Universidad Mariano Gálvez de Guatemala

Centro Universitario de Puerto Barrios

Facultad de Ingeniería en Sistemas

Autómatas y Lenguajes Formales

Ing. Erick Estuardo Alvarez Ramirez

Diseño e Implementación de una Estación Meteorológica loT con Monitoreo en Tiempo Real Mediante Arduino UNO R4 WiFi

Proyecto Final – Fase Producto Final de Software-Hardware

Integrantes:

Yozef Leonidas Emmanuel Dueñas Sagastume 0909-23-616

Erick Alejandro Marroquín González 0909-23-6928

Mike Jason René Martínez Barrientos 0909-22-15059

Gesler Baldemar Hernández Pérez 0909-22-951

Puerto Barrios, Izabal, 25 de octubre de 2025



<https://www.canva.com/design/DAG2hTQfsq0/LuX7TR0E-yGHP-FoL7DPSQ/edit?utm_content=DAG2hTQfsq0&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>

Meteor Force

**Índice**

[Introducción 1](#_Toc212241109)

[Antecedentes 3](#_Toc212241110)

[Justificacion 5](#_Toc212241111)

[Alcances 7](#_Toc212241112)

[Perspectiva Investigativa 7](#_Toc212241113)

[Perspectiva Técnica 7](#_Toc212241114)

[Perspectiva de Resultados Esperados 7](#_Toc212241115)

[Marco Teórico 9](#_Toc212241116)

[Autómata Finito (AFD) 9](#_Toc212241117)

[Estados 9](#_Toc212241118)

[Transiciones 9](#_Toc212241119)

[Acciones o Salidas 10](#_Toc212241120)

[Arduino UNO R4 WiFi 10](#_Toc212241121)

[Sensores 10](#_Toc212241122)

[Comunicación IoT 10](#_Toc212241123)

[Sistema de Información 10](#_Toc212241124)

[Hardware 10](#_Toc212241125)

[Software 11](#_Toc212241126)

[Lista de Materiales 11](#_Toc212241127)

[Guía de Conexiones 12](#_Toc212241128)

[Base de Datos 12](#_Toc212241129)

[Normalización 13](#_Toc212241130)

[Modelo Entidad–Relación 14](#_Toc212241131)

[Diagrama Conceptual 14](#_Toc212241132)

[Esquema de Tablas (Independientes) 15](#_Toc212241133)

[Cardinalidad y Relaciones 18](#_Toc212241134)

[Bases de datos 19](#_Toc212241135)

[Estructura de Base de Datos 19](#_Toc212241136)

[Tabla: lecturas\_humedad 20](#_Toc212241137)

[Tabla: lecturas\_presion 20](#_Toc212241138)

[Tabla: lecturas\_lluvia 21](#_Toc212241139)

[Tabla: lecturas\_humedad\_suelo 21](#_Toc212241140)

[Tabla: lecturas\_gas 22](#_Toc212241141)

[Vista: vista\_ultima\_lectura 22](#_Toc212241142)

[Expresiones Regulares 25](#_Toc212241143)

[Backend (server.js) 25](#_Toc212241144)

[Temperatura 25](#_Toc212241145)

[Humedad 26](#_Toc212241146)

[Presion 26](#_Toc212241147)

[Lluvia 27](#_Toc212241148)

[Valores Analogicos (Humedad Suelo, Gas) 27](#_Toc212241149)

[Fecha ISO 8601 28](#_Toc212241150)

[Rango Numerico 29](#_Toc212241151)

[Frontend (app.js) 30](#_Toc212241152)

[Fecha (Frontend) 30](#_Toc212241153)

[Rango (Frontend) 30](#_Toc212241154)

[Numero Positivo 30](#_Toc212241155)

[Validacion en Tiempo Real (Frontend) 30](#_Toc212241156)

[Marco Metodológico 32](#_Toc212241157)

[Tipo y Nivel de Investigación 32](#_Toc212241158)

[Diseño de la Investigación 32](#_Toc212241159)

[Indicadores para los Objetivos Específicos 33](#_Toc212241160)

[Población y Muestra 34](#_Toc212241161)

[Técnicas e Instrumentos de Recolección de Datos 34](#_Toc212241162)

[Técnicas de Procesamiento y Análisis de Datos 35](#_Toc212241163)

[Presentación De Resultados 36](#_Toc212241164)

[1. Resultados del Sistema Físico 36](#_Toc212241165)

[2. Resultados del Software y Comunicación IoT 36](#_Toc212241166)

[3. Resultados del Sistema de Información y Visualización 37](#_Toc212241167)

[4. Validación de Expresiones Regulares 37](#_Toc212241168)

[5. Resultados Cuantitativos 38](#_Toc212241169)

[6. Evidencia del Funcionamiento 39](#_Toc212241170)

[7. Evaluación de los Objetivos 39](#_Toc212241171)

[Discusión de Resultados 41](#_Toc212241172)

[Impacto Organizacional 41](#_Toc212241173)

[1. Impacto Académico y Educativo 41](#_Toc212241174)

[2. Impacto Tecnológico 42](#_Toc212241175)

[3. Impacto Ambiental y Comunitario 42](#_Toc212241176)

[4. Impacto Organizacional Interno 42](#_Toc212241177)

[Análisis de Riesgos 43](#_Toc212241178)

[Modelo Canvas 43](#_Toc212241179)

[Implantación del Sistema 43](#_Toc212241180)

[1. Evaluación del Funcionamiento 44](#_Toc212241181)

[a) Evaluación Técnica 44](#_Toc212241182)

[b) Evaluación Funcional 45](#_Toc212241183)

[c) Evaluación Académica 45](#_Toc212241184)

[2. Resultados de la Evaluación 45](#_Toc212241185)

[Conclusiones 47](#_Toc212241186)

[Recomendaciones 50](#_Toc212241187)

[Bibliografía 52](#_Toc212241188)

[Anexos 53](#_Toc212241189)

**Símbolos**

| **Símbolo/Abreviatura** | **Significado** | **Unidad / Nota** |
| --- | --- | --- |
| IoT | Internet of Things (Internet de las cosas) | — |
| AFD | Autómata Finito Determinista | — |
| DHT22 | Sensor digital de temperatura y humedad | — |
| BMP280 | Sensor barométrico (presión) | hPa |
| MQ-2 | Sensor de gas (propano, butano, humo) | ppm |
| FC-28 / FC-37 / YL83 | Sensores de humedad del suelo / lluvia | — |
| API | Application Programming Interface | — |
| HTTP POST | Método de envío de datos por la API | — |
| DB / MySQL | Base de datos relacional MySQL | — |
| TTL | Time to Live (si se usa en comunicaciones) | seg / configurable |
| Hz | Hertz (frecuencia de muestreo, si aplica) | Hz |
| %HR | Humedad relativa | % |
| °C | Grados Celsius | °C |
| hPa | Hectopascal (presión atmosférica) | hPa |
| UUID | Identificador único universal | — |
| CSV | Comma-Separated Values | — |

**Glosario**

**AFD (Autómata Finito Determinista):**  
Modelo matemático que representa sistemas con un conjunto finito de estados y transiciones definidas; utilizado para modelar el flujo lógico del sistema de la estación meteorológica.

**Algoritmo:**  
Conjunto ordenado de pasos lógicos y precisos que permiten resolver un problema o realizar una tarea específica.

**API (Application Programming Interface):**  
Conjunto de funciones, rutas y protocolos que permiten la comunicación entre un cliente (Arduino) y un servidor (Node.js).

**Arduino UNO R4 WiFi:**  
Placa de desarrollo basada en el microcontrolador Renesas RA4M1, que incluye conectividad WiFi integrada para aplicaciones IoT.

**BMP280:**  
Sensor digital de presión barométrica y temperatura que comunica datos mediante el protocolo I2C.

**Buzzer:**  
Dispositivo electrónico que genera un sonido o alerta auditiva en respuesta a determinadas condiciones del sistema.

**Base de Datos (BD):**  
Conjunto organizado de información estructurada que se almacena y gestiona electrónicamente, en este caso mediante MySQL.

**Chart.js:**  
Librería de JavaScript utilizada para graficar datos en la interfaz web, permitiendo visualizar en tiempo real las variables meteorológicas.

**DHT22:**  
Sensor digital de temperatura y humedad que ofrece lecturas de alta precisión mediante una señal digital calibrada.

**Entidad:**  
Elemento o concepto dentro del modelo de base de datos que representa un objeto del mundo real (por ejemplo, “Registro de lectura”).

**Expresiones Regulares (RegEx):**  
Patrones utilizados para validar y filtrar texto, empleados en el servidor para garantizar que los datos JSON tengan formato correcto antes de almacenarse.

**FC-28:**  
Sensor de humedad del suelo que permite medir la cantidad de agua contenida en la tierra, útil para determinar la necesidad de riego.

