



USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Control de Ambiente

Fase 2

Docente:	Erick Bernal y Samuel Perez
Curso:	Laboratorio de Arquitectura de Computadores y Ensambladores 2
Grupo:	7
Integrantes:	Kevin Estuardo Secaida Molina 201602404 Oscar Eduardo Morales Giron 201603028 Edwin Sandoval Lopez 202010856 Pedro Luis Pu Tavico 202000562

Fecha: 16 de marzo de 2024

Índice

Introducción.....	3
Objetivos.....	4
Funcionamiento.....	5
Sensores.....	5
Medición de temperatura.....	5
Cantidad de luz en el ambiente.....	6
Medición de calidad de aire.....	6
Medición de proximidad.....	6
Interacción con actuador.....	6
Arduino.....	7
Interacción con sensores:.....	7
Wifi.....	7
Comunicación entre arduino y exterior.....	7
MQTT.....	7
Capa de comunicación:.....	7
Plataforma web.....	8
Aplicación web:.....	8
Usos.....	8
Beneficios.....	8
Impacto ambiental.....	8
Fragmentos de código:.....	9
Librerías incluidas:.....	9
Inicialización:.....	9
Bucle Principal:.....	10
Guardado y recuperación de datos de EEPROM:.....	10
Bocetos:.....	11
Boceto de prototipo físico:.....	11
Prototipos propuestos.....	11
Prototipo Simulado.....	11
Prototipo físico.....	12
Prototipo de plataforma web.....	12
Descripción de las capas del Smart Connected Design Framework.....	13
Diagramas de flujo de la información:.....	13
Flujo de datos desde los sensores hasta la plataforma web:.....	13
Flujo de interacción entre la plataforma web y el actuador (ventilador):.....	14

Introducción

Este manual proporciona una guía detallada para implementar, configurar y utilizar una estación meteorológica IoT diseñada específicamente para habitaciones inteligentes. El sistema recopila datos climáticos en tiempo real, como temperatura, humedad, iluminación y concentración de CO₂, utilizando sensores, y los envía a través de Wi-Fi a una plataforma centralizada mediante el protocolo MQTT.

Esta plataforma web permite monitorear condiciones climáticas, controlar dispositivos como ventiladores y analizar datos históricos para optimizar el ambiente. Está diseñado para cualquier persona interesada en IoT para la gestión inteligente del ambiente en hogares y espacios habitacionales. No se requiere un conocimiento previo extenso, ya que se proporcionan explicaciones detalladas paso a paso.

