Universidad De San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Arquitectura de Computadores y Ensambladores 2 B

Primer semestre 2025

Catedrático: Ing. Jurgen Andoni Ramirez Ramirez

Auxiliar: Axel Calderón

Proyecto Único - Fase 1

Sistema Inteligente de Monitoreo Ambiental para Cuartos de Servidores mediante IoT.

Balam José Tomas Aju 201807028

Byron Enrique Rumpich sal 201907769

Iris Carolina Paz Guzmán 202101728

Miguel Adrian Tubac Agustin 202101927

Fecha de Entrega: 21/02/2025

INTRODCUCCION

La presente implementación se centra en el desarrollo de un Sistema Inteligente de Monitoreo Ambiental para cuartos de servidores mediante tecnología IoT, utilizando un microcontrolador Arduino que adquiere y procesa datos de múltiples sensores para medir variables críticas como temperatura, humedad, calidad del aire, corriente eléctrica y movimiento.

El código de Arduino gestiona la lectura de datos, su almacenamiento en EEPROM, la visualización en una pantalla LCD y la activación de alertas mediante LEDs cuando se detectan condiciones anómalas, mientras que el programa en Processing recibe esta información a través de comunicación serial para representarla gráficamente en tiempo real.

Este enfoque integral permite un monitoreo preciso del entorno, facilitando la toma de decisiones oportunas que aseguran un ambiente óptimo y seguro para el funcionamiento de los servidores.

OBJETIVOS

Objetivos Generales:

1. Desarrollar un sistema de monitoreo ambiental IoT para cuartos de servidores mediante el uso de sensores y microcontroladores, permitiendo la medición y análisis en tiempo real de variables críticas como temperatura, humedad, calidad del aire, corriente eléctrica y proximidad.

2. Implementar una interfaz visual en Processing que permita visualizar y analizar los datos recolectados por los sensores, facilitando la identificación de anomalías y la toma de decisiones en tiempo real.

Objetivos Especificos

1. Integrar y programar sensores en Arduino para capturar datos ambientales clave, como temperatura, humedad, calidad del aire, corriente eléctrica y proximidad, asegurando su correcto funcionamiento y calibración.

2. Desarrollar un sistema de alertas y almacenamiento de datos, utilizando una pantalla LCD y LEDs de advertencia para notificar niveles críticos, y empleando la memoria EEPROM para el almacenamiento de registros históricos.

3. Implementar la comunicación entre Arduino y Processing mediante puerto serie, permitiendo la visualización y análisis de datos en tiempo real a través de una interfaz gráfica en el software Processing.

MARCO TEORICO

En el desarrollo del sistema de monitoreo ambiental basado en IoT para cuartos de servidores, se emplearon diversas tecnologías y componentes electrónicos fundamentales para la captura, procesamiento y visualización de datos. A continuación, se describen los principales elementos y funciones utilizadas en el código de Arduino.

Microcontrolador Arduino

El microcontrolador **Arduino** es una plataforma de hardware libre utilizada para la automatización y el control de dispositivos electrónicos. Su facilidad de programación y amplia compatibilidad con sensores lo convierten en una excelente opción para el monitoreo de variables ambientales. En este proyecto, Arduino gestiona la recepción y procesamiento de datos de sensores, la visualización de información en una pantalla LCD y el almacenamiento de datos en la memoria EEPROM.

Sensores Empleados

- **DHT11:** Sensor de temperatura y humedad que permite medir estas variables en el entorno del cuarto de servidores.
- **MQ135:** Sensor de calidad del aire utilizado para medir la concentración de CO2.
- **HC-SR04:** Sensor ultrasónico que detecta la proximidad de personas dentro del cuarto de servidores.
- **ACS712 5A:** Sensor de corriente que mide el consumo eléctrico de los dispositivos conectados.
- **Fotorresistencia:** Sensor de luz utilizado para determinar la iluminación en el ambiente.

Sistema de Almacenamiento y Visualización

Para el almacenamiento de los datos históricos, se utiliza la memoria **EEPROM** de Arduino, que permite guardar valores de temperatura, humedad, calidad del aire, corriente eléctrica y luz para su posterior análisis. La información en tiempo real se presenta en una **pantalla LCD I2C** y, en caso de detectar valores críticos, se activan **LEDs de advertencia** en tres colores:

- **Rojo:** Indica temperatura elevada.
- **Azul:** Indica humedad excesiva.
- **Amarillo:** Indica un consumo anormal de corriente.