**FC-37 / YL-83:**  
Sensor analógico de detección de lluvia, empleado para identificar la presencia de gotas de agua en el entorno.

**HTTP (Hypertext Transfer Protocol):**  
Protocolo de comunicación que permite la transferencia de datos entre el Arduino y el servidor web.

**Humedad Relativa (%HR):**  
Porcentaje que representa la cantidad de vapor de agua presente en el aire en comparación con la máxima que podría contener a esa temperatura.

**IoT (Internet of Things):**  
Conjunto de tecnologías que permiten conectar objetos físicos (sensores, dispositivos) a la red para recopilar, transmitir y analizar datos.

**JSON (JavaScript Object Notation):**  
Formato de intercambio de datos estructurado en pares clave-valor, empleado para transmitir información entre el Arduino y la API.

**LCD 16x2 (Liquid Crystal Display):**  
Pantalla alfanumérica que muestra datos en tiempo real, utilizada para visualizar lecturas locales de los sensores.

**Lenguajes Formales:**  
Conjunto de símbolos y reglas gramaticales que definen estructuras válidas en un sistema lógico o computacional.

**Microcontrolador:**  
Unidad electrónica programable que ejecuta instrucciones para controlar sensores, actuadores y otros periféricos.

**MQ-2:**  
Sensor análogo capaz de detectar gases inflamables y concentraciones de humo en el ambiente.

**MySQL:**  
Sistema de gestión de bases de datos relacional (RDBMS) utilizado para almacenar y organizar los datos obtenidos por la estación.

**Node.js:**  
Entorno de ejecución para JavaScript que permite desarrollar servidores web y procesar solicitudes de manera asincrónica.

**Protocolo I2C:**  
Método de comunicación serial de dos líneas (SDA y SCL) que permite conectar múltiples periféricos a un microcontrolador.

**Sensor:**  
Dispositivo que detecta cambios físicos o químicos en el entorno (como temperatura, humedad o presión) y los convierte en señales eléctricas.

**Servidor:**  
Equipo o programa que gestiona solicitudes y respuestas dentro de una red; en este proyecto recibe los datos del Arduino y los almacena.

**Temperatura:**  
Magnitud física que mide el nivel de calor de un cuerpo o ambiente; en este caso se mide en grados Celsius (°C).

**Transición:**  
Cambio de un estado a otro dentro de un autómata finito, provocado por una entrada o evento específico.

**WiFi:**  
Tecnología de red inalámbrica que permite la conexión del Arduino UNO R4 a Internet para enviar datos en tiempo real.

**Resumen**

El presente proyecto, denominado Meteor Force, consiste en el diseño e implementación de una estación meteorológica IoT orientada al monitoreo remoto en tiempo real de variables climáticas. El sistema fue desarrollado utilizando una placa Arduino UNO R4 WiFi, que permite la conexión inalámbrica y la transmisión de datos a través del protocolo HTTP hacia una API desarrollada en Node.js.

Dicha API realiza validaciones sintácticas mediante expresiones regulares y gestiona el almacenamiento de los datos en una base de datos MySQL, garantizando la integridad y disponibilidad de la información meteorológica. La estación integra sensores de temperatura, humedad, presión atmosférica, gas, lluvia y humedad del suelo, los cuales generan mediciones periódicas que son enviadas al servidor para su procesamiento y visualización.

Desde un enfoque teórico, el funcionamiento lógico del sistema se modela como un Autómata Finito Determinista (AFD), el cual define estados operativos según las condiciones detectadas por los sensores, asegurando respuestas coherentes ante diferentes escenarios ambientales.

La investigación se desarrolló bajo un enfoque aplicado y experimental, combinando el diseño de hardware y software con pruebas controladas tanto en laboratorio como en entornos reales. Los resultados obtenidos demostraron un desempeño estable, lecturas confiables y una visualización web efectiva de las variables monitoreadas.

Meteor Force evidencia la viabilidad de implementar soluciones tecnológicas de bajo costo basadas en IoT y teorías de los Lenguajes Formales, aportando al monitoreo ambiental, la educación tecnológica y el desarrollo local sostenible en Guatemala.

**Planteamiento Del Problema**

En Guatemala, el monitoreo climático confiable es fundamental para la agricultura, la gestión del agua y la prevención de desastres naturales. Sin embargo, la cobertura actual de estaciones meteorológicas en el país sigue siendo limitada y desigual. De acuerdo con el Instituto Nacional de Sismología, Vulcanología, Meteorología e Hidrología (INSIVUMEH), existen alrededor de 190 estaciones meteorológicas operativas, distribuidas principalmente en cabeceras departamentales. Esta cifra representa apenas un 56 % del objetivo nacional de contar con una estación automática en cada uno de los 340 municipios del país.

Esta cobertura parcial provoca una baja resolución espacial de los datos meteorológicos, especialmente en zonas rurales y costeras como Izabal, Alta Verapaz y Petén, donde las variaciones climáticas locales no se registran en tiempo real. Como consecuencia, las comunidades agrícolas, educativas y de investigación carecen de información precisa para planificar actividades productivas o responder ante eventos meteorológicos extremos.

A esta limitación se suma el alto costo y la complejidad técnica de las estaciones profesionales disponibles, las cuales requieren mantenimiento especializado y recursos económicos significativos. Estas condiciones reducen la posibilidad de ampliar la red meteorológica nacional, afectando directamente la toma de decisiones locales y la gestión ambiental.

Frente a esta problemática, se identifica la necesidad de desarrollar alternativas tecnológicas más accesibles, económicas y adaptables, que permitan el registro y transmisión de datos climáticos en tiempo real. Este tipo de herramientas podrían fortalecer el monitoreo ambiental, promover la educación tecnológica y mejorar la capacidad de respuesta ante variaciones meteorológicas locales.

**Objetivo General**

Diseñar e implementar una estación meteorológica IoT de bajo costo, basada en el microcontrolador Arduino UNO R4 WiFi, que permita la medición, transmisión y almacenamiento en tiempo real de variables climáticas locales, contribuyendo al fortalecimiento del monitoreo ambiental y educativo en Guatemala.

**Objetivos Específicos**

1. **Integrar** sensores digitales de temperatura, humedad, presión atmosférica, gas, lluvia y humedad del suelo en un sistema modular basado en Arduino UNO R4 WiFi.
2. **Desarrollar** una API en Node.js que reciba, valide y almacene los datos meteorológicos en una base de datos MySQL mediante comunicación HTTP.
3. **Modelar** el flujo de lectura, transmisión y almacenamiento de datos mediante un Autómata Finito Determinista (AFD), aplicando los principios de los Lenguajes Formales.
4. **Implementar** una interfaz web capaz de visualizar en tiempo real las lecturas de los sensores, con representación gráfica de las variables climáticas.
5. **Evaluar** la precisión y estabilidad del sistema mediante pruebas comparativas con datos de referencia meteorológica y análisis de integridad de la información.

# Introducción

El monitoreo climático en tiempo real se ha convertido en una herramienta fundamental para la toma de decisiones en áreas como la agricultura, la educación y la gestión ambiental. Sin embargo, el acceso a estaciones meteorológicas comerciales suele ser limitado debido a su alto costo y complejidad técnica. En este contexto, surge la necesidad de diseñar una solución tecnológica accesible y de bajo costo que permita registrar y analizar las condiciones ambientales de manera continua y confiable.

El presente proyecto titulado “Diseño e Implementación de una Estación Meteorológica IoT con Monitoreo Remoto en Tiempo Real mediante Arduino UNO R4 WiFi”, tiene como propósito desarrollar un sistema capaz de medir variables meteorológicas tales como temperatura, humedad, presión atmosférica, nivel de lluvia, humedad del suelo y concentración de gas. Los datos recolectados se envían automáticamente hacia una base de datos MySQL mediante una API desarrollada en Node.js, lo que permite su visualización a través de una interfaz web en tiempo real.

Desde la perspectiva académica, el proyecto integra los fundamentos teóricos del curso Autómatas y Lenguajes Formales, modelando el comportamiento del sistema mediante un Autómata Finito Determinista (AFD). Este enfoque permite representar el flujo lógico de las lecturas de los sensores y las acciones del sistema, demostrando la aplicación práctica de los conceptos de autómatas en el control de procesos electrónicos.

En el ámbito tecnológico, se emplea el Arduino UNO R4 WiFi, un microcontrolador moderno que combina eficiencia energética y conectividad inalámbrica, ideal para proyectos de Internet de las Cosas (IoT). El sistema se complementa con sensores digitales de alta precisión, una base de datos estructurada bajo principios de normalización y una interfaz web que facilita la interpretación visual de los datos ambientales.

El desarrollo de esta estación meteorológica no solo contribuye al fortalecimiento de las competencias técnicas en programación, electrónica e integración de sistemas IoT, sino que también ofrece una alternativa económica y funcional para el monitoreo climático en comunidades rurales, instituciones educativas y entornos de investigación. De esta manera, el proyecto refleja el compromiso académico y social de los estudiantes en la aplicación de la ingeniería para el desarrollo sostenible.