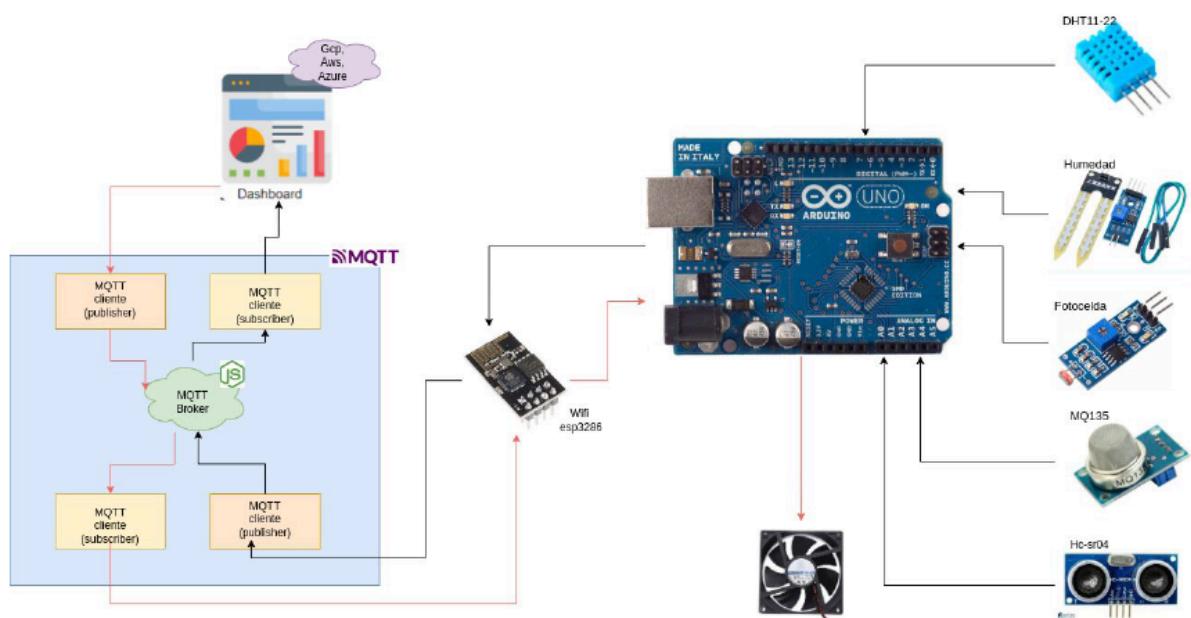
Objetivos

- Proporcionar instrucciones detalladas sobre cómo configurar y utilizar el dispositivo diseñado para medir variables meteorológicas en un dormitorio inteligente, permitiendo a los usuarios obtener información precisa y regular sobre las condiciones ambientales.
- Guiar a los usuarios en el entendimiento y aplicación del framework de IoT utilizado en el dispositivo, facilitando la correcta implementación y gestión de la tecnología necesaria para el monitoreo climático en el dormitorio.
- Explicar cómo interpretar los datos meteorológicos recopilados por el dispositivo y cómo utilizar esa información para predecir patrones climáticos relevantes, permitiendo a los usuarios optimizar el control climático en su dormitorio inteligente para mayor comodidad y eficiencia.
- Proporcionar instrucciones claras sobre cómo utilizar el sistema de visualización de datos para reconocer información y patrones climáticos relevantes, facilitando la comprensión de las condiciones ambientales actuales y pasadas en el dormitorio inteligente.

Funcionamiento

El sistema consiste en una estación meteorológica IoT que recopila datos climáticos como temperatura, iluminación, humedad y concentración de CO₂ en el aire. Estos datos son enviados a una plataforma web a través del protocolo MQTT, permitiendo a los usuarios monitorear las condiciones climáticas en tiempo real y accionar sobre dispositivos como un ventilador. Además, la plataforma web muestra datos históricos almacenados en la memoria EEPROM del microcontrolador Arduino.

A continuación se muestra el diagrama del funcionamiento lógico:



Sensores

Medición de temperatura

El dispositivo es capaz de medir la temperatura ambiente y proporcionar los datos precisos sobre la variación térmica, que existe en ese momento. Esto es posible a través del sensor de temperatura DHT11.



Cantidad de luz en el ambiente

El dispositivo es capaz de visualizar la cantidad de luz en el ambiente en una localidad determinada. Esto es posible a través del sensor de luz LDR1.



Medición de calidad de aire

El dispositivo es capaz de medir el CO₂ del ambiente para determinar la calidad del aire. Esto es posible a través del sensor MQ-135.



Medición de proximidad

El dispositivo es capaz de reconocer la proximidad de un objeto (persona), la acción se utilizará para comprobar la presencia o ausencia de un ser humano en la habitación. Esto es posible a través del sensor HC-SR04.



Interacción con actuador

El dispositivo tiene la capacidad de interactuar con un actuador, que en este caso es un ventilador de 5V.



Arduino

Interacción con sensores:

El dispositivo es capaz de comunicarse con los sensores a través de un arduino, para este dispositivo fue utilizado un Arduino Uno.



Wifi

Comunicación entre arduino y exterior

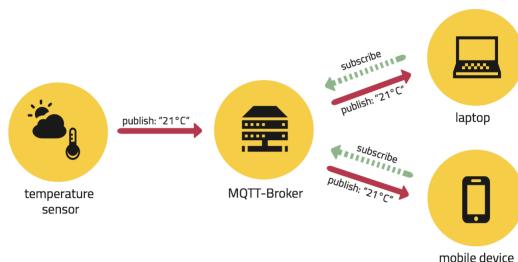
Para la comunicación entre el arduino y el exterior se implementó un módulo wifi ESP82266.



