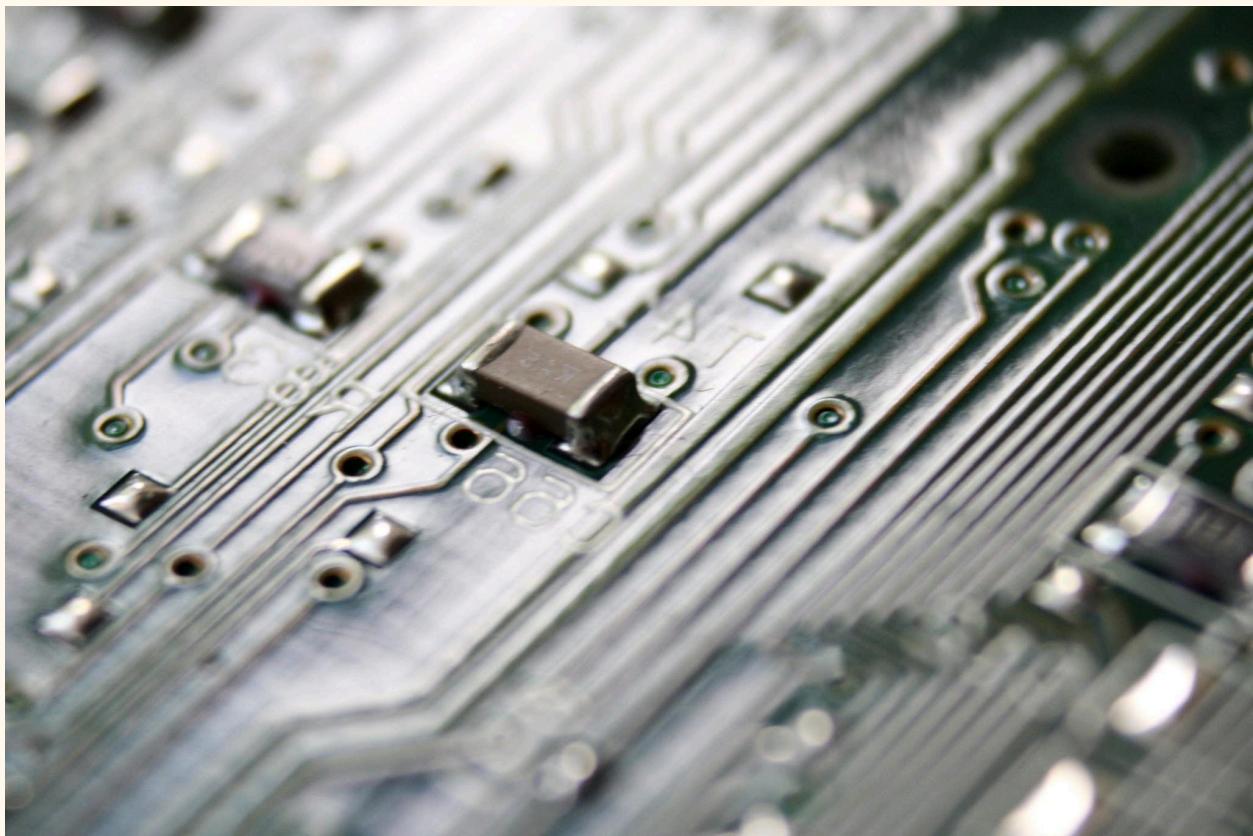


Michael Cristian Itzep Ixcayau 201610975
Luigi Anderson López Suy 202104727
Fernando Andhré González Espinoza 202202055
Vasti Abigail González Pereira 202300652

16/12/2025
Vac. Dic. 2do Semestre

Proyecto 1: Sistema de Monitoreo Data Center - FIUSAC

Manual Técnico



1. Introducción y Objetivos

Este documento detalla los aspectos técnicos del desarrollo e implementación del sistema de monitoreo y seguridad para el Data Center de la Facultad de Ingeniería de la USAC.