Funciones Implementadas en el Código

1. **Lectura de Sensores:** Captura los valores de los sensores de temperatura, humedad, corriente, proximidad y calidad del aire.
2. **Almacenamiento en EEPROM:** Guarda datos históricos de las mediciones para su análisis posterior.
3. **Sistema de Alertas:** Enciende los LEDs de advertencia cuando los valores de temperatura, humedad o corriente superan los umbrales críticos.
4. **Visualización en LCD:** Muestra los datos en la pantalla LCD para monitoreo en tiempo real.
5. **Interacción con Botones:** Permite al usuario visualizar datos almacenados y guardarlos en EEPROM a través de botones físicos

Este conjunto de código empleado en Arduino y las tecnologías y funciones empleadas se permitirá la creación de un sistema eficiente para el monitoreo y la gestión de condiciones ambientales en cuartos de servidores, asegurando su correcto funcionamiento y previniendo fallos por condiciones adversas.

A continuación, se desglosará el funcionamiento que Tendrá el Código de Arduino y el código de Processing y que funciones tendrá cada Uno:

Funciones en el Código de Arduino

1. **Lectura de Sensores:**
    - Captura valores de **temperatura, humedad, calidad del aire, corriente eléctrica y proximidad**.
2. **Almacenamiento de Datos:**
    - Guarda las mediciones en la **memoria EEPROM** para su consulta posterior.
3. **Sistema de Alertas:**

- Enciende **LEDs de advertencia** cuando los valores superan umbrales críticos.

4. **Visualización en LCD:**
    - Muestra en la pantalla LCD los valores en tiempo real.
5. **Interacción con Botones:**
    - Permite visualizar datos guardados y realizar registros en EEPROM.
6. **Promedio de Corriente:**
    - Calcula el consumo eléctrico en un período de tiempo.

Funciones en el Código de Processing

1. **Recepción de Datos:**
    - Obtiene la información enviada por Arduino a través del **puerto serie**.
2. **Visualización en Tiempo Real:**
    - Muestra datos mediante **gráficos dinámicos** para facilitar el monitoreo.
3. **Sistema de Alertas Visuales:**
    - Cambia colores y notificaciones según condiciones críticas.
4. **Interacción con el Usuario:**
    - Permite manipular datos y ajustar parámetros en la interfaz.

Diseño de la Maqueta

La maqueta debe representar un **cuarto de servidores** en **miniatura**, donde se colocarán los sensores estratégicamente para simular el monitoreo del ambiente.

Elementos que incluye la maqueta:

1. **Estructura del cuarto de servidores:**
    - Caja o contenedor que represente el espacio cerrado.

- Rejillas de ventilación simuladas para mayor realismo.
- Miniaturas de racks o servidores para ambientar la maqueta.

2. **Ubicación de los sensores:**
    - **Sensor DHT11 (temperatura y humedad):** Se coloco en la parte central del cuarto para capturar el estado general del ambiente.
    - **Sensor MQ135 (calidad del aire):** Ubicado en un punto alto del cuarto, ya que los gases tienden a concentrarse en la parte superior.
    - **Sensor HC-SR04 (proximidad):** Cerca de la puerta o entrada para detectar la presencia de personas.
    - **Sensor ACS712 (corriente):** Conectado a una fuente de alimentación o maqueta de servidor para monitorear el consumo eléctrico.
    - **Fotorresistencia (luz):** En el techo o cerca de una "ventana" simulada para medir la iluminación del ambiente.

3. **Componentes electrónicos:**
    - **Arduino:** Conectado a los sensores y colocado en un costado de la maqueta.
    - **Pantalla LCD:** Ubicada en la parte exterior de la maqueta para mostrar los datos en tiempo real.
    - **LEDs de advertencia:**
        - **Rojo:** Cerca del sensor de temperatura para indicar sobrecalentamiento.
        - **Azul:** En la parte superior para indicar alta humedad.
        - **Amarillo:** Junto a la fuente de alimentación para advertir sobre anomalías en la corriente.
    - **Botones físicos:** Colocados en el exterior para interactuar con la pantalla LCD y EEPROM.
4. **Cableado y conexiones:**
    - Estos están de forma oculta o bien distribuido para que la maqueta sea ordenada y funcional.

- Se uso cinta adhesiva o conductos pequeños para organizar los cables.

5. **Base y materiales:**
    - Carton chip, acrílico o bien presentación para las estructuras.
    - Se procedio a pintar y rotular cada parte para identificar sensores y funciones.