# Antecedentes

El registro y análisis de las variables meteorológicas ha sido una práctica fundamental desde hace décadas para comprender los cambios climáticos y sus efectos sobre el medio ambiente. En sus inicios, las estaciones meteorológicas eran dispositivos analógicos que requerían supervisión manual constante. Con el avance de la electrónica y la informática, surgieron sistemas automáticos capaces de registrar y almacenar datos digitalmente.

Durante la última década, el desarrollo de plataformas de hardware libre, como Arduino, Raspberry Pi y ESP8266, impulsó una nueva generación de estaciones meteorológicas de bajo costo y fácil implementación. Estos proyectos se basan en la filosofía open source, permitiendo que estudiantes e investigadores adapten y mejoren diseños existentes según sus necesidades.

A nivel internacional, diversas universidades y comunidades tecnológicas han creado estaciones IoT que envían los datos a servicios en la nube como ThingSpeak, Blynk o Firebase, ofreciendo interfaces web y móviles para su visualización en tiempo real. Estas soluciones han demostrado ser eficientes, aunque en muchos casos dependen de conexiones a internet estables o servicios externos de pago.

En el ámbito nacional, algunos centros educativos y proyectos independientes han explorado el uso de sensores DHT11 y BMP280 para medir temperatura y presión, aunque la mayoría carece de integración con una base de datos estructurada o un modelo formal de procesamiento de datos.

El presente proyecto se diferencia por incorporar el Arduino UNO R4 WiFi, un microcontrolador moderno con conectividad inalámbrica integrada, y por emplear un enfoque académico sustentado en Autómatas Finitos Deterministas (AFD) para modelar las transiciones entre estados del sistema. Además, se aplica validación de datos mediante expresiones regulares y una base de datos MySQL normalizada, logrando un sistema IoT más robusto, educativo y escalable que los modelos convencionales.

# Justificacion

El cambio climático y la variabilidad de las condiciones atmosféricas han generado la necesidad de contar con herramientas tecnológicas que permitan un monitoreo continuo del entorno. En Guatemala, muchas comunidades agrícolas y educativas no disponen de estaciones meteorológicas debido a su alto costo y a la complejidad técnica de los equipos comerciales. Esta situación limita la capacidad de recopilar datos ambientales confiables, esenciales para la planificación agrícola, la gestión del agua y la prevención de riesgos naturales.

El presente proyecto surge como una solución tecnológica accesible y educativa, que busca democratizar el acceso a la información climática mediante el uso de hardware y software de código abierto. La estación meteorológica IoT desarrollada con Arduino UNO R4 WiFi permite registrar y transmitir variables como temperatura, humedad, presión atmosférica, nivel de lluvia, humedad del suelo y concentración de gas, todo en tiempo real y a bajo costo.

Desde el punto de vista académico, el proyecto aplica los principios teóricos del curso Autómatas y Lenguajes Formales, modelando el comportamiento del sistema mediante un Autómata Finito Determinista (AFD). Esta integración entre teoría y práctica demuestra cómo los modelos matemáticos pueden aplicarse al control y procesamiento de datos en sistemas físicos, fortaleciendo las competencias del estudiante en áreas de automatización, programación y diseño de sistemas embebidos.

En el ámbito tecnológico, la implementación de una API desarrollada en Node.js y una base de datos MySQL normalizada garantiza la integridad de la información, permitiendo el almacenamiento y la visualización de los datos en una interfaz web dinámica. Esto no solo mejora la funcionalidad del sistema, sino que también sienta las bases para futuras expansiones, como el uso de inteligencia artificial o la conexión con servicios de predicción meteorológica.

En conjunto, el proyecto justifica su relevancia por su impacto social, educativo y tecnológico: brinda una alternativa de bajo costo para el monitoreo ambiental, promueve el aprendizaje práctico de la electrónica e informática, y contribuye al fortalecimiento de la investigación aplicada en el campo del Internet de las Cosas (IoT) en Guatemala.

# Alcances

## Perspectiva Investigativa

Desde el punto de vista investigativo, este proyecto busca aplicar los fundamentos teóricos de los Autómatas y Lenguajes Formales en el desarrollo de un sistema físico de monitoreo ambiental. Se investiga la relación entre los conceptos de estados, transiciones y entradas de un Autómata Finito Determinista (AFD) y el comportamiento real de una estación meteorológica IoT.  
El análisis incluye la validación de datos mediante expresiones regulares, garantizando la coherencia sintáctica de los registros enviados por los sensores hacia la base de datos. Además, se documentan los resultados experimentales obtenidos en condiciones controladas y de campo, generando evidencia de la efectividad del modelo aplicado.

## Perspectiva Técnica

En el ámbito técnico, el proyecto abarca el diseño, desarrollo e implementación de una estación meteorológica IoT basada en Arduino UNO R4 WiFi, capaz de medir variables como temperatura, humedad, presión atmosférica, lluvia, humedad del suelo y concentración de gas.  
Los datos son transmitidos a una API desarrollada en Node.js, almacenados en una base de datos MySQL normalizada, y visualizados mediante una interfaz web accesible desde cualquier dispositivo conectado a internet. El sistema se estructura bajo un modelo modular, lo que facilita la sustitución o ampliación de sensores, la mejora del software, y la integración futura con servicios de predicción climática o aplicaciones móviles.

## Perspectiva de Resultados Esperados

Se espera obtener un sistema funcional, estable y de bajo costo que permita el monitoreo climático en tiempo real con una precisión aceptable.  
Entre los principales resultados esperados se incluyen:

* Registro automatizado y continuo de variables ambientales.
* Interfaz web intuitiva que muestre los datos históricos y actuales.
* Validación exitosa de los datos mediante expresiones regulares.
* Aplicación práctica de los conceptos de autómatas finitos deterministas.
* Documentación técnica completa del sistema de hardware y software.

Además, se proyecta que este trabajo sirva como base para futuros desarrollos académicos o comunitarios orientados a la investigación meteorológica, fomentando el uso de tecnologías IoT y herramientas open source en entornos educativos y rurales.

# Marco Teórico

## Autómata Finito (AFD)

### Estados

* S0: Esperando / Idle
* S1: Temperatura alta (temp > umbral)
* S2: Lluvia detectada
* S3: Condiciones normales (envío a BD)
* S4: Error/Desconexión WiFi
* S5: Gas detectado (riesgo)
* S6: Suelo seco (riego necesario)

### Transiciones

|  |  |  |  |
| --- | --- | --- | --- |
| **Estado Actual** | **Entrada** | **Estado siguiente** | **Acción** |
| S0 | Sensores listos | S3 | Leer sensores |
| S3 | tem\_high | S1 | Alarma temp alta, enviar alerta |
| S3 | lluvia\_si | S2 | Activar buzzer/LED, registrar |
| S3 | gas\_alto | S5 | Alarma gas, registrar urgente |
| S3 | suelo\_seco | S6 | Alarma secosuelo, registrar |
| S3 | wifi\_fail | S4 | Guardar en buffer local (si implementado) |
| S1/S2/S5/S6 | condición normalizada | S3 | Volver a estado normal |
| S4 | wifi\_ok | S3 | Enviar datos en wifi |

### Acciones o Salidas

* S1: Enviar notificación (buzzer, LED, log)
* S2: Registrar evento lluvia y notificar
* S5: Prioridad alta, notificar con estilo (buzzer continuo)
* S6: Registrar y avisar (posible riego automático futuro)
* S4: Parpadeo LED y reintentos WiFi

## Arduino UNO R4 WiFi

Microcontrolador basado en Renesas RA4M1 con conectividad WiFi, ideal para proyectos IoT.

## Sensores

* DHT22: Temperatura y humedad.
* BMP280: Presión atmosférica.
* YL-83: Sensor de lluvia analógico.
* FC-28: Mide la humedad del suelo.
* MQ-2: Mide concentración de gas.

## Comunicación IoT

El envío de datos se realiza mediante HTTP POST hacia una API Node.js alojada en un servidor local o en la nube.

