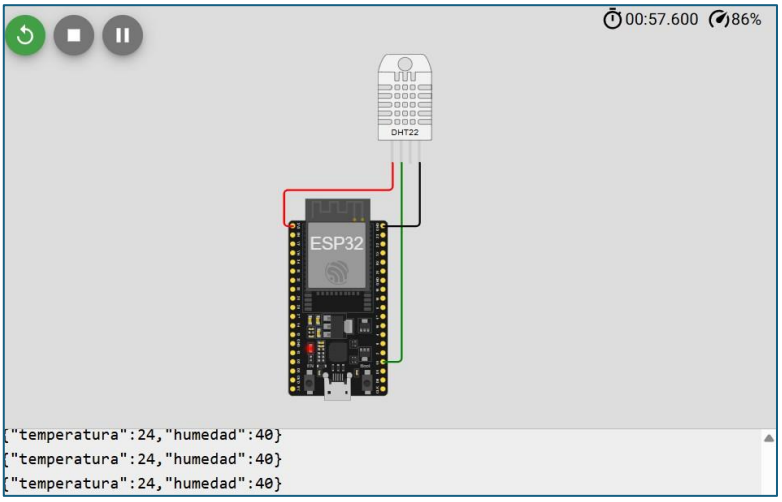
Consigna:

4) Implementar un código JASON, para comunicar un sensor de temperatura y humedad con un ESP32, Arduino, simulando los mismos en WOKWI, Proteus, LabView; etc. ¿Cuáles serían los campos mínimos para hacer la implementación?



Link: https://wokwi.com/projects/429516622590449665

Objetivo

El objetivo de este programa es leer datos de temperatura y humedad desde un sensor **DHT22** conectado a un **ESP32**, y mostrar los datos en formato **JSON** a través del **Monitor Serie**. Esta implementación permite una comunicación clara y estructurada que puede ser útil para integrar con otras plataformas, como aplicaciones móviles, sistemas IoT o bases de datos.

Creación y Envío del JSON

```
StaticJsonDocument<128> json;
json["temperatura"] = temp;
json["humedad"] = hum;

serializeJson(json, Serial);
Serial.println();
```

Ejemplo de salida esperada –

Estos son los campos mínimos en un código JSON para comunicar un sensor de temperatura y humedad con un ESP32

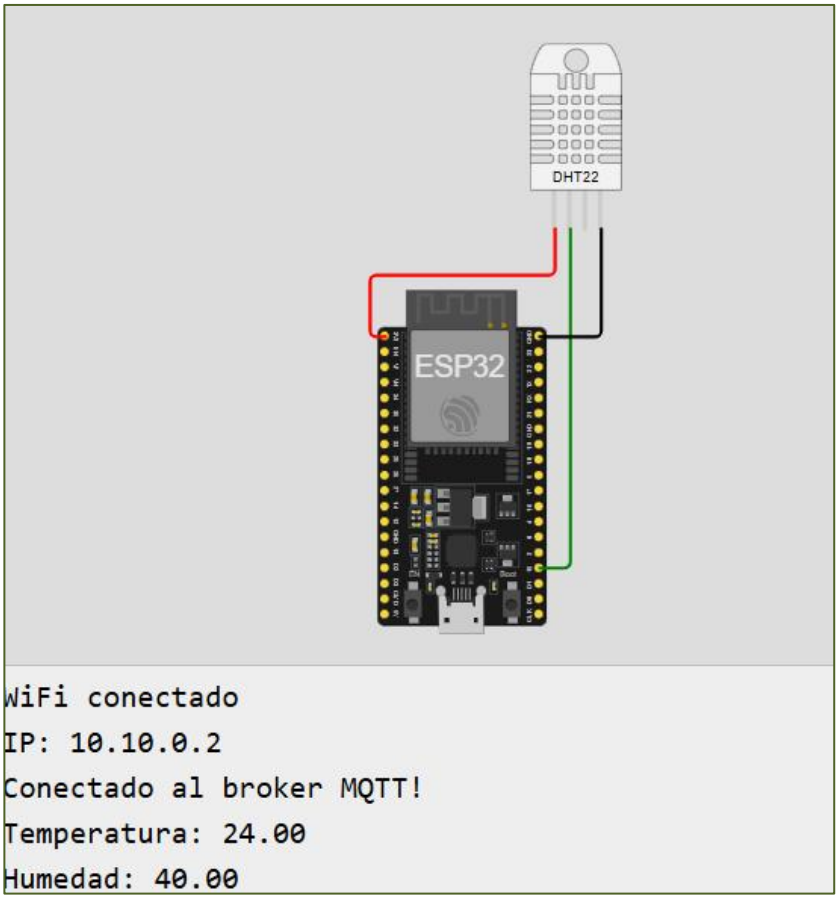
```
{"temperatura": 24.3, "humedad": 56.8}
```