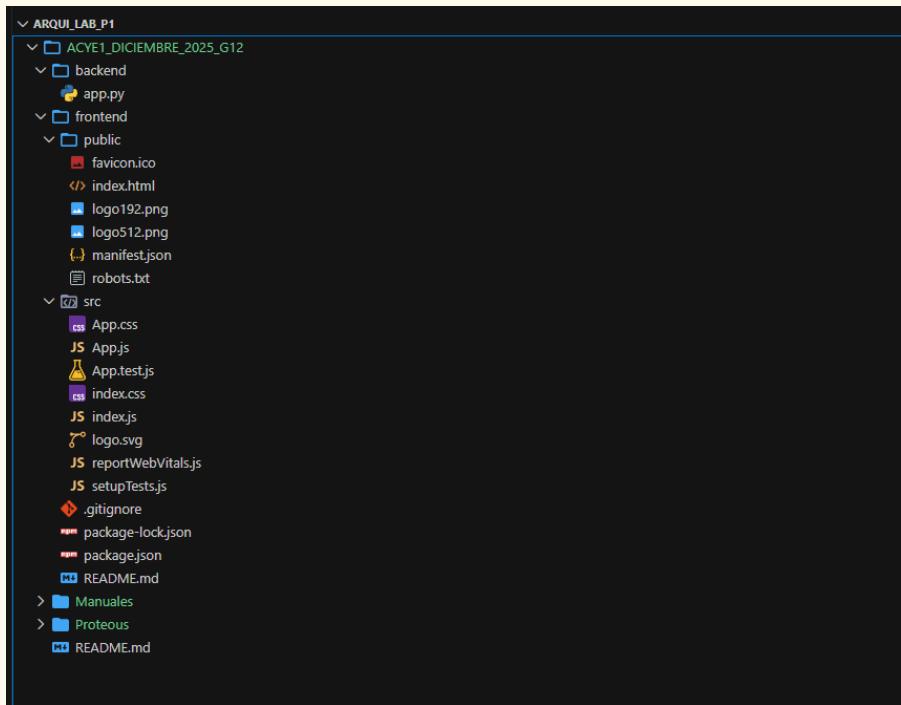
Objetivo General: Diseñar un sistema IoT robusto basado en Raspberry Pi que integre sensores ambientales y de movimiento, gestionado a través de una arquitectura web con comunicación MQTT y almacenamiento en la nube.

Alcance Técnico: El sistema monitorea temperatura y humedad en tiempo real (24/7), controla el acceso físico mediante servo-actuadores, gestiona ventilación automática y detecta intrusiones, registrando cada evento para auditoría forense.

2. Arquitectura del Sistema

El proyecto se basa en una arquitectura de tres capas:

1. **Capa Física (Edge):** Raspberry Pi 4 modelo B interactuando con sensores (DHT11/22, PIR) y actuadores (Ventiladores, Servos, LEDs, Buzzer).
2. **Capa de Comunicación y Datos:**
 - **Broker MQTT :** Para la transmisión de mensajes en tiempo real entre el hardware y la web.
 - **MongoDB:** Base de datos en la nube para el almacenamiento histórico de registros.
3. **Capa de Aplicación:** API REST (Backend) y Aplicación Web (Frontend) con dashboard interactivo.



3. Especificaciones Técnicas y Materiales

Hardware utilizado para la implementación del prototipo :

Componente	Especificación	Función
Controlador	Raspberry Pi 4B	Procesamiento central y conexión WiFi.
Sensor Ambiental	DHT11 / DHT22	Lectura de temperatura y humedad.
Sensor Movimiento	PIR	Detección de presencia e intrusiones.
Actuador Puerta	Servo Motor	Mecanismo de apertura/cierre de puerta.
Ventilación	Motor DC 5V / Ventilador	Sistema de enfriamiento activo.
Indicadores	LED RGB y LEDs Blancos	Estado del sistema, alarmas e iluminación.
Interfaz Local	LCD 16x2 / 20x4 (I2C)	Visualización de datos in-situ.