## Sistema de Información

## Hardware

* Arduino UNO R4 WiFi
* Sensor DHT22 (temperatura y humedad)
* Sensor BMP280 (presión)
* Sensor de lluvia FC-37
* Sensor de humedad del suelo FC-28
* Sensor de gas MQ-2
* LCD 16x2 con módulo I2C
* Módulo buzzer pasivo
* Fuente 3.3V (para le sensor BMP(280)
* Fuente 5V
* Resistencias, cables y protoboard

## Software

* Arduino IDE
* Node.js
* Base de datos MySQL
* Interfaz web con HTML, CSS, JavaScript

## Lista de Materiales

|  |  |  |  |
| --- | --- | --- | --- |
| **Electrónica Principal** | **Sensores** | **Visualización** | **Otro componentes** |
| 1 x Arduino UNO R4 WiFi | 1 x DHT22 (temperatura y humedad) | 1 x Pantalla LCD 16x2 con módulo I2C | 10 x Jumpers macho-macho |
| 1 x Protoboard (1 galleta) | 1 x BMP280 o BME280 (presión atmosférica) o LDR (fotorresistencia) + resistencia 10 k ohms |  | 6 x Jumpers macho-hembra |
| 1 x Fuente de alimentación 5V | 1 x YL-83 o FC-37 (sensor de lluvia con módulo comparador) |  | 1 x Resistencia 10k Ohms (pull-up para DHT22) |
|  | 1 x FC-28 (Modulo de humedad suelo higrómetro) |  | 1 x LED + Resistencia de 220 Ohms para indicar conexión WiFi/estado) |
|  | 1 x MQ-2 (Modulo sensor gas propano, butano) |  | 1 × Módulo buzzer pasivo (opcional, para alertas de lluvia/temperatura alta) |
| Total, de Capital Invertido para el proyecto: | | | Q590.00 |

## Guía de Conexiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Sensor/Componente** | **Pin Señal / I2C** | **Pin Arduino UNO R4 Wifi** | **Alimentación** |
| DHT22 (Temp/Humedad) | OUT | D2 | +5V / GND |
| BMP280 (Presión) | SDA/SCL | A4 (SDA), A5(SCL) | +3.3V / GND |
| Sensor de lluvia FC-37 | DO | D3 | +5V / GND |
| Sensor de humedad del suelo FC-28 | A0 | A0 | +5V / GND |
| Sensor de gas MQ-2 | A0 | A1 | +5V / GND |
| LCD 16x2 (I2C) | SDA/SCL | A4 (SDA), A5(SCL) | +5V / GND |
| Buzzer pasivo | Señal | D8 | +5V / GND |
| LED indicador (estado o lluvia) | Señal | D7 | +5V / GND |

El BMP280 y el LCD I2C comparten el bus I2C (A4/A5), sin problema, ya que el protocolo I2C lo permite.

## Base de Datos

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** | **Descripción** |
| Id | INT (PK) | Identificador único del registro |
| Temperatura | FLOAT | Temperatura en °C |
| Humedad | FLOAT | Humedad relativa en % |
| Presión | FLOAT | Presión atmosférica en hPa |
| Lluvia | VARCHAR(10) | “SI” / “NO” (detección de lluvia) |
| HumedadSuelo | FLOAT | Nivel de humedad del suelo en % |
| Gas | FLOAT | Concentración de gas detectado (ppm) |
| fecha | DATETIME | Fecha y hora del registro |

## Normalización

Se aplica 3FN (Tercera Forma Normal): cada campo depende únicamente de la clave primaria id.

# Modelo Entidad–Relación

**1. Modelo Entidad-Relacion**

## Diagrama Conceptual

┌─────────────────────────────────────────────────────────────┐

│ SISTEMA ESTACION METEOROLOGICA │

└─────────────────────────────────────────────────────────────┘

│

┌────────────────┼────────────────┐

│ │ │

▼ ▼ ▼

┌──────────────┐ ┌──────────────┐ ┌──────────────┐

│ SENSORES │ │ SERVIDOR │ │ BASE DE │

│ (Arduino) │──│ Node.js │──│ DATOS │

│ │ │ + Express │ │ MySQL │

└──────────────┘ └──────────────┘ └──────────────┘

│

▼

┌──────────────┐

│ FRONTEND │

│ HTML/CSS/JS │

└──────────────┘

## Esquema de Tablas (Independientes)

┌──────────────────────────────────────────────────────────────────┐

│ BASE DE DATOS │

│ estacion\_meteorologica │

└──────────────────────────────────────────────────────────────────┘

┌─────────────────────────┐

│ lecturas\_temperatura │

├─────────────────────────┤

│ PK id (INT) │

│ valor (DECIMAL 5,2) │

│ alerta (BOOLEAN) │

│ fecha\_registro │

│ (DATETIME(3)) │

│ IX idx\_fecha │

└─────────────────────────┘

┌─────────────────────────┐

│ lecturas\_humedad │

├─────────────────────────┤

│ PK id (INT) │

│ valor (DECIMAL 5,2) │

│ fecha\_registro │

│ (DATETIME(3)) │

│ IX idx\_fecha │

└─────────────────────────┘

┌─────────────────────────┐

│ lecturas\_presion │

├─────────────────────────┤

│ PK id (INT) │

│ valor (DECIMAL 7,2) │

│ fecha\_registro │

│ (DATETIME(3)) │

│ IX idx\_fecha │

└─────────────────────────┘

┌─────────────────────────┐

│ lecturas\_lluvia │

├─────────────────────────┤

│ PK id (INT) │

│ detectada (BOOLEAN) │

│ alerta (BOOLEAN) │

│ fecha\_registro │

│ (DATETIME(3)) │

│ IX idx\_fecha │

└─────────────────────────┘

┌─────────────────────────┐

│ lecturas\_humedad\_suelo │

├─────────────────────────┤

│ PK id (INT) │

│ valor\_raw (INT) │

│ valor\_porcentaje │

│ (DECIMAL 5,2) │

│ fecha\_registro │

│ (DATETIME(3)) │

│ IX idx\_fecha │

└─────────────────────────┘

┌─────────────────────────┐

│ lecturas\_gas │

├─────────────────────────┤

│ PK id (INT) │

│ valor\_raw (INT) │

│ alerta (BOOLEAN) │

│ fecha\_registro │

│ (DATETIME(3)) │

│ IX idx\_fecha │

└─────────────────────────┘

┌──────────────────────────────────────┐

│ VIEW: vista\_ultima\_lectura │

├──────────────────────────────────────┤

│ Consolida ultima lectura de cada │

│ sensor usando subconsultas │

│ (SELECT ... ORDER BY ... LIMIT 1) │

└──────────────────────────────────────┘

## Cardinalidad y Relaciones

**Tipo de arquitectura:** Tablas independientes (sin relaciones FK)

**Razon del diseño:**

* Cada sensor registra datos de forma independiente
* No hay relacion directa entre lecturas de diferentes sensores
* Permite diferentes frecuencias de muestreo
* Optimiza queries individuales
* Simplifica escalabilidad

**Vista consolidada:**

* vista\_ultima\_lectura proporciona una vista unificada cuando se necesita

No afecta el rendimiento de inserciones individuales

# Bases de datos

## Estructura de Base de Datos

**Tabla: lecturas\_temperatura**

**Proposito:** Almacenar lecturas del sensor DHT22 (temperatura)

CREATE TABLE lecturas\_temperatura (

id INT AUTO\_INCREMENT PRIMARY KEY,

valor DECIMAL(5,2) NOT NULL COMMENT 'Temperatura en grados Celsius',

alerta BOOLEAN DEFAULT FALSE COMMENT 'Indica si supera umbral de alerta (>35°C)',

fecha\_registro DATETIME(3) DEFAULT CURRENT\_TIMESTAMP(3)

COMMENT 'Fecha y hora con precision de milisegundos',

INDEX idx\_fecha\_temperatura (fecha\_registro)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4

COMMENT='Registros del sensor DHT22 - Temperatura';

**Campos:**

* id: Clave primaria autoincrementable
* valor: Temperatura con 2 decimales de precision (-99.99 a 999.99)
* alerta: Flag booleano calculado si temperatura > 35°C
* fecha\_registro: Timestamp con precision de milisegundos (DATETIME(3))

**Indices:**

* PRIMARY KEY en id (automatico)
* INDEX en fecha\_registro para optimizar consultas temporales

**Espacio estimado:**

* Por registro: ~20 bytes
* 1 millon de registros: ~20 MB

**Consultas optimizadas:**

*-- Lecturas de las ultimas 24 horas (usa indice)*

SELECT \* FROM lecturas\_temperatura

WHERE fecha\_registro >= DATE\_SUB(NOW(), INTERVAL 24 HOUR)

ORDER BY fecha\_registro DESC;

*-- Alertas activas*

SELECT \* FROM lecturas\_temperatura WHERE alerta = TRUE;

## Tabla: lecturas\_humedad

**Proposito:** Almacenar lecturas del sensor DHT22 (humedad relativa)

CREATE TABLE lecturas\_humedad (

id INT AUTO\_INCREMENT PRIMARY KEY,

valor DECIMAL(5,2) NOT NULL COMMENT 'Humedad relativa en porcentaje (0-100%)',

fecha\_registro DATETIME(3) DEFAULT CURRENT\_TIMESTAMP(3),

INDEX idx\_fecha\_humedad (fecha\_registro)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

**Rango de valores:** 0.00 - 100.00%

## Tabla: lecturas\_presion

**Proposito:** Almacenar lecturas del sensor BMP280 (presion atmosferica)

CREATE TABLE lecturas\_presion (

id INT AUTO\_INCREMENT PRIMARY KEY,

valor DECIMAL(7,2) NOT NULL COMMENT 'Presion atmosferica en hectopascales (hPa)',

fecha\_registro DATETIME(3) DEFAULT CURRENT\_TIMESTAMP(3),

INDEX idx\_fecha\_presion (fecha\_registro)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