Funcionamiento de la maqueta:

- Los sensores detectarán el estado del ambiente en la maqueta y enviarán los datos a **Arduino**.
- La **pantalla LCD** mostrará las mediciones en tiempo real.
- **Processing** procesará los datos y los representará gráficamente en la computadora.
- Los **LEDs de advertencia** se encenderán si hay anomalías en temperatura, humedad o corriente.
- Se podrán almacenar y recuperar datos históricos con los botones físicos.

ANEXOS

1. Recepción y Procesamiento de Datos Serial

```
float needleX = centerX + cos(radians(angle)) * (radio - 20);
float needleY = centerY + sin(radians(angle)) * (radio - 20);
stroke(255, 0, 0);
strokeWeight(3);
line(centerX, centerY, needleX, needleY);

fill(255);
noStroke();
ellipse(centerX, centerY, 10, 10);

String calidad = corrienteVal < 0.4 ? "NORMAL" : corrienteVal < 0.7 ? "PRECAUCIÓN" : "ALTA";
color textColor = corrienteVal < 0.4 ? color(0, 180, 0) : corrienteVal < 0.7 ? color(255, 165, 0) : color(255, 0, 0);

stroke(0);  // Color negro para la orilla
strokeWeight(6);  // Grosor del borde
noFill();
arc(centerX, centerY, radio * 2 + 20, radio * 2 + 20, radians(-135), radians(135));

// titulo-------------------------------------------------------------
myFont = createFont("Arial-Bold", 18); // Cargar la fuente Arial con tamaño 32
textFont(myFont); // Aplicar la fuente
fill(textColor);
text("CORRIENTE: " + calidad, x_aire + 30, y_aire + 640);// titulo
```

```
      if (distancia_mov >= 19){
        distancia_mov = 0;
      } else {
        numero_personas++;
      }
    }
  }
```

2. Visualización del Sensor de Corriente

```
void infromacion_SensorCorriente(){
  int centerX = x_corriente + 170;
  int centerY = y_corriente + 145;
  int radio = 120;
  float minCO2 = 0.0;
  float maxCO2 = 1.0;
  float angle = map(corrienteVal*5, minCO2, maxCO2, -135, 135);

  for (int i = -135; i <= 135; i++) {
    float lerpFactor = map(i, -135, 135, 0, 1);
    stroke(lerpColor(color(0, 255, 0), color(255, 0, 0), lerpFactor));
    float x1 = centerX + cos(radians(i)) * (radio - 10);
    float y1 = centerY + sin(radians(i)) * (radio - 10);
    float x2 = centerX + cos(radians(i)) * (radio + 10);
    float y2 = centerY + sin(radians(i)) * (radio + 10);
    line(x1, y1, x2, y2);
  }
```

3. Visualización del Termómetro

```
  termometro();
  texto_temperatura();
}

void termometro(){
  if(temperatura > 106){ // mantener un limimte a la temperatura
    temperatura = 106;
  }
  if (temperatura < 0){
    temperatura = 0;
  }
  float temp_termo = (temperatura) * 2.5 + 30;
  // primero dibunar lineas
  lineas_Temperatura();

   // Dibujar el termómetro
  stroke(0);
  strokeWeight(3);
  fill(200);
  rect(x_temp + 75, y_temp +60, 40, 300,90); // Tubo del termómetro
```

```
  // Dibujar el nivel de temperatura,
  // definir el color de la temperatura-----------------------------------------------------
  int r = 0, g = 0, b = 0;
  int aux_temp = int(temperatura);
  if(temperatura <= 20){
    r = aux_temp  * 4;
    g = 130 + (aux_temp * 3) ;
    b = 255;
  }
  if((temperatura > 20) && (temperatura <=40) ){
      r = (aux_temp-20)  * 15 ;
      g = 190 + ((aux_temp-20) * 3) +5;
      b = 255;
   }
   if((temperatura > 40) && (temperatura <=60) ){
      r = 255;
      g = 255;
      b = 255 - ((aux_temp-40) * 12);
   }
   if((temperatura > 60) && (temperatura <=100)){
      r = 255;
      g = 255 - ((aux_temp-60)*6);
      b = 15;
   }
   if((temperatura > 100)){
      r = 0;
      g = 80;
      b = 255;
   }
  // ---------------------------------------------------------------------------------------
  fill(r, g, b); // Color rojo para la temperatura
  rect(x_temp + 75, y_temp + 355 - temp_termo, 40, temp_termo,90); // Altura depende de temp

  // Dibujar la base redonda
  fill(r, g, b);
  ellipse(x_temp + 95, y_temp + 360, 60, 60);

}
```