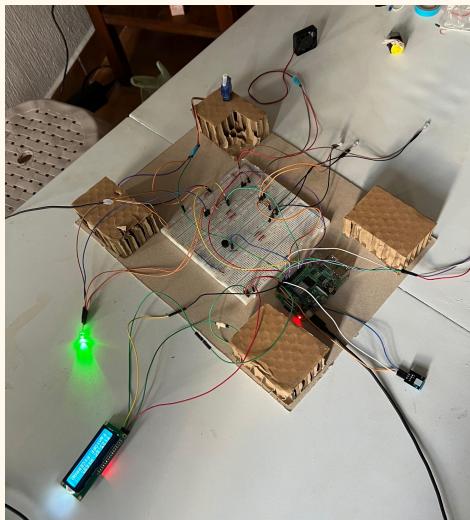
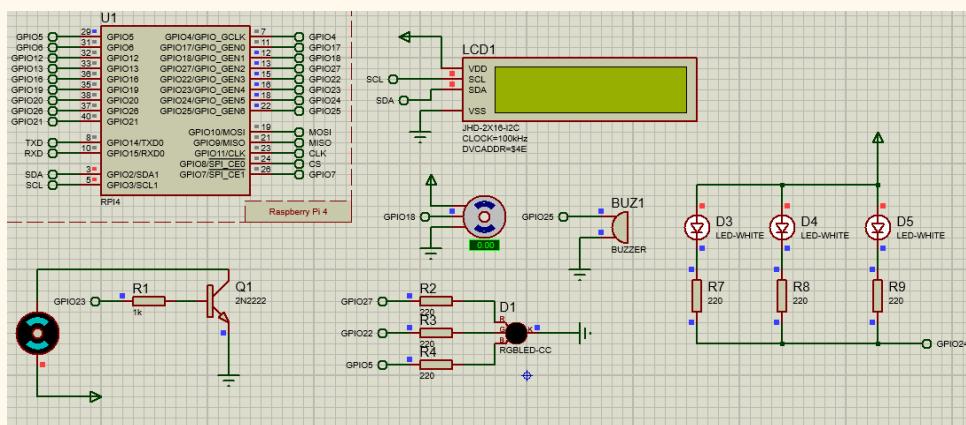
Sonido	Buzzer (Activo/Pasivo)	Alertas sonoras diferenciadas.
--------	---------------------------	--------------------------------

4. Diagrama de Conexiones (Electrónica)

El circuito se diseñó utilizando una protoboard para la integración de componentes. Se utilizaron transistores NPN (2N2222) para la conmutación de cargas inductivas como el ventilador.

Puntos clave de conexión:

- Sensores:** Conectados a pines GPIO configurados como entrada.
- I2C:** Pantalla LCD conectada a los pines SDA y SCL de la Raspberry Pi.
- PWM:** Control del Servo Motor para posicionamiento preciso (Abierto/Cerrado).



5. Configuración de Servicios

5.1 Protocolo MQTT

Se definieron los siguientes Tópicos (Topics) para la comunicación Pub/Sub :

- **Sensores:** `/fiusac/datacenter/temperature`,
`/fiusac/datacenter/humidity`
- **Alertas:** `/fiusac/datacenter/alerts`
- **Actuadores:** `/fiusac/datacenter/access` (Puerta), `/fiusac/datacenter/fan` (Ventilador)
- **Estado:** `/fiusac/datacenter/status`, `/fiusac/datacenter/maintenance`

5.2 Estructura de Base de Datos (MongoDB)

La base de datos NoSQL se organizó en las siguientes colecciones :

Temperature / Humidity:

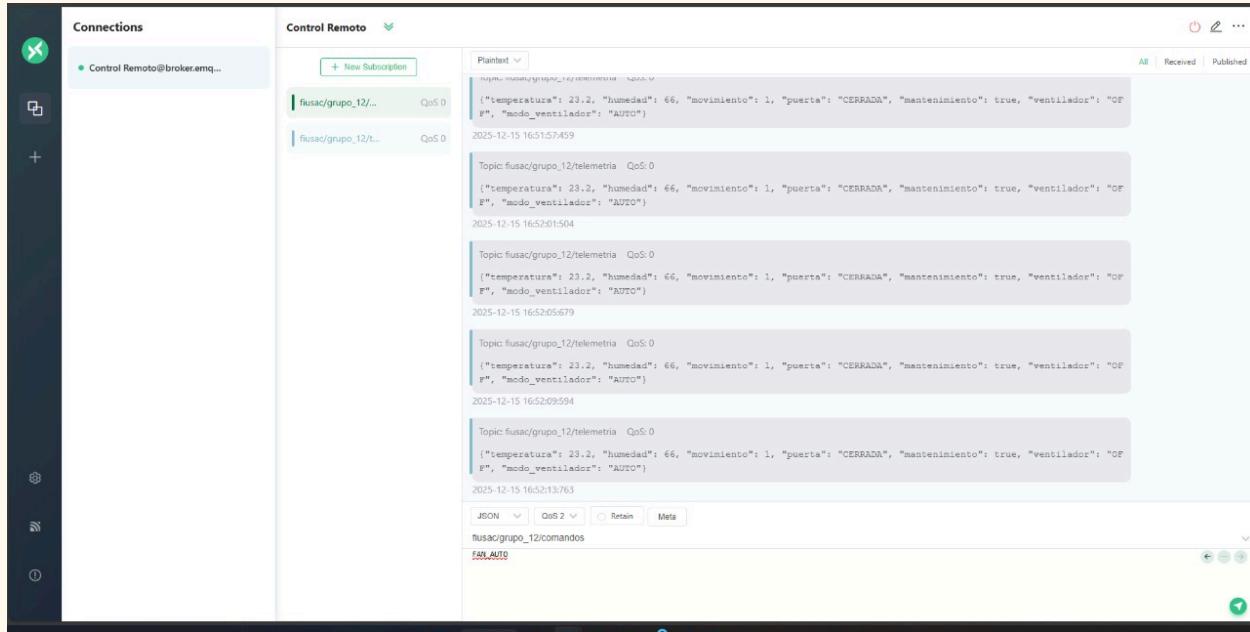
```
{ "timestamp": "ISO8601", "value": 26.5, "unit": "celsius" }
```

