Práctica IoT – versión simulada – UD2 Sesión 3

Simular un proyecto IoT completo **desde el navegador**, sin sensores reales:

Sensor virtual → microcontrolador simulado → Internet → plataforma cloud → panel de datos + alerta automática

Herramientas necesarias

Tipo	Herramienta	Descripción
Simulador IoT	Wokwi	Simula placas Arduino, ESP32, sensores, WiFi, etc.
Plataforma Cloud	Adafruit IO	Guarda y muestra datos (dashboard, gráficos).
Notificaciones	Sistema de alertas integrado en Adafruit IO	Envía emails o cambia color de widgets.

Parte 1 – Simular el sensor y el microcontrolador

1. Entra en https://wokwi.com
2. Crea un nuevo proyecto → elige **ESP32** (plantillas).
3. Borra el código por defecto y pega este código adaptado para **simular temperatura**:

```
#include <WiFi.h>
#include <WiFiClientSecure.h>

// --- WiFi (Wokwi) ---
const char* ssid     = "Wokwi-GUEST";
const char* password = "";

// --- Adafruit IO (REST HTTPS) ---
const char* host      = "io.adafruit.com";
const int   httpsPort = 443;
const char* aioUser   = "tu_usuario";           // tu usuario (no email)
const char* aioKey    = "tu_key"; // tu Active Key
const char* feedKey    = "nombre_de_tu_feed";               // Feed Key
EXACTA (Settings del feed)

WiFiClientSecure client;

bool httpWaitForData(WiFiClientSecure& c, uint32_t ms = 6000) {
  unsigned long start = millis();
  while (!c.available() && millis() - start < ms) delay(10);
  return c.available();
}
```

```
String httpRequest(const String& req) {
  client.setInsecure();
  if (!client.connect(host, httpsPort)) {
    return "ERR:CONNECT";
  }
  client.print(req);
  if (!httpWaitForData(client)) {
    client.stop();
    return "ERR:NOREPLY";
  }
  String resp;
  while (client.available()) resp += char(client.read());
  client.stop();
  return resp;
}

String httpStatus(const String& resp) {
  int s1 = resp.indexOf(' ');
  int s2 = resp.indexOf(' ', s1 + 1);
  if (s1 < 0 || s2 <= s1) return "???";
  return resp.substring(s1 + 1, s2); // ej. "200"
}

// ---- LISTAR FEEDS (diagnóstico de credenciales) ----
```

```
bool listFeedsOnce() {
  String url = String("/api/v2/") + aioUser + "/feeds";
  String req =
    String("GET ") + url + " HTTP/1.1\r\n" +
    "Host: " + host + "\r\n" +
    "User-Agent: ESP32-Wokwi\r\n" +
    "Connection: close\r\n" +
    "X-AIO-Key: " + String(aioKey) + "\r\n\r\n";

  String resp = httpRequest(req);
  Serial.println("===== LISTA FEEDS (GET) =====");
  Serial.println(resp);
  Serial.println("==============================");

  return resp.indexOf("200 OK") >= 0;
}

// ---- COMPROBAR/CREAR FEED SI NO EXISTE ----
bool ensureFeedExists() {
  // Intento rápido: GET al feed específico
  String urlFeed = String("/api/v2/") + aioUser + "/feeds/" + feedKey;
  String getReq =
    String("GET ") + urlFeed + " HTTP/1.1\r\n" +
    "Host: " + host + "\r\n" +
```

```
  "User-Agent: ESP32-Wokwi\r\n" +
  "Connection: close\r\n" +
  "X-AIO-Key: " + String(aioKey) + "\r\n\r\n";

String getResp = httpRequest(getReq);
String st = httpStatus(getResp);
Serial.println("===== GET FEED =====");
Serial.println(getResp);
Serial.println("====================");

if (st.startsWith("2")) {
  // Ya existe
  return true;
}

// Si 404, lo creamos
if (st == "404") {
  String urlCreate = String("/api/v2/") + aioUser + "/feeds";
  // Puedes enviar solo "name"; si quieres definir la key explícita, usa {"name":"Temperatura","key":"temperatura"}
  String body = String("{\"name\":\"") + feedKey + "\",\"key\":\"" + feedKey + "\"}";
  String postReq =
    String("POST ") + urlCreate + " HTTP/1.1\r\n" +
    "Host: " + host + "\r\n" +
```

```
    "User-Agent: ESP32-Wokwi\r\n" +
    "Connection: close\r\n" +
    "Content-Type: application/json\r\n" +
    "X-AIO-Key: " + String(aioKey) + "\r\n" +
    "Content-Length: " + String(body.length()) + "\r\n\r\n" +
    body;

  String postResp = httpRequest(postReq);
  Serial.println("===== CREATE FEED (POST) =====");
  Serial.println(postResp);
  Serial.println("================================");

  String st2 = httpStatus(postResp);
  return (st2 == "200" || st2 == "201");
 }

 // Cualquier otro status (401/403/429...)
 return false;
}

// ---- PUBLICAR VALOR EN FEED ----
bool postValueToAIO(float value) {
 String url  = String("/api/v2/") + aioUser + "/feeds/" + feedKey + "/data";
 String body = String("{\"value\":") + String(value, 2) + "}";
```

```
  String req =
    String("POST ") + url + " HTTP/1.1\r\n" +
    "Host: " + host + "\r\n" +
    "User-Agent: ESP32-Wokwi\r\n" +
    "Connection: close\r\n" +
    "Content-Type: application/json\r\n" +
    "X-AIO-Key: " + String(aioKey) + "\r\n" +
    "Content-Length: " + String(body.length()) + "\r\n\r\n" +
    body;

  String resp = httpRequest(req);
  String st = httpStatus(resp);

  Serial.println("===== RESPUESTA POST =====");
  Serial.println(resp);
  Serial.println("==========================");

  return st.startsWith("2");
}

// ---- WIFI / SETUP ----
void connectWiFi() {
  Serial.print("Conectando a WiFi");
```

```
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(300);
  }
  Serial.println("\n✅ WiFi OK");
}

void setup() {
  Serial.begin(115200);
  connectWiFi();

  bool feedsOk = listFeedsOnce();
  Serial.println(feedsOk ? "✅ Credenciales AIO válidas (GET feeds 200 OK)" :
                           "❌ Revisa usuario/AIO Key (GET feeds NO es 200)");

  bool feedReady = ensureFeedExists();
  Serial.println(feedReady ? "✅ Feed listo" : "❌ Feed no existe y no se pudo crear (mira el log arriba)");

  Serial.println("Publicaré valores cada 5s…");
}

void loop() {
```

```
  float t = 20.0 + (float)(esp_random() % 1600) / 100.0; // 20.00..35.99
  Serial.print("🌡️ Temp simulada = ");
  Serial.println(t, 2);

  bool ok = postValueToAIO(t);
  Serial.println(ok ? "📡 Enviado a AIO ✅\n" : "❌ Fallo al enviar (revisa 'RESPUESTA POST')\n");

  delay(7000);
}
```