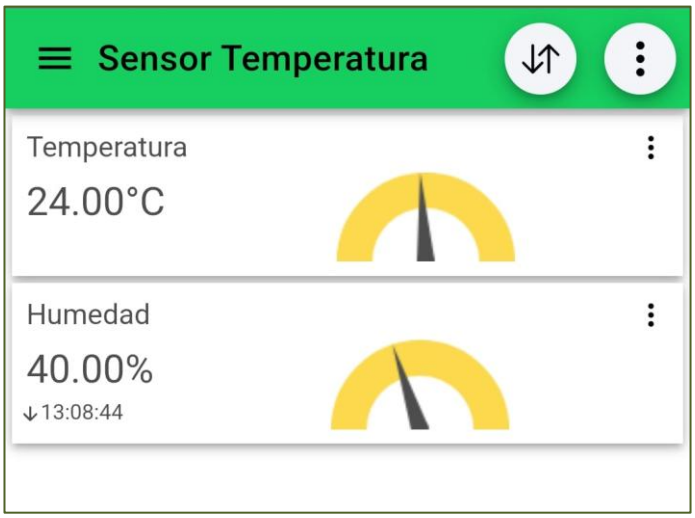
Código completo

```
#include <DHT.h>
#include <ArduinoJson.h>

#define DHTPIN 15
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  dht.begin();
}

void loop() {
  delay(2000); // espera 2 segundos entre lecturas

  float temp = dht.readTemperature();
  float hum = dht.readHumidity();

  if (isnan(temp) || isnan(hum)) {
    Serial.println("Error al leer el sensor");
    return;
  }

  StaticJsonDocument<128> json;
  json["temperatura"] = temp;
  json["humedad"] = hum;

  serializeJson(json, Serial);
  Serial.println();
}
```

IMPLEMENTACION DE MQTT – APP IOT MQTT PANEL

Link: https://wokwi.com/projects/428993783396750337





```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

#define DHTPIN 15         // Pin GPIO donde conectas el sensor DHT
#define DHTTYPE DHT22     // Tipo de sensor
DHT dht(DHTPIN, DHTTYPE);

// Configuración WiFi y MQTT
const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* mqtt_server = "test.mosquitto.org"; // Broker público de
pruebas

WiFiClient espClient;
PubSubClient client(espClient);

const long interval = 2000; // Intervalo para enviar datos (2 segundos)

void setup_wifi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi conectado");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
}

void reconnect() {
  while (!client.connected()) {
    String clientId = "ESP32Client-" + String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("Conectado al broker MQTT!");
    } else {
      delay(5000);
    }
  }
}
void publishData(float temp, float hum) {
  char tempString[8];
  char humString[8];

  dtostrf(temp, 6, 2, tempString);
  dtostrf(hum, 6, 2, humString);

  client.publish("iot/temperatura", tempString);
  client.publish("iot/humedad", humString);
}
```

```
void setup() {
  Serial.begin(115200);
  dht.begin();
  setup_wifi();
  client.setServer(mqtt_server, 1883); // Conectar al broker MQTT
}

void loop() {
  if (!client.connected()) reconnect();
  client.loop();

  static unsigned long lastMsg = 0;
  if (millis() - lastMsg > interval) {
    lastMsg = millis();

    float temp = dht.readTemperature();
    float hum = dht.readHumidity();

    if (!isnan(temp) && !isnan(hum)) {
      Serial.print("Temperatura: ");
      Serial.println(temp);
      Serial.print("Humedad: ");
      Serial.println(hum);
      publishData(temp, hum); // Publicar datos
    } else {
      Serial.println("Error leyendo del sensor DHT!");
    }
  }
}
```