Events: Registra alarmas críticas.

```
{ "type": "alarm_temperature", "description": "...", "duration": 120 }
```

Access_logs: Auditoría de apertura de puertas.

Intrusion_logs: Eventos de seguridad (movimiento con puerta cerrada).



6. Endpoints de la API REST

El servidor Backend expone los siguientes puntos de acceso para la aplicación web :

Consultas (GET):

`/api/temperature`: Historial de temperatura.

`/api/events`: Listado general de eventos y alarmas.

`/api/stats`: Estadísticas del día (máximos, mínimos, conteos).

Control (POST):

`/api/door/open`: Envía comando MQTT para abrir puerta.

`/api/maintenance/toggle`: Activa/Desactiva el modo mantenimiento.

`/api/fan/toggle`: Control manual del ventilador.

The screenshot shows the MongoDB Atlas Data Explorer interface. On the left, the navigation pane displays the organization (LUIGI ANDERSON's...) and project (Project O). Under 'Data Explorer', there are sections for 'My Queries', 'Data Modeling', and 'CLUSTERS (0)'. A search bar for clusters is also present. The main area shows the 'kiwi-BD > fiusec_datacenter > eventos' collection. The 'Documents' tab is selected, showing 60 documents. The first document in the list is expanded, revealing its structure:

```

{
  "_id": "ObjectId('69408f737d7421111aad4c3')",
  "fecha": "2025-12-15T16:44:18.337+00:00",
  "tipo_evento": "humedad",
  "descripcion": "humedad Alta",
  "valor_registro": 73
}

```

Below this, three more documents are listed in collapsed form. At the bottom of the interface, there are buttons for 'Add Data', 'Update', and 'Delete', along with pagination controls (100, 1-56 of 56).

7. Explicación del Código (Lógica de Control)

El script principal de Python ejecutado en la Raspberry Pi implementa la siguiente lógica de autómata:

Monitoreo Ambiental (Loop Principal):

- Lectura cada 30 segundos.
- **Si Temp >=24 C:** Activa ventilador (GPIO HIGH).
- **Si Temp >=28 C:** Estado de ALARMA CRÍTICA (LED Verde Parpadeante + Buzzer Continuo).
- **Si Humedad >=70%:** Estado de ALERTA HUMEDAD (LED Amarillo + Buzzer Intermitente).

Sistema de Seguridad (Interrupciones):

- El sensor PIR monitorea constantemente.
- **Lógica de Intrusión:** SI (Puerta == CERRADA) Y (Movimiento == DETECTADO) Y (Modo Mantenimiento == OFF) -> **ACTIVAR ALARMA** (LED Rojo/Azul + Notificación MQTT).
- **Iluminación:** Al detectar presencia, activa LEDs blancos por 2 segundos mediante temporizado.

7. Explicación del Código (Lógica de Control)

El script principal de Python ejecutado en la Raspberry Pi implementa la siguiente lógica de autómata:

Monitoreo Ambiental (Loop Principal):

- Lectura cada 30 segundos.
- **Si Temp >=24 C:** Activa ventilador (GPIO HIGH).
- **Si Temp >=28 C:** Estado de ALARMA CRÍTICA (LED Verde Parpadeante + Buzzer Continuo).
- **Si Humedad >=70%:** Estado de ALERTA HUMEDAD (LED Amarillo + Buzzer Intermitente).