1. Lectura de Sensores

```
// Leemos la humedad relativa
h = dht.readHumidity();
// Leemos la temperatura en grados centígrados (por defecto)
t = dht.readTemperature();

// Comprobamos si ha habido algún error en la lectura
if (isnan(h) || isnan(t)) {
  Serial.println("Error obteniendo los datos del sensor DHT11");
  return;
}

// Calcular el índice de calor en grados centígrados
float hic = dht.computeHeatIndex(t, h, false);
```

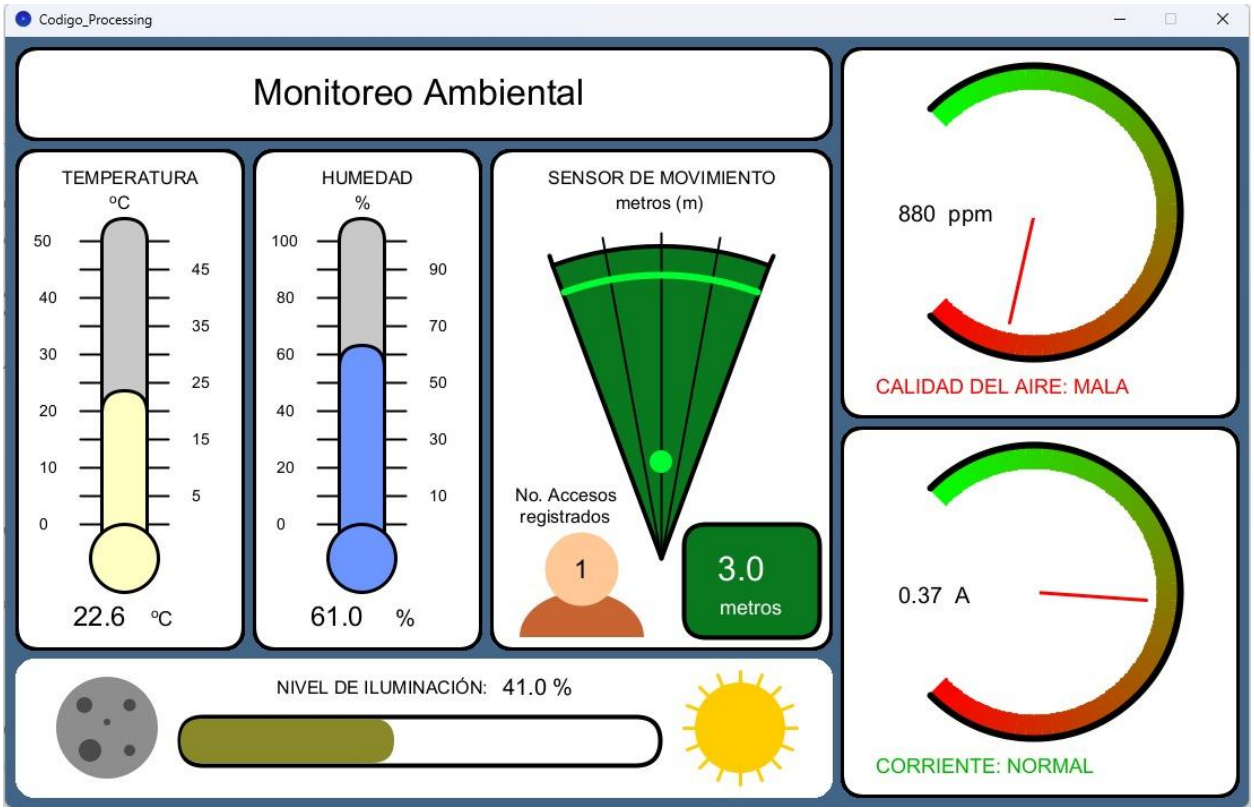
```
//Este es la parte del sensor de oxigeno
int sensorValue = analogRead(A0);
float voltage = sensorValue * (5.0 / 1023.0);
gasVal = voltage * 200; // Conversión aproximada para CO2
Serial.print(gasVal);// Se imprime en la consola el valor en ppm

//Esta es la parte del sensor de corriente
corriente = promedioCorriente(500); //Esto me calcula el promedio de 500 mediciones del sensor de corriente
Serial.print(",");
Serial.print(corriente);
```