MQTT

Capa de comunicación:

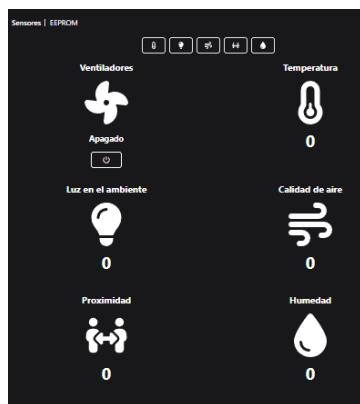
La comunicación de la app web y el dispositivo es posible con el protocolo MQTT ya que este está diseñado para la transferencia de datos entre dispositivos en redes de área local o en internet.



Plataforma web

Aplicación web:

Para controlar el dispositivo se implementó una aplicación web, la cual permite al usuario interactuar con el dispositivo: En la aplicación se muestran los datos recopilados en tiempo real, también es posible acceder a los datos que han sido almacenados en la memoria EEPROM del arduino.



Usos

- Monitoreo en tiempo real de las condiciones climáticas en una habitación inteligente.
- Accionamiento remoto de dispositivos como un ventilador para mantener condiciones confortables.
- Análisis de datos históricos para identificar patrones climáticos y optimizar el control ambiental.

Beneficios

- Mejora del confort en el dormitorio al mantener condiciones climáticas óptimas.
- Ahorro energético al optimizar el uso de dispositivos de climatización.
- Facilita la visualización y comprensión de datos climáticos a través de una plataforma web intuitiva.

Impacto ambiental

- Reducción del consumo energético al controlar eficientemente dispositivos de climatización.
- Mayor conciencia sobre las condiciones ambientales internas y su impacto en la calidad del aire.

Fragmentos de código:

Librerías incluidas:

Estas líneas incluyen las librerías necesarias para el funcionamiento del código, como para trabajar con el sensor DHT11, la comunicación a través de WiFi, y la comunicación serial.

```
#include <Wire.h>
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>
#include <SoftwareSerial.h>
#include <WiFiEsp.h>
#include <WiFiEspClient.h>
#include <PubSubClient.h>
```

Inicialización:

En la función `setup()`, se realizan las configuraciones iniciales necesarias, como la inicialización de los sensores, la configuración de los pines y la apertura de las comunicaciones seriales.

```
void setup() {
    //Inicializaciones de los dispositivos.
    Serial.begin(9600);
    mySerial.begin(9600);
    dht.begin();
    // Configuración de pines
    pinMode(trig, OUTPUT);
    pinMode(echo, INPUT);
    pinMode(MT, INPUT);
    pinMode(LA, INPUT);
    pinMode(MC, INPUT);
    pinMode(MP, INPUT);
    pinMode(GE, INPUT);
    pinMode(ME, INPUT);
    pinMode(FAN, OUTPUT) ;
    // Interrupciones
    attachInterrupt(digitalPinToInterrupt(GE), estado_guardarEeprom, FALLING);
    attachInterrupt(digitalPinToInterrupt(ME), estado_mostrarEeprom, FALLING);
    // Configuración de la LCD.
    lcd.init();
    lcd.backlight();
    lcd.print("ACE2 GRUPO 7");

    delay(2000); // Retraso de 2 segundos para salir del setup.
}
```

Bucle Principal:

En el bucle principal se verifica si hay datos disponibles para leer en el puerto serial (`mySerial.available() > 0`), lo que indica una posible solicitud MQTT entrante.

Si se reciben datos, se leen y se procesan para realizar acciones específicas. Por ejemplo, si se recibe "0", se apaga el ventilador; si se recibe "1", se enciende el ventilador; y si se recibe "2", se realizan lecturas de sensores y se envían los datos a través de la comunicación serial.