Sistema de Seguridad (Interrupciones):

- El sensor PIR monitorea constantemente.
- **Lógica de Intrusión:** SI (Puerta == CERRADA) Y (Movimiento == DETECTADO) Y (Modo Mantenimiento == OFF) -> **ACTIVAR ALARMA** (LED Rojo/Azul + Notificación MQTT).
- **Iluminación:** Al detectar presencia, activa LEDs blancos por 2 segundos mediante temporizador.

8. Problemas Encontrados y Soluciones

- **Problema:** Ruido en las lecturas del sensor DHT11.
- **Solución:** Se implementó un algoritmo de promedio simple en Python para descartar valores atípicos antes de enviarlos a la base de datos.
 - **Problema:** Latencia en la recepción de comandos MQTT.
- **Solución:** Se ajustó la calidad de servicio (QoS) a nivel 1 y se optimizó la conexión a internet de la Raspberry Pi.
 - **Problema:** "Rebote" o falsos positivos en el sensor PIR.
- **Solución:** Se ajustó la sensibilidad física del sensor (potenciómetro) y se añadió un pequeño *delay* en el código para confirmar la detección.

9. Costos del Prototipo

Tabla detallada de inversión en materiales:

Cantidad	Descripción	Costo Unitario (Q)	Total (Q)
1	Raspberry Pi (Modelo X)	Q965.00	Q965.00
1	Sensor DHT11	Q. 25.00	Q. 25.00
1	Servo Motor MG90S	Q. 30.00	Q. 30.00
1	Pantalla LCD con I2C	Q. 50.00	Q. 50.00
X	Componentes varios (LEDs, Resistencias)	Q. 50.00	Q. 50.00
1	Protoboard	Q. 79.00	Q. 79.00
TOTAL			Q. 1199.00

Explicación de Código

1. CONFIGURACIÓN MQTT

```

14     MQTT_BROKER = "broker.emqx.io"
15     MQTT_PORT = 1883
16     TOPICO_PUBLICAR = "fiusac/grupo_12/telemetria"
17     TOPICO_SUSCRIBIR = "fiusac/grupo_12/comandos"
18
19     client_mqtt = mqtt.Client()
20

```

2. CONFIGURACIÓN MONGODB

```

24     MONGO_URI = "mongodb+srv://admin_fiusac:0306@kiwi-bd.xi6ztan.mongodb.net/?retryWrites=true&w=majority&appName=kiwi-BD"
25     mongo_ok = False
26
27     print("--- INICIANDO SISTEMA FIUSAC V2.0 ---")
28     print("1. Conectando a Base de Datos...")
29
30     try:
31         client_mongo = pymongo.MongoClient(MONGO_URI, serverSelectionTimeoutMS=3000)
32         client_mongo.server_info()
33         db = client_mongo["fiusac_datacenter"]
34         collection_temp = db["temperatura"]
35         collection_hum = db["humedad"]
36         collection_events = db["eventos"]
37         mongo_ok = True
38         print("[BD] Conectado Correctamente")
39     except Exception as e:
40         print(f"[BD] Error: {e}")

```

3. CONFIGURACIÓN DE PINES

```

45     GPIO.setmode(GPIO.BCM)
46     GPIO.setwarnings(False)
47
48     PIR_PIN = 17
49     SERVO_PIN = 18
50     FAN_PIN = 23
51     BUZZER_PIN = 27
52     LEDS_BLANCOS_PIN = 22
53     BOTON_MANT_PIN = 26
54
55     RGB_RED = 5
56     RGB_GREEN = 6
57     RGB_BLUE = 13
58
59     GPIO.setup(PIR_PIN, GPIO.IN)
60     GPIO.setup(FAN_PIN, GPIO.OUT)
61     GPIO.setup(BUZZER_PIN, GPIO.OUT)
62     GPIO.setup(LEDS_BLANCOS_PIN, GPIO.OUT)
63     GPIO.setup([RGB_RED, RGB_GREEN, RGB_BLUE], GPIO.OUT)
64     GPIO.setup(BOTON_MANT_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
65
66     GPIO.setup(SERVO_PIN, GPIO.OUT)
67     servo_pwm = GPIO.PWM(SERVO_PIN, 50)
68     servo_pwm.start(0)
69
70     dht_device = adafruit_dht.DHT11(board.D4)
71     try:
72         lcd = LCD()
73     except:
74         lcd = None
75