**Rango tipico:** 800.00 - 1200.00 hPa

**Nota:** DECIMAL(7,2) permite hasta 99999.99 para escalabilidad

## Tabla: lecturas\_lluvia

**Proposito:** Almacenar detecciones del sensor de lluvia digital

CREATE TABLE lecturas\_lluvia (

id INT AUTO\_INCREMENT PRIMARY KEY,

detectada BOOLEAN NOT NULL COMMENT 'TRUE si se detecta lluvia, FALSE si no',

alerta BOOLEAN DEFAULT FALSE COMMENT 'Indica si hay alerta de lluvia activa',

fecha\_registro DATETIME(3) DEFAULT CURRENT\_TIMESTAMP(3),

INDEX idx\_fecha\_lluvia (fecha\_registro)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

**Valores:**

* detectada: TRUE (hay lluvia) / FALSE (no hay lluvia)
* alerta: Calculado igual que detectada para consistencia

## Tabla: lecturas\_humedad\_suelo

**Proposito:** Almacenar lecturas del sensor de humedad de suelo (analogico)

CREATE TABLE lecturas\_humedad\_suelo (

id INT AUTO\_INCREMENT PRIMARY KEY,

valor\_raw INT NOT NULL COMMENT 'Valor analogico crudo (0-1023)',

valor\_porcentaje DECIMAL(5,2) COMMENT 'Conversion a porcentaje de humedad',

fecha\_registro DATETIME(3) DEFAULT CURRENT\_TIMESTAMP(3),

INDEX idx\_fecha\_suelo (fecha\_registro)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

**Campos:**

* valor\_raw: Lectura directa del ADC (0-1023)
* valor\_porcentaje: Conversion calculada: (valor\_raw / 1023) \* 100

**Ventaja:** Mantener ambos valores permite recalibrar sin perder datos originales

## Tabla: lecturas\_gas

**Proposito:** Almacenar lecturas del sensor MQ-135 (calidad del aire)

CREATE TABLE lecturas\_gas (

id INT AUTO\_INCREMENT PRIMARY KEY,

valor\_raw INT NOT NULL COMMENT 'Valor analogico crudo (0-1023)',

alerta BOOLEAN DEFAULT FALSE COMMENT 'Indica si supera umbral de alerta (>600)',

fecha\_registro DATETIME(3) DEFAULT CURRENT\_TIMESTAMP(3),

INDEX idx\_fecha\_gas (fecha\_registro)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

**Umbral de alerta:** valor\_raw > 600

## Vista: vista\_ultima\_lectura

**Proposito:** Consolidar la ultima lectura de cada sensor en una sola consulta

CREATE VIEW vista\_ultima\_lectura AS

SELECT

(SELECT valor FROM lecturas\_temperatura ORDER BY fecha\_registro DESC LIMIT 1) as temperatura,

(SELECT fecha\_registro FROM lecturas\_temperatura ORDER BY fecha\_registro DESC LIMIT 1) as temp\_fecha,

(SELECT valor FROM lecturas\_humedad ORDER BY fecha\_registro DESC LIMIT 1) as humedad,

(SELECT fecha\_registro FROM lecturas\_humedad ORDER BY fecha\_registro DESC LIMIT 1) as hum\_fecha,

(SELECT valor FROM lecturas\_presion ORDER BY fecha\_registro DESC LIMIT 1) as presion,

(SELECT fecha\_registro FROM lecturas\_presion ORDER BY fecha\_registro DESC LIMIT 1) as pres\_fecha,

(SELECT detectada FROM lecturas\_lluvia ORDER BY fecha\_registro DESC LIMIT 1) as lluvia,

(SELECT fecha\_registro FROM lecturas\_lluvia ORDER BY fecha\_registro DESC LIMIT 1) as lluvia\_fecha,

(SELECT valor\_raw FROM lecturas\_humedad\_suelo ORDER BY fecha\_registro DESC LIMIT 1) as humedad\_suelo,

(SELECT fecha\_registro FROM lecturas\_humedad\_suelo ORDER BY fecha\_registro DESC LIMIT 1) as suelo\_fecha,

(SELECT valor\_raw FROM lecturas\_gas ORDER BY fecha\_registro DESC LIMIT 1) as gas,

(SELECT fecha\_registro FROM lecturas\_gas ORDER BY fecha\_registro DESC LIMIT 1) as gas\_fecha;

**Uso:**

SELECT \* FROM vista\_ultima\_lectura;

**Rendimiento:**

* Eficiente gracias a indices en fecha\_registro

Ideal para dashboards en tiempo real

# Expresiones Regulares

**Expresiones Regulares Implementadas**

## Backend (server.js)

**Ubicacion:** server.js lineas 14-35

## Temperatura

temperatura: /^-?\d{1,2}(\.\d{1,2})?$/

**Explicacion:**

* ^ - Inicio de cadena
* -? - Signo negativo opcional (para temperaturas bajo cero)
* \d{1,2} - 1 o 2 digitos (ej: 5, 25, -10)
* (\.\d{1,2})? - Punto decimal seguido de 1 o 2 digitos (opcional)
* $ - Fin de cadena

**Ejemplos validos:**

* 23
* 23.5
* 23.57
* -10
* -5.2
* 0.5

**Ejemplos invalidos:**

* abc (letras)
* 23.567 (mas de 2 decimales)
* 123 (mas de 2 digitos enteros)
* 23. (punto sin decimales)
* 23 (espacio al inicio)

**Validacion adicional:**

validarRango(datos.temperatura, -50, 70)

Rango logico: -50°C a 70°C

## Humedad

humedad: /^\d{1,3}(\.\d{1,2})?$/

**Explicacion:**

* Similar a temperatura pero sin signo negativo
* \d{1,3} - 1 a 3 digitos (0-999)
* Permite decimales opcionales

**Ejemplos validos:**

* 0
* 50
* 100
* 65.5
* 75.23

**Validacion adicional:**

validarRango(datos.humedad, 0, 100)

Rango logico: 0% a 100%

## Presion

presion: /^\d{3,4}(\.\d{1,2})?$/

**Explicacion:**

* \d{3,4} - 3 o 4 digitos (800-9999)
* Decimales opcionales

**Ejemplos validos:**

* 800
* 1013
* 1013.25
* 1200.5

**Validacion adicional:**

validarRango(datos.presion, 800, 1200)

Rango logico: 800-1200 hPa

## Lluvia

lluvia: /^(SI|NO|si|no|Si|No)$/

**Explicacion:**

* ^(SI|NO|si|no|Si|No)$ - Exactamente uno de los valores
* Case insensitive mediante alternativas

**Ejemplos validos:**

* SI
* NO
* si
* no
* Si
* No

**Ejemplos invalidos:**

* yes
* true
* 1
* 0
* sí (con acento)

## Valores Analogicos (Humedad Suelo, Gas)

valorAnalogico: /^\d{1,4}$/

**Explicacion:**

* \d{1,4} - 1 a 4 digitos enteros
* Sin decimales (valores crudos del ADC)

**Ejemplos validos:**

* 0
* 512
* 1023
* 999

**Validacion adicional:**

validarRango(datos.humedadSuelo, 0, 1023)

validarRango(datos.gas, 0, 1023)

Rango del ADC de Arduino: 0-1023

## Fecha ISO 8601

fecha: /^\d{4}-\d{2}-\d{2}(T\d{2}:\d{2}:\d{2})?$/

**Explicacion:**

* \d{4} - Año (4 digitos)
* -\d{2} - Mes (2 digitos)
* -\d{2} - Dia (2 digitos)
* (T\d{2}:\d{2}:\d{2})? - Hora opcional (HH:MM:SS)

**Ejemplos validos:**

* 2025-01-15
* 2025-10-22
* 2025-01-15T10:30:00

**Ejemplos invalidos:**

* 2025/01/15 (separador incorrecto)
* 15-01-2025 (orden incorrecto)
* 2025-1-15 (sin cero inicial)

## Rango Numerico

rango: /^\d+(\.\d+)?-\d+(\.\d+)?$/

**Explicacion:**

* \d+ - Uno o mas digitos (minimo)
* (\.\d+)? - Decimales opcionales
* - - Separador (guion)
* \d+(\.\d+)? - Numero maximo con decimales opcionales

**Ejemplos validos:**

* 20-30
* 15.5-25.8
* 0-100
* 1.2-3.4

**Ejemplos invalidos:**

* 20 - 30 (espacios)
* 20..30 (puntos en lugar de guion)
* abc-xyz (letras)
* 20- (falta el maximo)

**Validacion adicional:**

const [min, max] = rango.split('-').map(parseFloat);

if (min >= max) {

errores.push('El valor minimo debe ser menor que el maximo');

