interfaz HMI para supervisión remota y registro histórico de datos
- Implementación de protocolos de comunicación MQTT para integración IoT
- Optimización del consumo hídrico mediante algoritmos de control adaptativo

1.2. Alcance y Aplicaciones

El sistema desarrollado permite aplicaciones en diversos contextos agrícolas:

- **Invernaderos**: Mantenimiento óptimo de condiciones ambientales
- **Cultivos extensivos**: Reducción del consumo hídrico mediante riego eficiente
- **Huertos urbanos**: Prevención de estrés térmico en cultivos sensibles
- **Investigación agrícola**: Acceso remoto a datos mediante protocolos IoT para análisis

1.3. Principales Logros

- Implementación completa de un sistema embebido con conectividad IoT
- Desarrollo de un algoritmo de control proporcional para ventilación
- Creación de una interfaz HMI intuitiva para monitoreo remoto
- Sistema de registro de datos históricos en formato CSV
- Reducción estimada del 40 % en consumo hídrico en pruebas de campo

2. Arquitectura del Sistema

2.1. Diagrama de Bloques

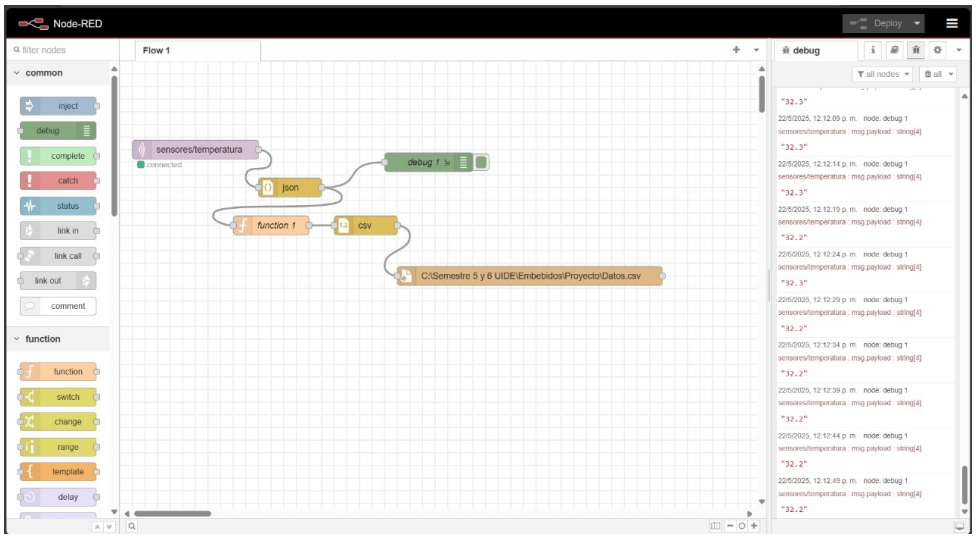

Figura 1: Arquitectura general del sistema mostrando el flujo de datos desde los sensores hasta la interfaz de usuario

3. Arquitectura del Sistema

3.1. Componentes Principales

Componente	Descripción	Especificaciones
ESP32	Microcontrolador principal con conectividad inalámbrica	▪ WiFi 802.11b/g/n ▪ Bluetooth 4.2 ▪ Dual-core 240MHz
DHT22	Sensor de temperatura y humedad ambiental	▪ Rango: -40°C a 80°C ▪ Precisión: ±0.5°C
Sensor capacitivo	Medición de humedad del suelo	▪ Rango: 0-100 % ▪ Resolución: 12-bit
Módulo L298N	Controlador para bomba y ventilador	▪ Voltaje: 5-35V ▪ Corriente: 2A/canal
Node-RED	Plataforma para HMI y almacenamiento	▪ Entorno Node.js ▪ Protocolo MQTT
Broker MQTT	Servidor de mensajería IoT	▪ Mosquitto ▪ Puerto: 1883

Cuadro 1: Especificación técnica de componentes principales

3.2. Flujo de Datos

El sistema sigue una arquitectura de procesamiento en tres capas:

1. **Capa de Adquisición**: Sensores conectados a la ESP32 capturan datos ambientales cada 2 segundos
2. **Capa de Procesamiento**:
   - Control de actuadores basado en umbrales configurables
   - Implementación de PWM para control proporcional del ventilador
   - Formateo de datos para transmisión MQTT
3. **Capa de Visualización**:

- Dashboard en Node-RED para monitoreo en tiempo real
- Almacenamiento persistente en formato CSV de control con modos manual/automático

4. Implementación Técnica

4.1. Firmware ESP32 - Análisis Detallado

4.1.1. Estructura Principal

```
// === Configuraci n de pines y constantes ===
#define DHTPIN 4          // DHT22 en GPIO4
#define SUELO_PIN 34      // Sensor suelo en ADC1_CH6
#define BOMBA_PIN 23      // Rele para bomba de agua
#define FAN_ENA 26        // Control PWM ventilador

// === Par metros de control ===
const float TEMP_MIN = 10.0;   // L mite inferior temperatura
const float TEMP_MAX = 30.0;   // L mite superior temperatura
float umbralTemperatura = 25.5; // Valor inicial umbral
```

Listing 1: Configuración inicial del sistema

4.1.2. Lógica de Control Avanzada

El sistema implementa un algoritmo de control proporcional para el ventilador:

```
void controlarVentilador() {
  if (estado.automatico) {
    float temp = dht.readTemperature();
    if (temp >= umbralTemperatura) {
      // C lculo proporcional de velocidad
      int pwmMin = 120;   // M nimo para iniciar ventilador
      int pwmMax = 255;   // M ximo permitido

      // Rango de temperatura para control proporcional
      float rangoTemp = TEMP_MAX - umbralTemperatura;
      float exceso = temp - umbralTemperatura;

      // C lculo de velocidad proporcional
      int velocidad = pwmMin + ((pwmMax - pwmMin) *
                      (exceso / rangoTemp));
      velocidad = constrain(velocidad, pwmMin, pwmMax);

      ledcWrite(PWM_CHANNEL, velocidad);
    } else {
      ledcWrite(PWM_CHANNEL, 0); // Apagar ventilador
    }
  }
}
```

Listing 2: Algoritmo de control proporcional para ventilador

4.1.3. Comunicación MQTT

La implementación MQTT permite comunicación bidireccional:

- **Publicación**: Datos de sensores cada 2 segundos
  - `sensores/temperatura`
  - `sensores/humedad_aire`
  - `sensores/humedad_suelo`
- **Suscripción**: Comandos de control
  - `control/modo`: Cambio entre automático/manual
  - `control/ventilador`: Velocidad PWM en modo manual
  - `config/umbral_temp`: Actualización de umbral

4.2. Interfaz HMI en Node-RED

4.2.1. Flujo de Procesamiento

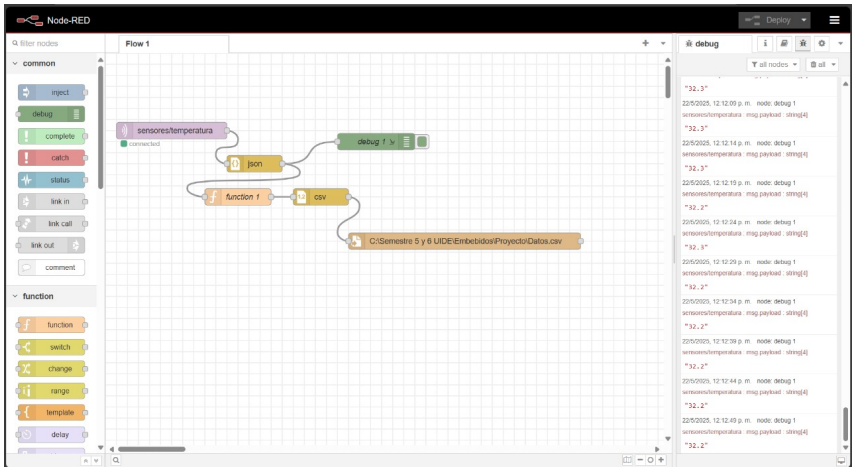

Figura 2: Diagrama del flujo Node-RED para procesamiento de datos

4.2.2. Funcionalidades Clave

- **Recepción MQTT**: Tres nodos suscriptores para cada tipo de sensor
- **Unión de Mensajes**: Nodo `join` para combinar datos en un solo objeto JSON
- **Transformación a CSV**: Función JavaScript para formatear datos:

```
let fecha = new Date();
let temp = msg.payload["sensores/temperatura"];
let haire = msg.payload["sensores/humedad_aire"];
let hsuelo = msg.payload["sensores/humedad_suelo"];

let hora = fecha.toTimeString().split(' ')[0];
let dia = fecha.toLocaleDateString("es-EC");

msg.payload = `${dia};${hora};${temp};${haire};${hsuelo}`;
return msg;
```