```

4. VARIABLES GLOBALES

```

79     TEMP_UMBRAL_FAN = 24
80     TEMP_UMBRAL_ALERTA = 28
81     HUM_UMBRAL_ALERTA = 70
82
83     puerta_abierta = False
84     modo_mantenimiento = False
85
86     # NUEVAS VARIABLES PARA CONTROL DE VENTILADOR
87     fan_modo_manual = False # False = Automatico, True = Manual
88     fan_estado_manual = False # True = ON, False = OFF
89
90     # Temporizadores
91     tiempo_ultimo_log_mongo = 0
92     INTERVALO_MONGO = 30
93     tiempo_ultimo_mqtt = 0
94     INTERVALO_MQTT = 3
95     ultimo_evento_tipo = ""
96     tiempo_ultimo_evento = 0
97

```

5. FUNCIONES MQTT (Lógica Remota)

```
102  def on_connect(client, userdata, flags, rc):
103      if rc == 0:
104          print("[MQTT] Conectado al Broker")
105          client.subscribe(TOPICO_SUSCRIBIR)
106      else:
107          print(f"[MQTT] Fallo conexión: {rc}")
108
109  def on_message(client, userdata, msg):
110      """
111          Comandos: ABRIR, CERRAR, MANT_ON, MANT_OFF
112          NUEVOS: FAN_ON, FAN_OFF, FAN_AUTO
113      """
114
115      global puerta_abierta, modo_mantenimiento, fan_modo_manual, fan_estado_manual
116      orden = msg.payload.decode().upper()
117      print(f"\n> [ORDEN REMOTA] Recibido: {orden}")
118
119      # --- CONTROL PUERTA ---
120      if orden == "ABRIR":
121          puerta_abierta = True
122          mover_servo("ABRIR")
123          registrar_evento("control_remoto", "Puerta Abierta Web")
124      elif orden == "CERRAR":
125          puerta_abierta = False
126          mover_servo("CERRAR")
127          registrar_evento("control_remoto", "Puerta Cerrada Web")
128
129      # --- CONTROL MANTENIMIENTO ---
130      elif orden == "MANT_ON":
131          modo_mantenimiento = True
132          registrar_evento("control_remoto", "Mantenimiento ON Web")
133      elif orden == "MANT_OFF":
134          modo_mantenimiento = False
135          registrar_evento("control_remoto", "Mantenimiento OFF Web")
```

```
136      # --- CONTROL VENTILADOR (NUEVO) ---
137      elif orden == "FAN_ON":
138          fan_modo_manual = True
139          fan_estado_manual = True
140          registrar_evento("control_remoto", "Ventilador FORZADO ENCENDIDO")
141          print("    -> Fan en MODO MANUAL: ON")
142
143      elif orden == "FAN_OFF":
144          fan_modo_manual = True
145          fan_estado_manual = False
146          registrar_evento("control_remoto", "Ventilador FORZADO APAGADO")
147          print("    -> Fan en MODO MANUAL: OFF")
148
149      elif orden == "FAN_AUTO":
150          fan_modo_manual = False
151          registrar_evento("control_remoto", "Ventilador en AUTOMATICO")
152          print("    -> Fan regresó a MODO AUTOMATICO")
153
154      # Iniciar MQTT
155      client_mqtt.on_connect = on_connect
156      client_mqtt.on_message = on_message
157      try:
158          client_mqtt.connect(MQTT_BROKER, MQTT_PORT, 60)
159          client_mqtt.loop_start()
160      except Exception as e:
161          print(f"    [MQTT] Error conectando: {e}")
162
```