También se realizan lecturas de los sensores de humedad, temperatura, CO₂, luminosidad y proximidad, como también se calcula la distancia utilizando el sensor de ultrasonidos.

```
void loop() {
    //Acciones realizadas por petición MQTT
    if (mySerial.available() > 0) {
        String receivedData = mySerial.readString();
        receivedData.trim(); // Elimina espacios en blanco al principio y al final

        if (receivedData.equals("0")) {
            digitalWrite(FAN, LOW);
            Serial.println("Apagando el ventilador");
        } else if (receivedData.equals("1")) {
            digitalWrite(FAN, HIGH);
            Serial.println("Encendiendo el ventilador");
        } else if (receivedData.equals("2")) {
            // Variables de almacenamiento para las lecturas de los sensores
            float humedad = dht.readHumidity();
            float temp = dht.readTemperature();
            int raw_data = analogRead(A0);
            int lum_data = analogRead(A1);
            long t;
            long d;

            digitalWrite(trig, HIGH);
            delayMicroseconds(10);
            digitalWrite(trig, LOW);
        }
    }
}
```

Guardado y recuperación de datos de EEPROM:

Estas funciones se encargan de guardar y recuperar datos en la memoria EEPROM del Arduino. Esto permite almacenar datos de forma permanente incluso después de apagar el dispositivo.

```
void guardarDatos(float dist_val, float lum_val, float co2_val, float temp_val, float hum_val) {
    int direccion = 0;
    EEPROM.put(direccion, dist_val);
    direccion += sizeof(float);
    EEPROM.put(direccion, lum_val);
    direccion += sizeof(float);
    EEPROM.put(direccion, co2_val);
    direccion += sizeof(float);
    EEPROM.put(direccion, temp_val);
    direccion += sizeof(float);
    EEPROM.put(direccion, hum_val);
}

void recuperarDatos(float& dist_val, float& lum_val, float& co2_val, float& temp_val, float& hum_val) {
    int direccion = 0;
    EEPROM.get(direccion, dist_val);
    direccion += sizeof(float);
    EEPROM.get(direccion, lum_val);
    direccion += sizeof(float);
    EEPROM.get(direccion, co2_val);
    direccion += sizeof(float);
    EEPROM.get(direccion, temp_val);
    direccion += sizeof(float);
    EEPROM.get(direccion, hum_val);
}
```

MQTT

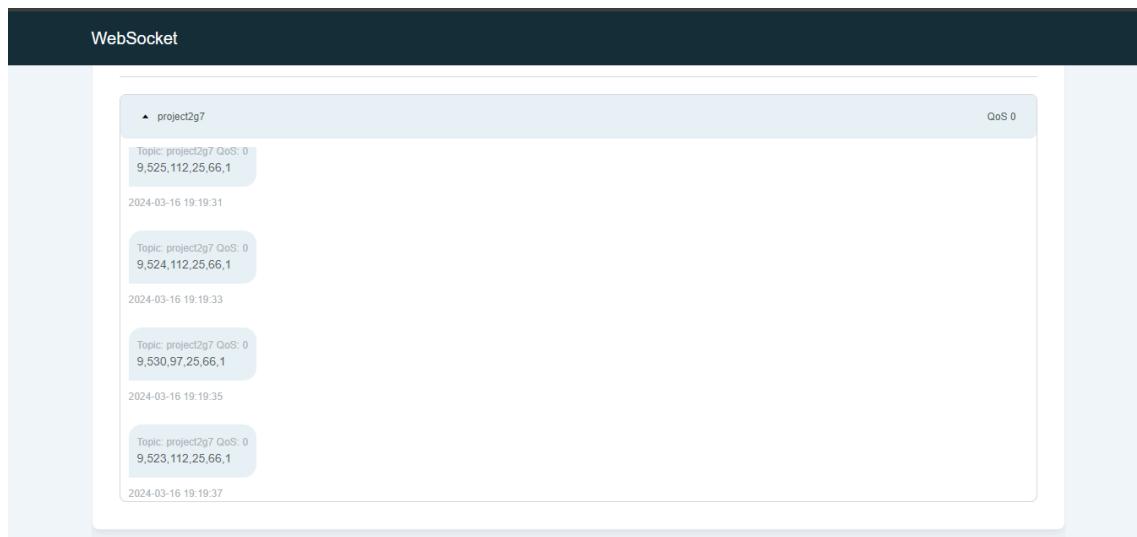
Configuración para la conexión MQTT

```
// MQTT Broker settings
const int mqtt_port = 8883;                                // MQTT port (TLS)
const char *mqtt_broker = "ebc4ebe4.ala.us-east-1.emqxsl.com"; // EMQX broker endpoint
const char *mqtt_topic = "project2g7";                      // MQTT topic
const char *mqtt_username = "edwin";                         // MQTT username for authentication
const char *mqtt_password = "edwin";                         // MQTT password for authentication

// NTP Server settings
```

Topic

En la siguiente imagen se muestra la recepción de datos enviados desde un Arduino en un topic nombrado "Project2g7".