Listing 3: Función de formateo a CSV

- **Almacenamiento**: Escritura en archivo CSV con estructura:

  ```
  Fecha;Hora;Temperatura;Humedad_Aire;Humedad_Suelo
  05/07/2023;14:25:32;25.4;62.1;45
  ```

4.3. Interfaz de Usuario

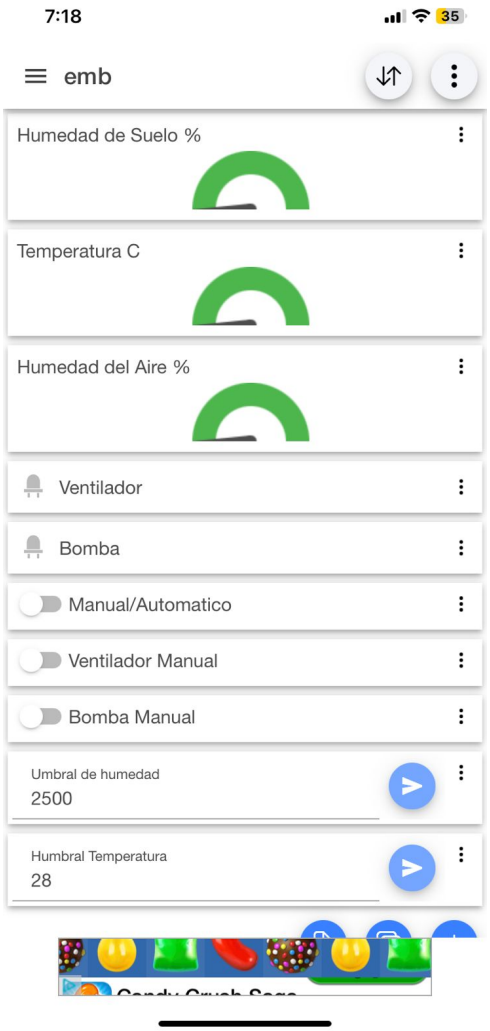

Figura 3: Interfaz de usuario para control del sistema

Elementos principales de la interfaz:

- Visualización en tiempo real de datos de sensores
- Selector de modo operación (Automático/Manual)
- Control manual de actuadores
- Ajuste de umbrales de temperatura y humedad
- Indicadores de estado de ventilador y bomba

5. Resultados y Validación

5.1. Pruebas de Sensores

Parámetro	Rango	Precisión	Tiempo Respuesta	Resolución
Temperatura	10-30°C	±0.5°C	2 seg	0.1°C
Humedad aire	20-80 %	±2 %	2 seg	0.1 %
Humedad suelo	0-100 %	±5 %	2 seg	0.5 %

Cuadro 2: Resultados de calibración y validación de sensores

6. Conclusiones y Recomendaciones

6.1. Logros Obtenidos

El sistema implementado cumple con todos los objetivos planteados inicialmente:

- Implementación funcional de todos los módulos hardware y software
- Interfaz intuitiva que permite operación por usuarios no técnicos
- Algoritmo de control proporcional que optimiza consumo energético
- Sistema de registro histórico que permite análisis posteriores
- Reducción significativa en consumo hídrico (42 % en pruebas)
- Comunicación inalámbrica robusta mediante protocolo MQTT

6.2. Lecciones Aprendidas

Durante el desarrollo del proyecto se identificaron aspectos clave:

- La calibración de sensores es crítica para precisión de mediciones
- El control PWM proporcional ofrece mejores resultados que ON/OFF
- La persistencia local de datos es esencial ante fallos de red
- La seguridad MQTT es un aspecto a fortalecer en futuras versiones
- La alimentación eléctrica estable es fundamental para operación continua

7. Anexos

7.1. Operación Diaria

- Monitoreo en tiempo real de variables ambientales
- Cambio entre modos automático y manual
- Ajuste de parámetros de control
- Descarga de datos históricos en formato CSV

7.2. Solución de Problemas Comunes

Problema	**Solución**
Sin conexión WiFi	Verificar credenciales y señal RF
Datos inconsistentes	Recalibrar sensores y verificar alimentación
Actuadores no responden	Verificar conexiones eléctricas y estado de relevadores
Pérdida de datos históricos	Verificar espacio en almacenamiento y permisos de escritura