6. FUNCIONES AUXILIARES

```

166  def set_rgb(color):
167      GPIO.output([RGB_RED, RGB_GREEN, RGB_BLUE], GPIO.LOW)
168      if color == "VERDE": GPIO.output(RGB_GREEN, GPIO.HIGH)
169      elif color == "ROJO": GPIO.output(RGB_RED, GPIO.HIGH)
170      elif color == "AMARILLO":
171          GPIO.output(RGB_RED, GPIO.HIGH)
172          GPIO.output(RGB_GREEN, GPIO.HIGH)
173      elif color == "AZUL": GPIO.output(RGB_BLUE, GPIO.HIGH)
174      elif color == "OFF": pass
175
176  def mover_servo(estado):
177      duty = 7.5 if estado == "ABRIR" else 2.5
178      GPIO.output(SERVO_PIN, True)
179      servo_pwm.ChangeDutyCycle(duty)
180      time.sleep(0.5)
181      GPIO.output(SERVO_PIN, False)
182      servo_pwm.ChangeDutyCycle(0)
183
184  def lcd_print(linea1, linea2):
185      if lcd:
186          lcd.text(str(linea1), 1)
187          lcd.text(str(linea2), 2)
188
189  def guardar_mongo_sensores(temp, hum):
190      if mongo_ok:
191          try:
192              ahora = datetime.datetime.now()
193              collection_temp.insert_one({"fecha": ahora, "valor_temperatura": float(temp), "unidad": "Celsius"})
194              collection_hum.insert_one({"fecha": ahora, "valor_humedad": float(hum), "unidad": "Porcentaje"})
195              print(">> [MONGO] Sensores guardados (30s)")
196          except: pass
197

```

```

198     def publicar_mqtt(temp, hum, pir, puerta, mant, fan_status, fan_mode):
199         """Envía estado COMPLETO a la Web (Incluyendo Fan)"""
200         try:
201             payload = {
202                 "temperatura": temp,
203                 "humedad": hum,
204                 "movimiento": int(pir),
205                 "puerta": "ABIERTA" if puerta else "CERRADA",
206                 "mantenimiento": mant,
207                 "ventilador": fan_status,      # ON / OFF
208                 "modo_ventilador": fan_mode # AUTO / MANUAL
209             }
210             client_mqtt.publish(TOPICO_PUBLICAR, json.dumps(payload))
211         except: pass
212
213     def registrar_evento(tipo, descripcion, valor_asociado=None):
214         global ultimo_evento_tipo, tiempo_ultimo_evento
215         es_alarma = "alarma" in tipo
216         if es_alarma:
217             if tipo == ultimo_evento_tipo and (time.time() - tiempo_ultimo_evento < 10):
218                 return
219
220         if mongo_ok:
221             try:
222                 doc = {
223                     "fecha": datetime.datetime.now(),
224                     "tipo_evento": tipo,
225                     "descripcion": descripcion,
226                     "valor_registrado": valor_asociado
227                 }
228                 collection_events.insert_one(doc)
229                 print(f">> [MONGO] EVENTO: {tipo} - {descripcion}")
230                 ultimo_evento_tipo = tipo
231                 tiempo_ultimo_evento = time.time()
232             except: pass