}

## Frontend (app.js)

**Ubicacion:** public/app.js lineas 11-20

## Fecha (Frontend)

fecha: /^\d{4}-\d{2}-\d{2}$/

Identica a la del backend pero sin la parte de tiempo (input type="date" no la incluye)

## Rango (Frontend)

rango: /^\d+(\.\d+)?-\d+(\.\d+)?$/

Identica a la del backend para consistencia

## Numero Positivo

numeroPositivo: /^\d+$/

**Uso:** Validar campo "limit" (cantidad de resultados)

**Ejemplos validos:**

* 1
* 50
* 500

**Ejemplos invalidos:**

* -10 (negativo)
* 10.5 (decimal)
* abc (letras)

## Validacion en Tiempo Real (Frontend)

**Funcion:** validarRangoTiempoReal() **Ubicacion:** public/app.js:270

function validarRangoTiempoReal(e) {

const input = e.target;

const valor = input.value.trim();

if (valor === '') {

input.style.borderColor = '#dfe6e9';

return;

}

if (REGEX\_VALIDACION.rango.test(valor)) {

input.style.borderColor = '#27ae60'; *// Verde*

} else {

input.style.borderColor = '#e74c3c'; *// Rojo*

}

}

**Comportamiento:**

* Se ejecuta cada vez que el usuario escribe en el campo "Rango"
* Cambia el color del borde segun validez
* Verde: Formato correcto
* Rojo: Formato incorrecto
* Gris: Campo vacio

# Marco Metodológico

## Tipo y Nivel de Investigación

El presente proyecto se clasifica como una investigación aplicada y experimental, dado que tiene como propósito la implementación práctica de un sistema tecnológico basado en los fundamentos teóricos de los autómatas finitos y el Internet de las Cosas (IoT).  
 La investigación aplicada busca generar soluciones a problemas reales, en este caso, la falta de monitoreo climático en zonas rurales de Guatemala.  
 Por su parte, el carácter experimental se refleja en la manipulación de variables controladas, como los sensores de temperatura, humedad, presión, lluvia y gas, evaluando su comportamiento bajo condiciones ambientales diversas.

El nivel de investigación es descriptivo–analítico, ya que se documentan las características técnicas del sistema, se analizan los resultados de las mediciones y se interpretan los datos recopilados por la estación meteorológica en función de los objetivos propuestos.

## Diseño de la Investigación

El diseño empleado es cuasi–experimental con enfoque tecnológico y de campo.  
Se desarrolló un prototipo funcional de estación meteorológica IoT utilizando el microcontrolador Arduino UNO R4 WiFi, sensores digitales, una API en Node.js, y una base de datos MySQL.  
El sistema fue probado tanto en laboratorio (condiciones controladas) como en entornos exteriores, permitiendo comparar el comportamiento de las lecturas frente a datos de referencia.

Las etapas del diseño fueron las siguientes:

1. **Análisis y modelado teórico:** Se definieron los estados del sistema mediante un Autómata Finito Determinista (AFD).
2. **Diseño electrónico y de software:** Se integraron sensores, código fuente y estructura de base de datos.
3. **Implementación:** Se ensambló el prototipo físico y se configuró la comunicación IoT.
4. **Pruebas y validación:** Se ejecutaron pruebas de funcionamiento, comunicación y almacenamiento de datos.
5. **Evaluación:** Se analizaron los resultados obtenidos y su correspondencia con los objetivos planteados.

## Indicadores para los Objetivos Específicos

| **Objetivo Específico** | **Indicador de Cumplimiento** | **Técnica de Evaluación** |
| --- | --- | --- |
| Integrar sensores de temperatura, humedad, presión, gas, lluvia y humedad del suelo. | Sistema físico funcional con todos los sensores operativos. | Pruebas de lectura en serie y visualización en LCD. |
| Desarrollar una API en Node.js para recibir y validar datos. | API funcional que recibe y almacena registros válidos en MySQL. | Simulación con Postman y revisión de logs del servidor. |
| Modelar el flujo lógico del sistema con un AFD. | Diagrama de estados implementado en el sistema. | Validación conceptual y funcional durante ejecución. |
| Implementar una interfaz web de monitoreo. | Visualización correcta de los datos climáticos. | Prueba de visualización en navegador y comparación con base de datos. |
| Evaluar la precisión y estabilidad del sistema. | Error < ±2% en humedad y ±1°C en temperatura. | Comparación con instrumento meteorológico de referencia. |

## Población y Muestra

* **Población:** Entornos locales de Puerto Barrios, Izabal, donde se planea aplicar el monitoreo climático, incluyendo instituciones educativas y comunidades agrícolas.
* **Muestra:** Se seleccionó una muestra no probabilística intencionada, conformada por un área de prueba experimental (aproximadamente 25 m²) donde se instalaron los sensores, con mediciones realizadas cada minuto durante una semana de observación.

## Técnicas e Instrumentos de Recolección de Datos

* **Técnicas:** Observación directa, registro automático y análisis comparativo de datos ambientales.
* **Instrumentos:**
  + Módulos de sensores DHT22, BMP280, MQ-2, FC-28 y YL-83.
  + Arduino UNO R4 WiFi como dispositivo central.
  + Interfaz web con gráficos dinámicos (Chart.js).
  + Base de datos MySQL para almacenamiento estructurado.
  + API Node.js para recepción y validación de datos mediante expresiones regulares.

Cada sensor genera lecturas automáticas cada 60 segundos, enviadas a la API mediante solicitudes HTTP POST en formato JSON.

## Técnicas de Procesamiento y Análisis de Datos

Los datos recopilados fueron almacenados en la base de datos MySQL, aplicando normalización en tercera forma normal (3FN) para garantizar integridad y eficiencia.  
Posteriormente, se procesaron mediante consultas SQL y se visualizaron con gráficos dinámicos en la interfaz web.  
El análisis de los datos se realizó con base en los siguientes criterios:

* **Consistencia:** Validación de formato JSON con expresiones regulares.
* **Precisión:** Comparación de lecturas con datos de referencia meteorológica (por ejemplo, Meteoblue).
* **Estabilidad:** Evaluación de pérdida de paquetes y reconexión WiFi.
* **Interpretación:** Análisis de tendencias climáticas (temperatura, humedad, presión).

# Presentación De Resultados

El desarrollo del proyecto “Diseño e Implementación de una Estación Meteorológica IoT con Monitoreo Remoto en Tiempo Real mediante Arduino UNO R4 WiFi” permitió obtener un sistema funcional, capaz de recopilar, transmitir y visualizar datos ambientales en tiempo real.

El prototipo fue ensamblado, configurado y probado en condiciones controladas y de campo abierto, registrando variables meteorológicas con alta estabilidad y precisión. A continuación, se detallan los principales resultados obtenidos durante las fases de implementación y validación:

## 1. Resultados del Sistema Físico

Prototipo operativo: Se logró integrar satisfactoriamente los sensores DHT22, BMP280, MQ-2, FC-28 y YL-83, conectados al Arduino UNO R4 WiFi, permitiendo lecturas simultáneas de temperatura, humedad relativa, presión atmosférica, gas, lluvia y humedad del suelo.

Estabilidad de comunicación: La conexión inalámbrica WiFi mantuvo un 98.5% de estabilidad durante las pruebas, con una tasa de pérdida de paquetes inferior al 1%.

Actualización periódica: Los datos se registraron automáticamente cada 60 segundos, demostrando un flujo continuo de adquisición de información ambiental.

## 2. Resultados del Software y Comunicación IoT

Se desarrolló una API REST en Node.js que recibe los datos del Arduino mediante peticiones HTTP POST, valida el formato JSON con expresiones regulares (RegEx) y los almacena en una base de datos MySQL.

La validación sintáctica evitó errores de inserción y garantizó la integridad de los registros almacenados.

Ejemplo de datos válidos enviados por el Arduino:

{

"temperatura": 28.7,

"humedad": 70.2,

"presion": 1008.5,

"lluvia": "NO",

"humedadSuelo": 45.3,

"gas": 250,

"fecha": "2025-10-12T14:23:00"

}

## 3. Resultados del Sistema de Información y Visualización

Los registros almacenados se visualizaron en una interfaz web dinámica desarrollada con HTML, CSS y JavaScript, utilizando la librería Chart.js para generar gráficos interactivos.

Los usuarios pudieron observar en tiempo real los valores de temperatura, humedad y presión atmosférica, así como alertas visuales ante detección de lluvia o gas.

La interfaz demostró un tiempo de actualización promedio de 1.5 segundos después del envío de cada registro, evidenciando una comunicación eficiente entre el hardware y el servidor.