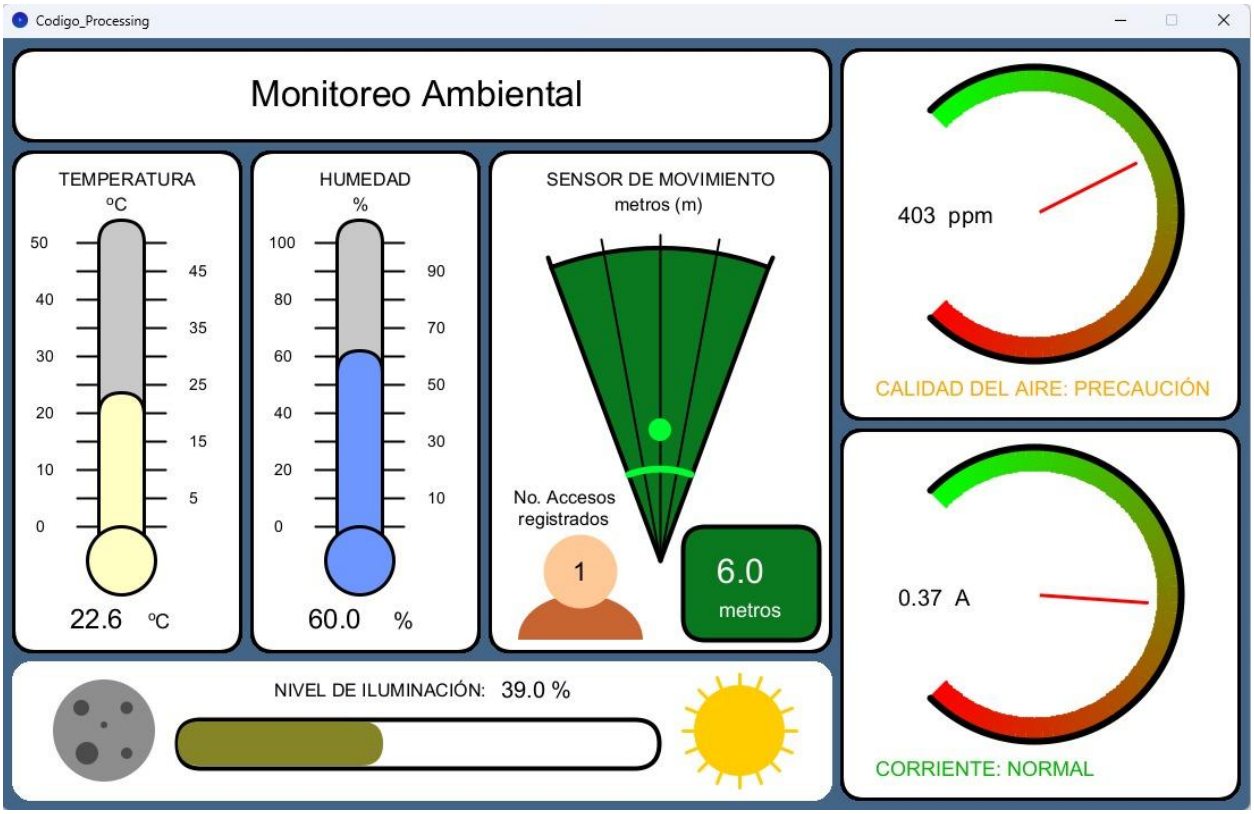
```
// Lectura de corriente
corriente = promedioCorriente(500);

// Enviar datos por Serial
Serial.print(gasVal);
Serial.print(",");
Serial.print(corriente);
Serial.print(",");
Serial.print(h);
Serial.print(",");
Serial.println(t);
```

1. Visualización interfaz grafica



2. Sensores en movimiento



1. Maqueta Fisica

2. Ubicaciones de Sensores

3. Entrada

CONCLUSIONES

La integración de tecnologías IoT permite el monitoreo en tiempo real del ambiente en cuartos de servidores. La adquisición y procesamiento de datos con Arduino aseguran precisión y fiabilidad en las mediciones.
La visualización interactiva en Processing facilita la toma de decisiones oportunas para mantener un entorno seguro.

El almacenamiento en EEPROM respalda el análisis histórico de las variables críticas del ambiente. Las alertas visuales, mediante LEDs y una pantalla LCD, garantizan respuestas inmediatas ante anomalías. La comunicación serial entre Arduino y Processing evidencia la eficacia de la interconexión hardware-software.

La implementación integral del sistema mejora significativamente la gestión de la infraestructura de TI. La sinergia entre Arduino y Processing optimiza el rendimiento y la seguridad del entorno.
Este enfoque modular y escalable abre la posibilidad a futuras mejoras y a la integración de nuevos sensores.

BIBLIOGRAFIA

Banzi, M., & Shiloh, M. (2014). Getting started with Arduino: The open source electronics prototyping platform (3ª ed.). Maker Media, Inc.

Reas, C., & Fry, B. (2007). Processing: A programming handbook for visual designers and artists. MIT Press.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645–1660.