Parte 2 – Configurar Adafruit IO

1. Entra en https://io.adafruit.com
2. Crea una cuenta gratuita.
3. Ve a **Feeds** → **New Feed** → **“temperatura”**.
4. Copia tus **credenciales AIO**:
    - Usuario → TU_USUARIO
    - Clave AIO → TU_CLAVE_AIO
      (las pegas en el código del paso anterior).
5. Pulsa **Run** en Wokwi.

Verás en el monitor serie:

Conectado a WiFi.

Temperatura simulada: 29

Dato enviado a Adafruit IO.

En Adafruit IO → abre el feed “temperatura” verás los valores subir.

También, puedes crear un nuevo dashboard y vincularlo con el feed temperatura.

Este programa simula un proyecto de Internet de las Cosas (IoT) con un sensor virtual de temperatura. Primero, el ESP32 se conecta al WiFi de Wokwi, lo que le permite comunicarse por Internet.

Luego accede a la cuenta de Adafruit IO usando el nombre de usuario y la clave secreta (AIO Key), que funcionan como una contraseña.

Después, el programa comprueba si existe un espacio de datos llamado **temperatura** (feed).

Si no existe, lo crea automáticamente. Cada pocos segundos genera un valor de temperatura aleatorio entre 20 y 36 grados y lo envía a Adafruit IO, donde puede verse en tiempo real como una lista o un gráfico.

En el monitor del programa aparecen mensajes que confirman la conexión, el envío correcto y el valor de cada temperatura simulada. De esta forma, se puede observar cómo un dispositivo IoT envía información desde un “sensor” virtual hacia la nube, igual que haría un dispositivo real conectado a Internet.

Mi ejemplo:

https://wokwi.com/projects/445867519710619649