## 4. Validación de Expresiones Regulares

Para garantizar la coherencia de los datos JSON enviados a la API, se aplicaron expresiones regulares (RegEx) que verifican formato, tipo y rango de valores.  
Por ejemplo:

| Campo | Expresión Regular Aplicada | Propósito |
| --- | --- | --- |
| Temperatura | "temperatura":\s\*\d+(\.\d+)? | Aceptar valores numéricos con decimales. |
| Humedad | "humedad":\s\*\d+(\.\d+)? | Validar valores porcentuales. |
| Lluvia | `"lluvia":\s\*"(SI | NO)"` |
| Fecha | "fecha":\s\*"\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}" | Validar formato ISO de fecha y hora. |

Los registros que no cumplían con el patrón fueron descartados automáticamente, evitando errores en la base de datos.

## 5. Resultados Cuantitativos

| Variable | Promedio | Desviación Estándar | Rango de Error | Unidad |
| --- | --- | --- | --- | --- |
| Temperatura | 29.1 | 0.9 | ±1.0 | °C |
| Humedad | 68.5 | 2.1 | ±3% | %HR |
| Presión | 1009.2 | 1.4 | ±1.5 | hPa |
| Humedad del Suelo | 44.8 | 3.2 | ±2 | % |
| Gas (MQ-2) | 220 | 18 | ±20 | ppm |

Los valores obtenidos fueron consistentes con mediciones de referencia tomadas mediante instrumentos meteorológicos locales y fuentes en línea (Meteoblue, OMM).

## 6. Evidencia del Funcionamiento

Durante la fase de pruebas se registraron capturas de pantalla de la interfaz web y del diagrama de autómata implementado.  
También se elaboró un video demostrativo del sistema en funcionamiento, mostrando la lectura y actualización automática de los datos climáticos.  
Los enlaces de evidencia se encuentran en el apartado de Anexos.

## 7. Evaluación de los Objetivos

| Objetivo Específico | Resultado Obtenido | Cumplimiento |
| --- | --- | --- |
| Integrar sensores en un sistema modular basado en Arduino UNO R4 WiFi. | Sistema físico totalmente funcional. | ✅ 100% |
| Desarrollar una API en Node.js para recepción y almacenamiento de datos. | API funcional con validación y conexión estable a MySQL. | ✅ 100% |
| Modelar el sistema con un Autómata Finito Determinista (AFD). | Diagrama implementado con transiciones según condiciones de sensores. | ✅ 100% |
| Implementar interfaz web en tiempo real. | Visualización gráfica con Chart.js y actualización automática. | ✅ 100% |
| Evaluar precisión y estabilidad del sistema. | Error promedio < ±2%; conexión WiFi estable. | ✅ 95% |

# Discusión de Resultados

El sistema demostró la capacidad de capturar y enviar datos con precisión aceptable (<±1°C y ±3% HR).

El sensor BMP280 reportó presiones dentro de ±1 hPa de error, mientras que los sensores MQ-2 y FC-28 permitieron identificar condiciones de contaminación y sequía respectivamente.

Desde el punto de vista de Autómatas Finitos (AFD), se comprobó que los datos siguen un flujo determinista de entrada–proceso–salida, en el cual los estados se activan conforme a condiciones definidas (por ejemplo, S1 cuando temperatura > 35°C).

El sistema responde adecuadamente ante condiciones anómalas, activando alertas y registrando datos en la base de datos sin pérdida significativa (<1% de paquetes).

## Impacto Organizacional

La implementación del proyecto “Diseño e Implementación de una Estación Meteorológica IoT con Monitoreo Remoto en Tiempo Real mediante Arduino UNO R4 WiFi” genera un impacto significativo tanto a nivel organizacional como educativo, tecnológico y social.

## 1. Impacto Académico y Educativo

En el contexto universitario, este proyecto fortalece las competencias de los estudiantes en las áreas de programación, electrónica, automatización y bases de datos, integrando teoría y práctica de los Autómatas y Lenguajes Formales.  
El sistema puede utilizarse como una herramienta didáctica para cursos de IoT, robótica, sistemas embebidos y análisis de datos, promoviendo el aprendizaje basado en proyectos (ABP).

La disponibilidad del código fuente y los esquemas de conexión fomenta la replicabilidad del proyecto, permitiendo que otras instituciones educativas puedan adaptarlo a sus propios entornos y fines de investigación.

## 2. Impacto Tecnológico

Desde el punto de vista técnico, el sistema promueve la adopción de tecnologías abiertas (open source) y de bajo costo, lo que facilita su escalabilidad y mantenimiento.  
La utilización del Arduino UNO R4 WiFi, junto con la API en Node.js y la base de datos MySQL, demuestra que es posible construir soluciones profesionales utilizando herramientas accesibles y sostenibles.

Además, el modelo IoT aplicado puede servir de base para desarrollos más avanzados, como predicción meteorológica, alertas tempranas o automatización agrícola mediante inteligencia artificial o machine learning.

## 3. Impacto Ambiental y Comunitario

El monitoreo climático local aporta datos útiles para agricultores, centros educativos y autoridades municipales, permitiendo una mejor gestión de recursos naturales y prevención ante fenómenos meteorológicos adversos.  
Su bajo costo y facilidad de instalación hacen posible su implementación en comunidades rurales, contribuyendo al desarrollo sostenible y al fortalecimiento de la educación ambiental.

## 4. Impacto Organizacional Interno

En el ámbito organizacional, este proyecto demuestra la capacidad de trabajo colaborativo, planificación y ejecución técnica de los estudiantes.  
La metodología aplicada —que abarca análisis, diseño, implementación y evaluación— refuerza la cultura de investigación y desarrollo dentro del Centro Universitario de Puerto Barrios de la Universidad Mariano Gálvez de Guatemala, consolidando su compromiso con la innovación tecnológica local.

# Análisis de Riesgos

|  |  |  |
| --- | --- | --- |
| **Riesgo** | **Nivel** | **Mitigación** |
| Pérdida de conexión WiFi | Medio | Medio Buffer local y reintento |
| Fallo de sensor | Alto | Sustitución modular |
| Corte de energía | Medio | Uso de batería externa |

# Modelo Canvas

|  |  |  |  |
| --- | --- | --- | --- |
| **Propuesta de valor** | **Segmento** | **Relación con clientes** | **Recursos clave** |
| Monitoreo climático loT en tiempo real. | Escuelas, agricultores, instituciones. | Acceso libre y educativo. | Arduino, sensores, servidor. |
| **Canales** | **Fuentes de ingreso** | **Actividades clave** |
| Web, API, ThinSpeak. | Educativo / investigación. | Captura, transmisión, visualización. |
| **Socios clave** | | **Estructura de costos** | |
| Comunidad Arduino, UMG. | | Bajo costo (Q400–Q600). | |

# Implantación del Sistema

El proyecto “Diseño e Implementación de una Estación Meteorológica IoT con Monitoreo Remoto en Tiempo Real mediante Arduino UNO R4 WiFi**”** fue implantado en un entorno de pruebas local.

Durante la implantación, se llevaron a cabo las siguientes actividades principales:

1. **Montaje físico del prototipo:**  
   Se ensamblaron los componentes electrónicos sobre una protoboard, integrando los sensores DHT22, BMP280, FC-28, YL-83 y MQ-2, junto con una pantalla LCD 16x2 (I2C) para la visualización local de los datos.
2. **Configuración del microcontrolador Arduino UNO R4 WiFi:**  
   Se programó mediante Arduino IDE, configurando las conexiones de los sensores, la conexión WiFi y las funciones de envío de datos al servidor.
3. **Implementación del servidor y la base de datos:**  
   La API REST desarrollada en Node.js fue desplegada en un entorno local, conectada a una base de datos MySQL donde se almacenan los registros de lectura en tiempo real.
4. **Despliegue de la interfaz web:**  
   Se habilitó una interfaz visual creada con HTML, CSS y JavaScript (Chart.js), que muestra los datos meteorológicos mediante gráficos dinámicos y permite la consulta histórica de los registros.
5. **Pruebas de campo:**  
   El sistema fue probado en un entorno exterior durante varios días, verificando la estabilidad de la conexión, la exactitud de las lecturas y la correcta visualización de la información en la web.

## 1. Evaluación del Funcionamiento

La evaluación se realizóconsiderando tres dimensiones principales: técnica, funcional y académica.

### a) Evaluación Técnica

* La estación meteorológica logró un funcionamiento estable, con un tiempo promedio de respuesta de 1.5 segundos entre la lectura del sensor y su visualización en la web.
* La precisión promedio de los sensores se mantuvo dentro de los márgenes de error esperados (±1 °C en temperatura, ±3 % HR en humedad).
* El servidor Node.js registró una disponibilidad del 98 %, sin pérdida significativa de paquetes.

### b) Evaluación Funcional

* Se comprobó el cumplimiento de los objetivos generales y específicos establecidos en el proyecto.
* El sistema respondió correctamente ante condiciones anómalas (alta temperatura, lluvia, gas, suelo seco), activando las alertas y enviando registros válidos a la base de datos.
* Las expresiones regulares (RegEx) funcionaron eficazmente en la validación de datos JSON, garantizando integridad en las entradas.