```

7. BUCLE PRINCIPAL

```
237     lcd.print("FIUSAC DATA CNTR", "Iniciando...")
238     set_rgb("VERDE")
239     mover_servo("CERRAR")
240     time.sleep(2)
241
242     print("Sistema Listo. Esperando sensores...")
243
244     try:
245         while True:
246             # A. BOTÓN MANTENIMIENTO
247             if GPIO.input(BOTON_MANT_PIN) == GPIO.LOW:
248                 modo_mantenimiento = not modo_mantenimiento
249                 estado_txt = "ACTIVO" if modo_mantenimiento else "INACTIVO"
250                 print(f">> BOTON: Mantenimiento {estado_txt}")
251                 registrar_evento("cambio_modo", f"Mantenimiento {estado_txt} (Boton)")
252                 time.sleep(0.5)
253
254             # B. LÓGICA DE SENSORES
255             try:
256                 temperature = dht_device.temperature
257                 humidity = dht_device.humidity
258                 movimiento = GPIO.input(PIR_PIN)
259
260                 if temperature is None or humidity is None:
261                     time.sleep(0.1)
262                     continue
263
264             # --- LÓGICA DEL VENTILADOR (HÍBRIDA) ---
265             fan_is_on = False
266
267             if fan_modo_manual:
268                 # 1. Modo Manual: Obedece MQTT
269                 if fan_estado_manual:
270                     GPIO.output(FAN_PIN, GPIO.HIGH)
```

```
272             fan_status_txt = "ON (MANUAL)"
273     else:
274         GPIO.output(FAN_PIN, GPIO.LOW)
275         fan_is_on = False
276         fan_status_txt = "OFF (MANUAL)"
277     else:
278         # 2. Modo Automático: Obedece Sensor
279         if temperature >= TEMP_UMBRAL_FAN:
280             GPIO.output(FAN_PIN, GPIO.HIGH)
281             fan_is_on = True
282             fan_status_txt = "ON (AUTO)"
283         else:
284             GPIO.output(FAN_PIN, GPIO.LOW)
285             fan_is_on = False
286             fan_status_txt = "OFF (AUTO)"
287
288         # --- ENVÍO DE DATOS ---
289         tiempo_actual = time.time()
290
291         if tiempo_actual - tiempo_ultimo_log_mongo >= INTERVALO_MONGO:
292             guardar_mongo_sensores(temperature, humidity)
293             tiempo_ultimo_log_mongo = tiempo_actual
294
295         if tiempo_actual - tiempo_ultimo_mqtt >= INTERVALO_MQTT:
296             # Enviamos también el estado del ventilador
297             mode_txt = "MANUAL" if fan_modo_manual else "AUTO"
298             status_simple = "ON" if fan_is_on else "OFF"
299             publicar_mqtt(temperature, humidity, movimiento, puerta_abierta, modo_mantenimiento, status_simple, mode_txt)
300             tiempo_ultimo_mqtt = tiempo_actual
301
302         # --- MODO MANTENIMIENTO ACTIVO ---
303         if modo_mantenimiento:
304             lcd_print("MODO MANTENIMIENTO", "ACTIVO")
305             set_rgb("AMARILLO")
306             # En mantenimiento forzamos apagado de actuadores ruidosos
307             GPIO.output(FAN_PIN, GPIO.LOW)
```

```
308         GPIO.output(BUZZER_PIN, GPIO.LOW)
309         time.sleep(0.5)
310         set_rgb("OFF")
311         time.sleep(0.5)
312         continue
313
314     # --- MODO NORMAL (ALARMAS) ---
315     print(f"T:{temperature}C H:{humidity}% PIR:{movimiento} FAN:{fan_status_txt}")
316
317     # Alarma Intrusión
318     if movimiento and not puerta_abierta:
319         print(">> ALERTA: INTRUSION")
320         lcd_print("! INTRUSION !", "DETECTADA")
321         registrar_evento("alarma_intrusion", "Movimiento detectado")
322         for _ in range(3):
323             set_rgb("ROJO")
324             GPIO.output(BUZZER_PIN, GPIO.HIGH)
325             time.sleep(0.1)
326             set_rgb("AZUL")
327             GPIO.output(BUZZER_PIN, GPIO.LOW)
328             time.sleep(0.1)
329             GPIO.output(LEDS_BLANCOS_PIN, GPIO.HIGH)
330
331     # Alarma Temperatura (Solo visual/sonora, el fan ya se manejó arriba)
332     elif temperature >= TEMP_UMBRAL_ALERTA:
333         lcd_print("! TEMP CRITICA !", f"{temperature}C")
334         registrar_evento("alarma_temperatura", "Temp Critica", temperature)
335         set_rgb("ROJO")
336         GPIO.output(BUZZER_PIN, GPIO.HIGH)
337
338     # Alarma Humedad
339     elif humidity >= HUM_UMBRAL_ALERTA:
340         lcd_print("! HUMEDAD ALTA !", f"{humidity}%")
341         registrar_evento("alarma_humedad", "Humedad Alta", humidity)
342         set_rgb("AMARILLO")
343         GPIO.output(BUZZER_PIN, GPIO.HIGH)
```

```
344             time.sleep(0.2)
345             GPIO.output(BUZZER_PIN, GPIO.LOW)
346
347         # Estado Normal
348     else:
349         set_rgb("VERDE")
350         GPIO.output(BUZZER_PIN, GPIO.LOW)
351         GPIO.output(LED_BIANCO_PIN, GPIO.LOW)
352         estado_puerta_txt = "ABIERTA" if puerta_abierta else "CERRADA"
353         # Mostrar estado del ventilador en LCD para debug visual
354         fan_lcd = "ON" if fan_is_on else "OFF"
355         lcd.print(f"T:{temperature}C H:{humidity}%", f"F:{fan_lcd} P:{estado_puerta_txt}")
356
357         time.sleep(0.5)
358
359     except RuntimeError:
360         continue
361
362     except KeyboardInterrupt:
363         print("\nApagando sistema...")
364         client_mqtt.loop_stop()
365         GPIO.cleanup()
366         if lcd: lcd.clear()
```