Bocetos:

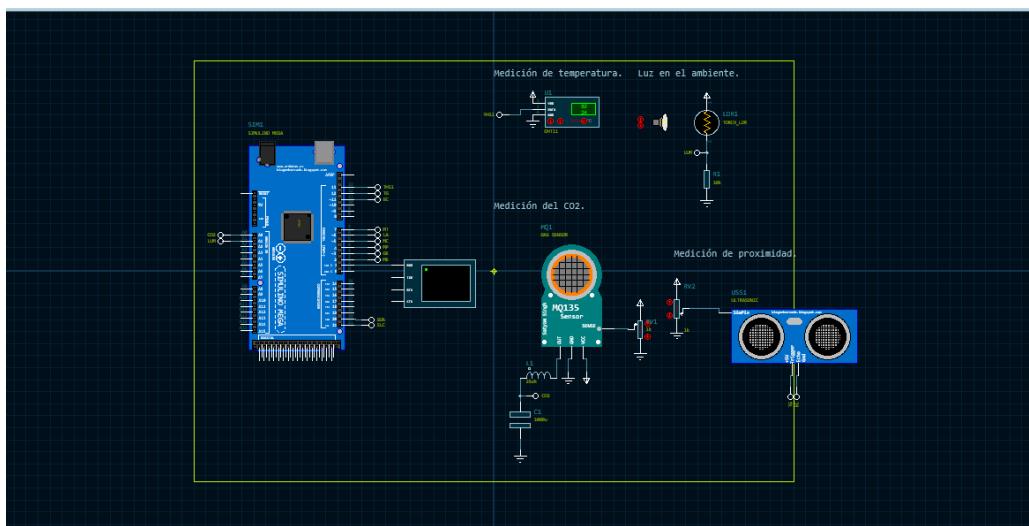
Boceto de prototipo físico:



El boceto representa el diseño del dispositivo físico el cual consiste en un microcontrolador Arduino conectado a sensores de temperatura, humedad, iluminación y CO₂, así como a un módulo ESP8266 para la comunicación Wi-Fi.

Prototipos propuestos

Prototipo Simulado



El prototipo simulado consiste en un microcontrolador Arduino conectado a sensores de temperatura, humedad, iluminación y CO₂, así como a un módulo ESP8266 para la comunicación Wi-Fi.

Prototipo físico



El prototipo físico consiste en un microcontrolador Arduino conectado a sensores de temperatura, humedad, iluminación y CO₂, así como a un módulo ESP8266 para la comunicación Wi-Fi.

Prototipo de plataforma web

A screenshot of a web-based monitoring interface titled "Monitoreo de sensores". At the top, there is a configuration section with fields for Host (localhost:8884), Port (8884), ClientId (esp8266_8C1190), Username (johan), Password (johan), Topic (publish), Payload ({"temperatura": "0"}), and several buttons: Conectar, Desconectar, Suscribir, Desuscribir, Publicar, and Loop.send 2. Below this, there is a section titled "Sensores | EEPROM" with five data cards: Ventiladores (Apagado), Temperatura (0), Luz en el ambiente (0), Calidad de aire (0), Proximidad (0), and Humedad (0). Each card has a small icon representing the sensor type.

El prototipo de la plataforma web muestra datos en tiempo real y permite el control del ventilador a través de botones virtuales.