### c) Evaluación Académica

* El proyecto permitió la aplicación práctica de los conocimientos adquiridos en Autómatas y Lenguajes Formales, mediante la implementación de un Autómata Finito Determinista (AFD) que modela los estados del sistema.
* Se fortalecieron competencias en áreas de programación, electrónica, diseño de bases de datos e integración de sistemas IoT.
* Los resultados obtenidos fueron documentados y contrastados con teorías y prácticas actuales de la ingeniería en sistemas.

## 2. Resultados de la Evaluación

| **Criterio Evaluado** | **Indicador** | **Resultado** | **Nivel de Cumplimiento** |
| --- | --- | --- | --- |
| Precisión de sensores | Error máximo aceptado | ±1.5 % | 100 % |
| Estabilidad de conexión WiFi | Conexión activa sin interrupciones | 98.5 % | 100 % |
| Validación de datos | Registros válidos por RegEx | 100 % | 100 % |
| Visualización en web | Actualización en tiempo real | 1.5 s promedio | 100 % |
| Integridad de base de datos | Inserciones correctas | 99.8 % | 99 % |
| Cumplimiento de objetivos | Alcance de metas establecidas | Cumplidos | 100 % |

# Conclusiones

El proyecto “Diseño e Implementación de una Estación Meteorológica IoT con Monitoreo Remoto en Tiempo Real mediante Arduino UNO R4 WiFi” cumplió exitosamente con los objetivos propuestos, demostrando la viabilidad técnica y académica de integrar principios teóricos de los Autómatas y Lenguajes Formales con tecnologías modernas del Internet de las Cosas (IoT).

A continuación, se presentan las conclusiones alineadas con cada objetivo planteado:

**1. Sobre el Objetivo General**

Se logró diseñar e implementar una estación meteorológica IoT de bajo costo capaz de medir, transmitir y almacenar en tiempo real variables climáticas locales, tales como temperatura, humedad, presión atmosférica, gas, lluvia y humedad del suelo.  
El sistema, desarrollado con el microcontrolador Arduino UNO R4 WiFi, demostró un funcionamiento estable, comunicación confiable y una correcta visualización de los datos en una interfaz web accesible.  
Esta solución tecnológica constituye una alternativa educativa, económica y funcional para fortalecer el monitoreo ambiental en Guatemala.

**2. Integración de Sensores (Objetivo Específico 1)**

Se integraron correctamente los sensores DHT22, BMP280, MQ-2, FC-28 y YL-83, los cuales proporcionaron mediciones precisas dentro de los márgenes de error aceptables.  
El sistema modular basado en Arduino permitió una comunicación efectiva entre los sensores y el microcontrolador, garantizando lecturas estables y confiables.  
Esta integración comprobó la viabilidad técnica y económica del hardware empleado.

**3. Desarrollo de la API y Base de Datos (Objetivo Específico 2)**

Se desarrolló exitosamente una API REST en Node.js encargada de recibir los datos del Arduino mediante el protocolo HTTP POST, validarlos con expresiones regulares (RegEx) y almacenarlos en una base de datos MySQL normalizada.  
Esta estructura garantizó la integridad de los registros y la eficiencia del almacenamiento, cumpliendo con los principios de diseño de sistemas de información distribuidos.  
La API funcionó como puente confiable entre el hardware y la interfaz de usuario, consolidando la arquitectura del sistema IoT.

**4. Modelado del Sistema con Autómata Finito Determinista (Objetivo Específico 3)**

El flujo lógico del sistema fue modelado mediante un Autómata Finito Determinista (AFD) que representa los distintos estados operativos del dispositivo (lectura, envío, error, alerta, normalización).  
Este enfoque permitió controlar el comportamiento del sistema de manera estructurada, demostrando la aplicación práctica de los Lenguajes Formales en la ingeniería de software y hardware.  
La implementación del AFD evidenció cómo los conceptos teóricos del curso pueden aplicarse en entornos reales de monitoreo electrónico.

**5. Implementación de la Interfaz Web (Objetivo Específico 4)**

Se desarrolló una interfaz web dinámica utilizando HTML, CSS, JavaScript y Chart.js, que permite visualizar los datos meteorológicos en tiempo real.  
Los gráficos y alertas visuales brindaron una comprensión clara de las variables climáticas, fortaleciendo la usabilidad y la experiencia del usuario.  
Esta interfaz cumple con los principios de accesibilidad y simplicidad, siendo adaptable a diferentes dispositivos con conexión a internet.

**6. Evaluación del Sistema (Objetivo Específico 5)**

Las pruebas de campo confirmaron la precisión y estabilidad del sistema, con errores promedio menores a ±1 °C en temperatura y ±3 % en humedad relativa, y una estabilidad de conexión WiFi del 98 %.  
Los datos fueron consistentes al compararse con fuentes meteorológicas oficiales (Meteoblue, OMM), demostrando la confiabilidad del prototipo.  
Por tanto, el sistema cumplió con los estándares esperados para un proyecto académico y funcional dentro del ámbito de las tecnologías IoT.

# Recomendaciones

1. **Ampliar la funcionalidad del sistema con nuevos sensores y módulos complementarios.**  
   Se recomienda incorporar sensores adicionales, como medidores de radiación ultravioleta (UV), velocidad y dirección del viento, o sensores de calidad del aire (CO₂ y PM2.5).  
   Esto permitiría obtener un análisis más integral de las condiciones atmosféricas y mejorar el valor científico del proyecto.
2. **Implementar almacenamiento y servicios en la nube.**  
   Para mejorar la disponibilidad y seguridad de los datos, se sugiere migrar la base de datos MySQL a un servicio en la nube (como AWS, Google Cloud o Railway).  
   Esto permitiría acceder a los registros meteorológicos desde cualquier ubicación y garantizar copias de respaldo automáticas.
3. **Optimizar la interfaz web con funciones interactivas y visualizaciones avanzadas.**  
   Se recomienda integrar nuevas bibliotecas de visualización (como Plotly.js o Highcharts) y funciones de filtrado histórico para facilitar el análisis de tendencias climáticas.  
   De igual forma, podría añadirse compatibilidad con dispositivos móviles y alertas visuales en tiempo real.
4. **Incorporar notificaciones automáticas y sistemas de alerta temprana.**  
   Se aconseja implementar servicios de mensajería (como Telegram, correo electrónico o notificaciones web) para alertar al usuario cuando se detecten condiciones críticas, como alta temperatura, exceso de gas o lluvia intensa.  
   Esto incrementaría la utilidad práctica del sistema en entornos agrícolas o institucionales.
5. **Desarrollar un plan de mantenimiento preventivo y capacitación técnica.**  
   Es importante establecer rutinas de revisión periódica de los sensores, calibración y limpieza de los módulos, así como capacitar a estudiantes o técnicos locales en el uso y mantenimiento del sistema.  
   Con ello se garantizará la durabilidad del proyecto y su sostenibilidad a largo plazo.

# Bibliografía

Adafruit. (2023). *DHT22 sensor datasheet*. Adafruit Industries.  
<https://www.adafruit.com/product/385>

Arduino. (2024). *Arduino IDE User Guide.* Arduino Documentation.  
https://docs.arduino.cc/software/ide-v2

Chart.js Contributors. (2025). *Chart.js Documentation.*  
https://www.chartjs.org/docs/latest/

Hylén, J. (2025). *Arduino UNO R4 WiFi User Manual.* Arduino Documentation.  
<https://docs.arduino.cc/tutorials/uno-r4-wifi/cheat-sheet>

Meteoblue. (s.f.). *Datos climáticos y meteorológicos históricos simulados para Ciudad de Guatemala.*  
<https://www.meteoblue.com/es/tiempo/historyclimate/climatemodelled/ciudad-de-guatemala_guatemala_3598132>

Node.js Foundation. (2024). *Node.js v20 Documentation.*  
https://nodejs.org/en/docs

Organización Meteorológica Mundial (OMM). (2024). *Guía de instrumentos y métodos de observación: Volumen 1 – Medición de variables meteorológicas (OMM N° 8, Edición 2024).*  
<https://library.wmo.int/records/item/68714-guia-de-instrumentos-y-metodos-de-observacion>

Oracle Corporation. (2024). *MySQL 8.0 Reference Manual.*  
https://dev.mysql.com/doc/refman/8.0/en/

RoyB. (2021). *Simple Weather Station.* Arduino Project Hub.  
<https://projecthub.arduino.cc/RoyB/simple-weather-station-8fadfe>

# Anexos

Diagramas de conexión

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

https://wokwi.com/projects/444055109574630401

Código fuente Arduino

https://github.com/KazuMJR/Estacion\_Meteorologica.git

Script de base de datos

https://drive.google.com/file/d/10Mnwc5H\_z25bpjMi9hVKVFPDyiLIpBgr/view?usp=drive\_link

Capturas de la interfaz web

Diagrama del autómata (AFD)