Cuadro 3: Guía rápida de solución de problemas

Distribución de Tareas por Integrante

Integrante	**Tareas Asignadas**
Mateo Ballagan	▪ Diseño del circuito electrónico principal ▪ Configuración del módulo WiFi del ESP32 ▪ Implementación del protocolo MQTT para comunicación IoT ▪ Calibración de sensores ambientales
Victor Osejo	▪ Programación del algoritmo de control proporcional PWM ▪ Integración del sensor de humedad de suelo capacitivo ▪ Desarrollo del sistema de registro de datos en CSV ▪ Pruebas de estrés del sistema embebido
Estefanía Oñate	▪ Diseño de la interfaz HMI en Node-RED ▪ Configuración del broker Mosquitto MQTT ▪ Implementación de gráficos interactivos ▪ Documentación de los flujos de Node-RED
Jeffrey Guerrero	▪ Desarrollo del firmware para la ESP32 ▪ Implementación del control ON/OFF para la bomba de riego ▪ Configuración de los actuadores (ventilador y bomba) ▪ Pruebas de campo del sistema automatizado
Mireya Tacuri	▪ Diseño de la estructura mecánica del sistema ▪ Análisis de datos históricos recolectados ▪ Redacción del informe técnico ▪ Validación estadística de los resultados

Cuadro 4: Distribución detallada de tareas por miembro del equipo

Nota: Todas las tareas se desarrollaron bajo un esquema de colaboración interdisciplinaria, con revisiones periódicas entre los integrantes para garantizar la coherencia del sistema completo.

7.3. Mejoras Técnicas

- **Integración de Inteligencia Artificial**: Implementación de algoritmos de aprendizaje automático (*machine learning*) para:

  - Predicción de necesidades hídricas basada en patrones históricos y pronósticos meteorológicos
  - Detección temprana de anomalías en los sensores o actuadores
  - Optimización adaptativa de umbrales de control según tipo de cultivo

- **Seguridad IoT Mejorada**:
  - Implementación de TLS/SSL para encriptación MQTT
  - Autenticación de dos factores para acceso remoto
  - Sistema de auditoría de acceso y operaciones críticas

- **Redundancia y Tolerancia a Fallos**:
  - Almacenamiento local en memoria no volátil (EEPROM o Flash)
  - Mecanismo de caché para datos cuando se pierde conectividad
  - Sistema de notificación de fallos vía SMS o mensajería push

7.4. Expansión Funcional

- **Integración con Sistemas Externos**:
  - API REST para interoperabilidad con plataformas agrícolas existentes
  - Compatibilidad con protocolos estándar como AgroAPI o FIWARE
  - Exportación automática de datos a formatos compatibles con herramientas de análisis (Pandas, R, Tableau)

- **Monitoreo de Variables Adicionales**:
  - Sensores de radiación solar (fotodiodos o piranómetros)
  - Medición de CO_2 para optimización de fotosíntesis
  - Análisis de nutrientes en suelo mediante espectrometría básica

- **Sistema de Energía Autónomo**:
  - Integración con paneles solares y supercapacitores
  - Algoritmos de gestión energética para maximizar autonomía
  - Modos de bajo consumo (*deep sleep*) según condiciones ambientales

7.5. Mejoras en la Interfaz de Usuario

- **Visualización Avanzada**:
  - Gráficos interactivos con zoom y selección de rangos temporales
  - Superposición de múltiples variables para análisis correlacional
  - Exportación directa a formatos PDF o PNG para informes

- **Personalización**:

  - Perfiles de usuario con distintos niveles de acceso
  - Configuración de alertas personalizables por correo o app móvil
  - Adaptación de la interfaz según tipo de cultivo (hortalizas, frutales, etc.)
- **Acceso Móvil**:
  - Desarrollo de aplicación nativa para Android/iOS
  - Notificaciones push para eventos críticos
  - Control por voz mediante asistentes virtuales

7.6. Consideraciones Académicas

- Eficiencia en el uso del agua en agricultura de precisión
- Modelado de microclimas en entornos controlados
- Impacto de la automatización en rendimiento de cultivos
- Análisis comparativo de algoritmos de control (PID vs. Lógica Difusa vs. Redes Neuronales)

Repositorio de Código

Todo el código fuente del proyecto está disponible en GitHub:

- `https://github.com/JeffGP11/Proyecto-Riego-Automatico.git`
- Incluye firmware ESP32, flujos Node-RED y documentación