Instructivo de instalación y configuración de la comunicación MQTT

1. Descargar el instalador “mosquitto-2.x.x-install-win64.exe” desde la página oficial, ejecútarlo y aceptar todas las opciones por defecto; el broker y los clientes de línea de comandos quedarán en C:\Program Files\mosquitto y funcionarán en el puerto 1883 con acceso anónimo, suficiente para pruebas en una red local.

2. Abrir una consola nueva y lanzar el broker escribiendo

```
mosquitto
```

Verás un mensaje que confirma que el servidor está escuchando; deja esa ventana abierta para que el servicio siga activo.

3. Abrir otra consola y suscríbirte a todos los topics del proyecto con

```
mosquitto_sub -h localhost -v -t "iot/#" -t "camara/foto"
```

El modificador -v hace que se muestren tanto el nombre del topic como el contenido del mensaje, lo que facilita verificar que temperatura, humedad, niveles de agua y la imagen lleguen correctamente.

4. Compila y graba en un ESP32 con sensor DHT22 este sketch que publica cada cinco segundos en iot/ambiente; solo cambia SSID, contraseña y la IP de tu PC con el comando ipconfig y seleccionando el ipv4 de el adaptador wifi

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

#define DHTPIN 4
#define DHTTYPE DHT22

const char* ssid = "TU_SSID";
const char* pass = "TU_CLAVE";
const char* broker = "IP_BROKER";

WiFiClient espClient;
PubSubClient client(espClient);
```

```
18.DHT dht(DHTPIN, DHTTYPE);
19.
20.void setup() {
21.  Serial.begin(115200);
22.  WiFi.begin(ssid, pass);
23.  while (WiFi.status() != WL_CONNECTED) delay(500);
24.  client.setServer(broker, 1883);
25.  dht.begin();
26.}
27.
28.void loop() {
29.  if (!client.connected()) client.connect("esp32_dht");
30.  client.loop();
31.
32.  float t = dht.readTemperature();
33.  float h = dht.readHumidity();
34.
35.  if (!isnan(t) && !isnan(h)) {
36.    String payload = "[{\"temperatura\":" + String(t,1) +
   ",\"humedad\":" + String(h,1) + "}]";
37.    client.publish("iot/ambiente", payload.c_str());
38.  }
39.
40.  delay(5000);
41.}
42.
```

5. Grabar en otro ESP32 el sketch que reporta con texto **MAXIMO** o **VACIO** el estado de dos flotadores conectados a los pines 34 y 35; publica el JSON resultante en iot/nivel.

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

const char* ssid = "TU_SSID";
const char* pass = "TU_CLAVE";
const char* broker = "IP_BROKER";

WiFiClient espClient;
PubSubClient client(espClient);

#define SENSOR_HOR 34
#define SENSOR_VER 35
```

```
String estado(int pin) {
  return digitalRead(pin) ? "MAXIMO" : "VACIO";
}

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) delay(500);
  client.setServer(broker, 1883);
  pinMode(SENSOR_HOR, INPUT);
  pinMode(SENSOR_VER, INPUT);
}

void loop() {
  if (!client.connected()) client.connect("esp32_nivel");
  client.loop();

  String payload = "{\"horizontal\":\"" + estado(SENSOR_HOR) +
"\",\"vertical\":\"" + estado(SENSOR_VER) + "\"}";
  client.publish("iot/nivel", payload.c_str());

  delay(4000);
}
```

6. Para una ESP32-CAM que envíe fotografías JPEG como base 64 cada 20 segundos al topic cámara/foto, usar este sketch; necesita la librería **Base64** (disponible en el gestor de librerías) y los pines corresponden al módulo AI-Thinker.

```
#include "esp_camera.h"
#include <WiFi.h>
#include <PubSubClient.h>
#include "base64.h"

const char* ssid = "TU_SSID";
const char* pass = "TU_CLAVE";
const char* broker = "IP_BROKER";

WiFiClient espClient;
PubSubClient client(espClient);

void startCamera() {
  camera_config_t c;
  c.ledc_channel = LEDC_CHANNEL_0;
  c.ledc_timer = LEDC_TIMER_0;
```

```
  c.pin_d0 = 17; c.pin_d1 = 35; c.pin_d2 = 34; c.pin_d3 = 5;
  c.pin_d4 = 39; c.pin_d5 = 18; c.pin_d6 = 36; c.pin_d7 = 19;
  c.pin_xclk = 0; c.pin_pclk = 21; c.pin_vsync = 25; c.pin_href = 23;
  c.pin_sscb_sda = 26; c.pin_sscb_scl = 27;
  c.pin_pwdn = 32; c.pin_reset = -1;
  c.xclk_freq_hz = 20000000;
  c.pixel_format = PIXFORMAT_JPEG;
  c.frame_size = FRAMESIZE_VGA;
  c.jpeg_quality = 10;

  esp_camera_init(&c);
}

void setup() {
  Serial.begin(115200);
  startCamera();
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) delay(500);
  client.setServer(broker, 1883);
}

void loop() {
  if (!client.connected()) client.connect("esp32_cam");
  client.loop();

  camera_fb_t *fb = esp_camera_fb_get();
  if (!fb) return;

  String img64 = base64::encode(fb->buf, fb->len);
  String payload = "" + img64;
  client.publish("camara/foto", payload.c_str());

  esp_camera_fb_return(fb);
  delay(20000);
}
```

7.Mientras el broker está ejecutándose y los dispositivos envían datos, la consola del subscriber mostrará cada mensaje y tu backend en Python, conectado a MongoDB y GridFS, procesará temperatura, humedad, niveles de agua y guardará las imágenes asociadas, permitiendo consultarlas después mediante los endpoints que ya tienes implementados.