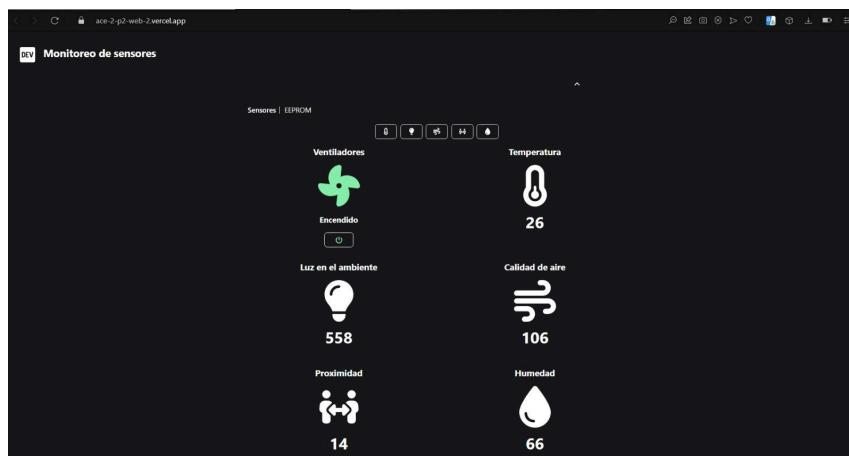
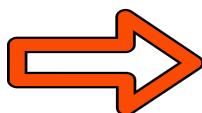
Descripción de las capas del Smart Connected Design Framework

1. Capa de Hardware (Fase 1): Incluye el microcontrolador Arduino, sensores de temperatura, humedad, iluminación y CO₂, así como un módulo ESP8266 para la conectividad Wi-Fi.
2. Capa de Software (Fase 1): En esta fase se desarrollaron algoritmos para la adquisición de datos de los sensores y el envío de estos datos a la memoria EEPROM del Arduino.
3. Capa de Comunicación (Fase 2): Se implementa el protocolo MQTT para la comunicación entre el Arduino y la plataforma web, permitiendo la transferencia de datos en tiempo real.
4. Capa de experiencia de Usuario (Fase 3): Para esta fase se utilizó Grafana para mostrar todos los datos procesados en las capas anteriores, este software ocupa una conexión a la base de datos en este caso PostgreSQL para poder almacenarlos y ya luego solo que el usuario pueda observar en tiempo real lo que está recibiendo de los diferentes sensores.

Diagramas de flujo de la información:

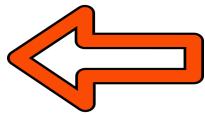
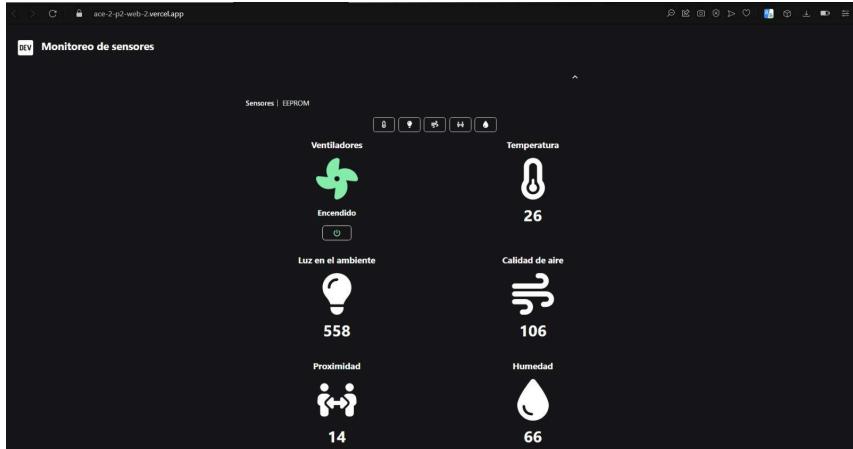
Flujo de datos desde los sensores hasta la plataforma web:

Sensores -> Arduino -> Wi-Fi -> MQTT -> Plataforma Web



Flujo de interacción entre la plataforma web y el actuador (ventilador):

Plataforma Web -> Wi-Fi -> MQTT -> Arduino -> Actuador (Ventilador)



Flujo de interacción entre la plataforma web, la base de datos y el software Grafana:

Plataforma web → ApiRest → PostgreSQL → Grafana.

